The 5th Malaysian
MySEC'2011
Software Engineering Conference

CD PROCEEDING

ISBN 978-967-0194-14-1

CO-LOCATED WORKSHOP
SPISME 2011

1 2 DECEMBER 2011

THE PUTERI PACIFIC
JOHOR BAHRU

ORGANISED BY:

UTM
UNIVERSITI TEKNOLOGI MALAYSIA

Research Alliance on K-Economy (RAKE)

Faculty of Computer Science and Information Systems

Advanced Informatics School (AIS)

Horst Lichter, Toni Anwar
(eds.)

International Workshop on CMMI based Software Process Improvement in Small and Medium Sized Enterprises

(SPISME 2011)

Workshop Proceedings

12 December 2011

Puteri Pacific Hotel
Johor Bahru, Malaysia

Organizers

- Toni Anwar, UTM Johor Bahru, Malaysia
- Firdaus Harun, UTM Johor Bahru, Malaysia
- Suhaimi bin Ibrahim, UTM Kuala Lumpur, Malaysia
- Simona Jeners, RWTH Aachen University, Germany
- Chumpol Krootkaew, NECTEC Bangkok,Thailand
- Tachanun Kangwantrakool, ISEM Nonthaburi, Thailand
- Horst Lichter, RWTH Aachen University, Germany
- Apinporn Methawachananon, NECTEC Bangkok, Thailand
- Mohd. Naz'ri Mahrin, UTM Kuala Lumpur, Malaysia
- Chayakorn Piyabunditkul, NECTEC Bangkok, Thailand
- Shamsul Sahibuddin, UTM Kuala Lumpur, Malaysia
- Ali Selamat, UTM Johor Bahru, Malaysia
- Taratip Suwannasart, Chulalongkorn University Bangkok, Thailand

Preface

The Workshop on CMMI based Process Improvement in Small and Medium Sized Enterprises aims at gathering together re-searchers and practitioners to discuss experiences in the application of CMMI in industrial software organizations. CMMI is one of the most accepted process improvement approaches. In software development CMMI-DEV is applied by many organizations. SEI reports that 60% of the appraised organizations are small and medium sized organizations. Another report shows that this number is increasing. Because a CMMI based process improvement program takes time and is expensive many small and medium size software development organizations are still facing problems in applying CMMI.

One goal of this workshop is to exchange experience on how to set up, apply and organize a CMMI based process improvement processes in small and medium size enterprises.

Horst Lichter and Toni Anwar
Aachen/Johor Bahru, December 2011

Table of Contents

Software Processes in an Agile World

Horst Lichter

RWTH Aachen University, Research Group Software Construction
`lichter@swc.rwth-aachen.de`

Abstract. In this paper we relate classical software process models to new agile development processes and software process improvement. We argue that there is no single process model that always fits and that organizations have to re-use the best out of classical and agile processes. Furthermore we question “classical” software process improvement because it is often done isolated from people and technology issues. Finally, we present ten propositions about software process models and software process improvement.

Keywords: software process model, software process improvement, agile development

1 Introduction

Software development is a complex, challenging, and creative task. Although we have more than 50 years of software development experience and we are living in world where software is playing an outstanding and important role, we do not have one single completely sound Software Engineering approach that always leads to high quality software. But there are plenty of methods, languages, and techniques that have proven to be successfully applicable in industrial software development (nevertheless, they are not applied always and everywhere). Software development processes play an important role in software development. The IEEE Software Engineering Glossary [3] defines this term as follows:

software development process — The process by which user needs are translated into a software product. The process involves translating user needs into software requirements, transforming the software requirements into design, implementing the design in code, testing the code, and sometimes, installing and checking out the software for operational use.
Note: These activities may overlap or be performed iteratively.

We always should clearly distinguish the concepts *software process model* and *software process*. A software process model generically describes tasks and artifacts of software development. It serves as a template that can be instantiated many times in different projects; we call a concretely instantiated software process model a software process [9].

There are two different kinds of software process models: *abstract* and *concrete* ones. Abstract software process models describe as a matter of principle how to organize software development (e.g., Software Life Cycle, Prototyping, and Spiral Model). Hence, abstract process models cannot be instantiated in a project off the shelf. In contrast, concrete software process models, e.g. the Unified Process [6], define all aspects that are needed to instantiate the process model in a project (usually after some tailoring). That means a concrete software process model defines all activities, all roles and all artifacts that have to be established in a project.

Software Process Improvement (SPI) has been introduced in 1990th by the Software Engineering Institute (SEI) as a systematical and continuous approach to improve the maturity of software process models based on the Capability Maturity Model (CMM). Since then, the CMM has evolved to the CMMI [4] and SPI has become very popular in the software industry.

In this paper we want to relate classical software process models to new agile process models and SPI. The paper is organized as follows: At first we present a brief historical overview and describe the commonalities of agile development approaches. Then we focus on how to get the right software process model and present our concerns regarding classical SPI. We conclude by formulating ten propositions about software process models and SPI.

2 Software Process Models – A Brief Historical Overview

Software process models have been introduced in Software Engineering in the 1970th when developers encountered that the problems that were solved by means of software had become so complex that software could no longer be developed by a single person in a single step (namely programming). The historical paper of Kron and DeRemer [2] who introduced the term *Programming in the Large* reflects this situation pretty well. Hence, new software development organization rules were needed. An important milestone in the development of process models is the so called *Waterfall Model* which was introduced by Walter Royce [1]. For the very first time, this model identifies all major software development activities or steps. Although there is lots of criticism of this model and today we know that it does not fit for innovative developments, it has influenced and impelled Software Engineering a lot. To overcome the drawbacks of the Waterfall Model, Berry Boehm introduced the so called *Spiral Model* [5]. It focuses on development risks and arranges the development process in a way that the top level risks are solved first and so forth. With this in mind, risk driven development as proposed by the Spiral Model is another important contribution of the early process models to Software Engineering.

In the 1990th it was encountered that a sequential process model does not appropriately reflect typical development scenarios where developers are not able to capture all requirements at the beginning and the operation of a software systems changes its own requirements. To cope with changing requirements *iterative* and *incremental* software process models have been introduced. The core idea of both iterative and incremental software development is (1) to create a solution quickly and to revise it systematically based on user feedback until it fits to the user's needs and (2) not to develop the complete functionality in one big project but to evolve the functionality step by step. Typically, iterative and incremental process models include *Prototyping* as a means to cheaply build an early version of the system that can be assessed by the users [10]. The Unified Process [6] and the V-Model XT [13] are well known iterative, prototyping-based, and incremental process models. For instance, in UP each increment development is divided into four development phases; inside each phase the work is done iteratively based on prototypes.

3 Agile Software Development

In response to software process models aiming to organize software development completely by defining lots of roles, activities and documents (e.g. UP) a group of important software engineers have published the so called *Agile Manifesto* [8] in 2001. Agile approaches expect that during the implementation users discover the possibilities of the system, adjust requirements, and want to profit immediately from the system. The aim is to deliver operating functionality of the software as fast as possible, starting with the components that are most important for the user.

Hence, the main idea of agile software development is to concentrate on the users' needs and to develop solutions quickly without unnecessary overhead (i.e. documents). Since then, a series of agile software process models has been proposed. Some of them are more or less abstract frameworks for software development like Crystal; others have been used intensively in industry like Extreme Programming or Scrum. A detailed review can be found in [7]. Although these process models are very diverse they share the following common characteristics [9]:

- Development is done iteratively; the development cycles are not longer than three month (preferably much shorter).
- The size of the project team is small (typically six to ten people); the team usually shares one or two offices (i.e., the development is not distributed).
- The client is integrated into the team and most of the time available.
- The work is organized and assigned by the team itself.
- Documentation is reduced to a minimum; the most important result is a running system.

Agile process models are often called *light-weight* in order to clearly distance them from documentation-centric or *heavy-weight* process models, e.g. UP.

All in all, nowadays software organizations have at command a tool-box filled with a couple of very different process models. Therefore, software organizations have to decide which process model or which portfolio of process models to apply. This decision is not a simple one because the applied process model heavily influences the organization, its projects, its culture and especially its developers.

4 The Search for the Best Software Process Model

The choice of the “right” software process model is crucial for each software development organization. Based on published and own experiences it can be stated, that there is no single software process model that fits for all organizations. But, what are the most important criteria that influence the choice of a software process model? Again, there is no single answer. The following aspects and questions should be taken into consideration when choosing a software process model:

- What kind of software do we develop (e.g., embedded software, information systems, safety critical software)?
- Do we always develop similar software of the same application domain?
- What is the size of our typical projects and project teams?
- Do we develop software at one site or is the development distributed over different sites?
- What are the skills of our developers (e.g. programming, quality assurance, etc.)?
- Do we have close relationships to the customer?
- Are we the principal contractor or are we a sub-contractor?
- Are we free to choose a software process model or do we have to apply the one that the customer requires?

This list of aspects is far from being complete. But, it demonstrates that there are lots of impact factors regarding the choice of a software process model. These factors might be conflicting and might not be assessed in isolation. Thus, an organization has to make compromises. A large organization that runs many different projects in different domains in parallel cannot expect that one single process model is sufficient. It needs a portfolio of process models with defined tailoring options in order to cover its project variety. One the other hand, a small software organization that is specialized in one specific domain may be able to manage its projects based on one or two very similar process models. Sometimes the discussion whether to apply classical or agile software processes is a “religious” one. But, there is no single truth. Obviously, organizations have to combine the best of classical process models and agile ones. It would be careless not to re-use everything that has been proven to effectively support software development. Fortunately, we do have choices and we should base our decisions on explicit impact factors and on all available software process models.

To sum up, each organization is responsible for selecting and applying those process models that support their business goals and that fit best to their specific project constraints. A selected software process model has to support the project team; it should define all those aspects that are really needed to successfully complete projects. Everything else is “dead freight” that costs resources and does not contribute to project success. Organizations should never apply process models for their own sake!

As the world is changing and software development is changing as well, each organization has to monitor its software processes continuously in order to know whether they still fit or whether they have to be revised or exchanged by new ones.

5 Process Improvement – Valuable Investment or Waste

Software process improvement (SPI) has been promoted by SEI and some other protagonists for many years. Based on reference models like CMMI, SPICE or ITIL process assessment and improvement has become a big business. Beside this economical perspective, each organization has to answer the following questions: (1) How much SPI is needed? (2) What SPI measures have to be performed? (3) How to organize and manage SPI in the organization? As there are many publications addressing these issues (e.g. [11], [12]) we want to focus on other important aspects that are related to SPI. We see the following three close connections:

- First, the applied software process has to support the project and has to be flexibly customizable to the project's needs. A process model is only beneficially if all involved persons (i.e. developers and managers) accept it; i.e., they have to be convinced that the process model supports their work and contributes to the project's success.
- Second, all technologies (i.e. languages, methods and tools) applied in a project have to be selected in a way, that they enable the team to develop the software efficiently and effectively.
- Third, all team members have to have all skills needed to apply the software process model as well as the technologies.

Hence, a software process model should not be considered and improved in isolation; it is always embedded in an organization and linked to people and technologies. This implies that each SPI measure should always take into consideration both the people and the technology dimension. Therefore, each SPI measure should not only define how to improve a special process (e.g. test process) but also include a training and technology component (e.g. training on test methods, selection of appropriate test tools). If an organization does not improve process, people and technology in sync, SPI is waste, otherwise is might be a valuable investment. A last remark: Very often organizations decide to go the CMMI way. And sometimes this leads – besides high costs – to frustrated developers with decreased motivation. If SPI is implemented wrongly it is counterproductive and finally produces "scorched earth".

6 Conclusion

Instead of a traditional conclusion and outlook paragraph we want to conclude this paper with ten propositions:

1. Organizations have to apply process models in order to successfully run development projects!

2. There are many process models that have proven to support software development!
3. There is and there will be no process model that fits to all project conditions!
4. Organizations have to choose actively and explicitly process models that best fit to their specific environment and conditions!
5. A process model has to support and not to burden the project team!
6. A process model is beneficial if and only if it is accepted by the developers!
7. Applied process models have to be evolved or exchanged by new ones, because the world is changing!
8. Organizations have to combine the best out of classical and agile models!
9. SPI has to consider the people and technology dimensions as well!
10. SPI has to be implemented in a way that all involved persons are convinced of the benefits. It should not been done for its own sake!

7 References

1 Royce, W.W. (1970): Managing the development of large software systems. IEEE WESCON, Los Angeles, CA, 1-9; re-printed in Proceedings of 9th ICSE, Monterey, CA, IEEE Computer Society Press, 328-338.

2 DeRemer, F., H.H. Kron (1976): Programming-in-the-Large Versus Programming-in-the-Small. IEEE Transactions on Software Engineering, Vol. 2 (5), 80-86.

3 IEEE Std 610.12 (1990): IEEE Standard Glossary of Software Engineering Terminology. IEEE Standards Association.

4 CMMI (2006a): CMMI for Development, Version 1.2. CMU/SEI-2006-TR-008, ESC-TR-2006-008, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

5 Boehm, B.W. (1988): A Spiral model of Software Development and Enhancement. IEEE Computer, Vol. 21 (5), Mai 1988, 61-72.

6 Jacobson, I., G. Booch, J. Rumbaugh (1999): The Unified Software Development Process. Addison-Wesley, Boston, MA.

7 Abrahamsson, P., O. Salo, J. Ronkainen, J. Warsta (2002): Agile software development methods. Review and analysis. Espoo 2002, VTT Publications 478.

8 Manifesto for Agile Software Development. http://www.agilemanifesto.org/

9 Ludewig, J., H. Lichter: Software Engineering – Grundlagen, Menschen, Prozesse, Techniken. 2. Aufl. (in German), dpunkt.verlag Heidelberg, 2010. ISBN 9783898646628.

10 Bäumer, D., W. Bischofberger, H. Lichter, H. Züllighoven (1996): User Interface Prototyping – Concepts, Tools, and Experience. Proceedings of 18th ICSE, Berlin, IEEE Computer Society Press, 532-541.

11 Pricope, S., H. Lichter (2011): A Model Based Integration Approach for Reference Models. Second Proceedings of 12th International Conference on Product Focused Software Development and Process Improvement, Danilo Caivano et al., Torre Canne, Italy, June 20-22, ACM, New York, NY, USA, ISBN: 978-1-4503-0783-3, pp 6-9.

12 McFeeley, R. (1996): IDEAL: A User's Guide for Software Process Improvement. CMU/SEI-96-HB-001, ADA 305472, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA.

13 Fundamentals of the V-Model: XThttp://ftp.tu-clausthal.de/pub/institute/informatik/v-modell-xt/Releases/1.3/Dokumentation/V-Modell%20XT%20HTML%20English/

NECTEC-CMMI ML3 Implementation Project

Chayakorn Piyabunditkul, Apinporn Methawachananont, Chumphol Krootkaew

National Electronic and Computer Technology Center (NECTEC), NSTD
chayakorn@nectec.or.th

Abstract. Currently each organization has developed separately. Some Units which has budget can procure high technologies or tools for improving their operation. An important issue which has been mentioned from failure in worldwide competition is work quality. This article proposes a strategy to improve Thai industry's product by NECTEC. It focuses on developing standard processes which affect directly to solve above issues. However NECTEC needs to have experience for consulting them and NECTEC' s executives request to improve the processes especially in R&D laboratories. Thus NECTEC-CMMI project has been established to fulfill this goal. CMMI for development is a popular framework in Thailand and other countries. The standard process has to be designed by regarding existing related processes, problems, policies and best practices. Furthermore, realization to possibility in operating the process in the real situation is key point because there are a lot of stakeholders. The best channel is getting acceptance from stakeholders before implementing the processes. How to know if standard process leads to better quality, measures should be identified for measurement and analysis. Appropriate measures of related processes have to solve current problems and/or answer business goals. Various issues from research environment may differ to others.

Keywords: CMMI, SCAMPI, Implementation, R&D

1 NECTEC-CMMI

For develop quality of software product, organization need to have a quality process. The clarify of organization standard for Software Process Improvement is need to define and tailoring for continued improvement. Thailand's National Electronics and Computer Technology Center (NECTEC) is a statutory government organization under the National Science and Technology Development Agency (NSTDA), Ministry of Science and Technology. Its main responsibilities are to undertake, support, and promote the development of electronic, computing, telecommunication, and information technologies through research and development activities. NECTEC also disseminates and transfers such technologies for contribution to the economic growth and social development in the country, following the National Economic and Social Development Plan.

NECTEC has a mission to promote Software Process Improvement strategy for SMEs software industry in Thailand, however, at the initial phase, NECTEC has to prove the SPI concept by implemented CMMI (Capability Maturity Model Integration) Maturity Level 3 (version 1.2) from Software Engineering Institute (SEI)as a prototype for Government and private organizations. The result of pilot project will be a best practice to establish tailoring's SPI model and SPI tool which fit for Thai SMEs organization environment.

2 CMMI-SCAMPI

The Standard CMMI Appraisal Method for Process Improvement (SCAMPI) is designed to provide benchmark quality ratings relative to Capability Maturity Model Integration (CMMI) models and the People CMM. The SCAMPI Method Definition Document (MDD) describes the requirements, activities, and practices associated with the processes that compose the SCAMPI method. The MDD also contains precise descriptions of the method's context, concepts, and architecture.

SCAMPI A satisfies all of the Appraisal Requirements for CMMI (ARC) requirements for a Class A appraisal method. Although designed for conducting appraisals against CMMI-based reference models, the SCAMPI A method can also be applied for conducting appraisals against the People CMM and other reference models.

3 NECTEC-CMMI Tool

NECTEC established tool as semi-automated assistant equipment to provide configuration management system for NECTEC staff to implement CMMI and finally, prepared self-assessment and maintain for formal SCAMPI A.

4 Conclusion

NECTEC aims to improved their process quality for R&D organization including VSEs/SMEs. CMMI is one of the famous software process improvement model which is NECTEC apply to establish a standard procedure for development software as the best practices based on SCAMPI A regulation. As from our result, NECTEC-CMMI project is passed SCAMPI A ML3 with some opportunities for improvement in some areas by implement NECTEC-CMMI Tool.

5 References

1 Apinporn Methawachananont, SWE-SPI team (2011), Research & Development report (P0040210): Phase 1- NECTEC's Process Improvement by CMMI-DEV ML 3, Pathumtthani, Thailand.

2 CMMI Product Team (August 2006), CMMI® for Development, Version 1.2 (CMMI-DEV, V1.2), CMU/SEI-2006-TR-008.

3 SCAMPI Upgrade Team (March 2011), Standard CMMI® Appraisal Method for Process Improvement (SCAMPISM) A, Version 1.3: Method Definition Document, CMU/SEI-2011-HB-001.

Evaluation of Code Quality Best Practices into Dashboard

Fariha Motherudin, Wong Wai Tong

MIMOS Berhad, Technology Park Malaysia, 57000 Bukit Jalil, Kuala Lumpur
fariha.motherudin@mimos.my, waitong.wong@mimos.my

Abstract. There are various best practices in developing high quality codes that are used by software engineers in order to produce high quality product. The practices are then translated and innovated into automated tools which are quite friendly to be used by other software engineers even in their local development machines. In this paper we studied some of the practices and tools in generating the code quality metrics to measure the performance consistently throughout various projects in the organization. As a high maturity organization, statistical approach is used to quantitatively evaluate its process performance to facilitate decision making for release management. Understanding the patterns of coding process would help not only quality assurance group but to the development team as well to perceive their skills and knowledge and hence, taking the necessary preventive and corrective measures in its software development process.

Keywords: code quality, code quality best practices, code quality tools, code quality metrics, high maturity

1 Introduction

High quality product has been a vital factor in delivering a software project besides meeting the project's dateline, project budget and customer's requirements. There are numerous quality assurance activities that reside on various industrial assurance models and frameworks. Software quality activities including reviews, inspections, testing, quality gates are part of quality assurance checking in making the product and process complied with the model's requirements.

Due to lacking clearly established processes and highly depending on individual skills and experience, code that developed by software engineers is prone to raising faults. Code development itself requires creativity and individuals developers maintain a significant influence on the final results. It is fortunate that the faults discovered upfront during the internal development phase either during unit test execution or Code Inspection stage which gives high impact of higher cost of quality than finding faults at later phases.

In this paper, we studied coding faults and how we came-up with selection of code quality metrics thru automating the process with tools embedded. Our motivation is to establish a standard platform in standardizing the verification in the code process and usage of quantitative measurement in determining code quality level. Section 2 describes the root causes identified in the coding stage which collected from Java projects. Section 3 introduces our code level quality improvement plans and tool which includes compiling the software quality metrics used in the organization which seldom being recorded in one single research paper. The paper concludes with a discussion of the contributions and implications of our study.

2 Code development stages

Software development which already deployed in our organization follows a typical Software Development Life Cycle (SDLC) which has a series of phases; Requirement, Design, Code (Implementation), Testing and Maintenance. Inside each phases there are another set of low level processes. Low level processes of code phase illustrate in Fig. 1.

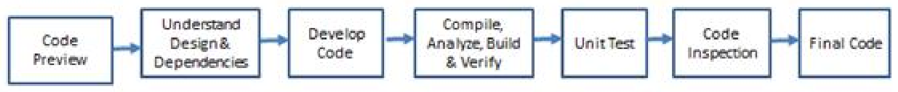

Code Preview
Understand Design & Dependencies
Develop Code
Compile, Analyze, Build & Verify
Unit Test
Code Inspection
Final Code

Fig. 1. Code stages

During code preview, team members deliberate and decide how the next phase's activities will be carried out. The development tools readiness are discussed and require to be ready before the developers could start the development activities e.g IDE, Build, Unit test, configuration management environment, scripts. Prior project execution, some of tools are deployed at the organization level which hand-coupled tightly with organization processes. While some of the tools were pre-installed at development's side as the tools depending on various factors e.g coding language, technology to be used and so forth.

Since the organization members were quite new to the process at the time of data collected, they followed classic way of development process as in Fig. 1 where they develop the code and then run the unit test cases. After Unit test report was finalized, the team inspected their codes in an inspection meeting which follow formal team inspections which was introduced by Fagan 1976[1] conducted on checklist based, proper structured process, defined roles where the meeting focus is to identify and find defects. Data was gathered from several inspection meetings. The studies highlighted highest faults found by the teams which then transferred into code level quality improvement plans.

2.1 Root causes

We selected about ten running and completed Java projects in order to understand and analyze the highest code inspection faults. There is a set of fault categories in which reviewer needs to select the appropriate fault category and recorded in the code inspection forms. Root causes analyses were performed and data was compiled into a Pareto chart to distinguish the causes quantitatively.

The highest faults incurred were related to coding standard violation. This was followed by Control flow and incorrect use of functions. Since some of the engineers are quite new to the process, they were not familiar with the organization standard way of coding. Examples of coding violations are various naming formats of package and classes name, hard-coded codes, usage of public and private methods, missing comments in the code files and others.

At the organization level, there is long list of coding standard guidelines which captured all the coding standards and documented based on different coding languages. The idea of coding standard is not new but in term of utilizing it is another matter to consider. The purposes of putting such emphasize in the coding standard are to have codes consistency across projects, reduce code wastes, and deliver high maintainability codes.

Once of the ways to improve control flows and incorrect use of functions faults is to perform effective unit tests. In the selected projects, unit test cases were developed based on Requirement coverage [2] which then leads to manual way of developing the unit test report. Some limitation that we found while performing manual unit test was that they are prone to be bias in selecting easier test cases which then leads to lower coverage of testing.

3 Improvement Plan

For the three main causes found during the studies, improvement plan requires to be explored, studied, and strategized accordingly. Manual checking of coding standards shall be replaced with more effective methods in order for the teams to have the standard compliance across the organization. Automating unit test process might help in tracing faults and recorded in a very systematic approach and would be useful especially in multiple releases development.

There were a number of independent tools and software metrics being used to access the level of code quality for each software release. Example of metrics that used by software industries are Coverage, Cyclomatic Complexity, code quality and rework effort, etc. The metrics are generated from tools where most of the tools used are open source tools which easily be found on the net and is a matter of embedded it with the current organization process.

One of the most used tools is Static analysis which can be used as soon as the code gets compiled to report the code compliance. The code does not have to be complete, or integrated with the rest of the program in order for the technique to be able to begin to find bugs. It can run on a single file at a time, or a complete codebase, or anything in between. The results are better when the entire program is analysed, but an analysis of small parts can also be useful. Thus developers can get very quick feedback on their code compliance quality.

Besides code compliance index which indirectly tell the skills level of developers which then be collected for competency development purpose, another index to be considered is unit test success rate. Since there are different types of projects in the organization, there are multiple ways of doing unit test either using conventional manual way, Requirements based testing or using unit test tool e.g JUNIT. JUNIT is a unit test tool for Java codes with our intention to reduce effort and time for compiling and executing unit test exercise.

We have put our assumption into a hypothesis statement whether or not automated tools help in reducing testing effort. Effort data was collected from a project in which the developers have experienced in JUNIT. More than 10 data points were taken from each test cases method. The data was taken with assumption for each comparison, the IDE and build environment has already been setup and same developer executing the same unit test to reduce the data variation. The data was then being transferred into Minitab tool.

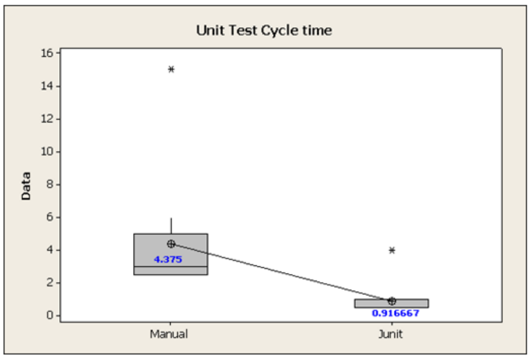


Fig. 2. Box plot of unit test cycle time

After accessing the data and performed some hypothesis, we summarized that less time were taken using JUNIT compared to the manual ways as recorded in Fig.2.

Rationally, most of the automated tools will give positive results in reducing cost, effort and time in various situations. While dealing with CMMI Level 5 maturity practices, it is very important to have a good understanding of the improvement plans by managing it statistically and handle the data in a very efficient for future reference and recording of continuous improvement. However, in some cases the manual way is more effective in finding faults. Therefore, a combination of both manual and automated is still acceptable and practical in software development projects.

The unit test result could be summarized into a report named Coverage report. Coverage report will assist the team to identify which area has the lowest coverage of unit test. Coverage is clearly a good estimator for the fault detection of exceptional test cases but a poor one for test cases in normal operations [3]. The coverage report is substantiated with complexity metrics which believe the justification of the low coverage might be because of high complexity of the classes or methods that been developed [4]. This paper is not focusing on Coverage which we found there are a lot papers and discussions on Coverage metrics can be found separately in other reports.

One of the most useful tools to give a comprehensive report on the status of code quality is Sonar. Sonar is an open source quality management platform, dedicated to continuously analyze and measure source code quality from the portfolio to the method [5].

3.1 Establishment of code quality best practices

We foresee that Static Analysis effectively perform as the first stage where compliance errors and potential faults could be discovered earlier in the coding stage. Not following organization's naming convention, usage of public and private methods, hard-coded graphics numbers should be avoided and clean-up before the engineers shift to the next process.

After the compliance codes have been established, unit test process is performed and briefly show whether the unit or component worked as expected. Unit test helps in finding errors in single component early instead of finding multiple errors in multiple components which is exponentially more difficult. The full report of Unit test rate is transferred into Coverage report where this report is beneficial for the engineers to give importance on the code area which has low percentage in order to increase the code quality.

After achieving the expected compliance, unit test and coverage rate, codes are ready to be inspected. Codes are reviewed together with Sonar result. For some cases where the compliance is not achievable, the author will justify with the reviewers and to seek for any recommendation to improve on their codes. Eventually it also improves their coding skills.

Summary of the outcome proposed by process improvement group to the software developers in order to maintain the code quality and then the process being tracked in metrics format in Sonar dashboard shows in Fig. 3.

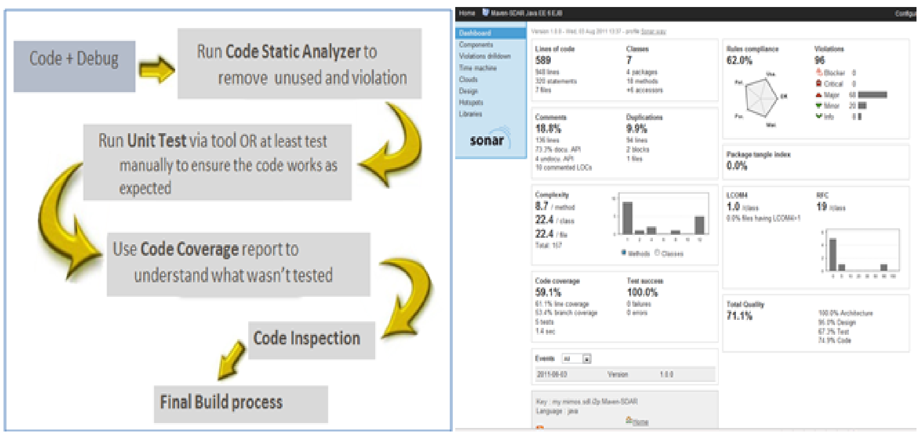

Code + Debug
Run Code Static Analyzer to remove unused and violation
Run Unit Test via tool OR at least test manually to ensure the code works as expected
Use Code Coverage report to understand what wasn't tested
Code Inspection
Final Build process
sonar

Fig. 3. Code quality process and Sonar dashboard

The processes are to be followed as per proposed order. However, it is also depending on engineer's skills, competency and preferences. At the end, the processes are traceable thru metrics which presented in the Sonar dashboard.

4 Conclusion

Although there are several other metrics being introduced by other industrial practitioners, the three metrics (Compliance, Unit test, coverage) are used as our starting point in compiling the project data to gauge the level of software code quality.

By introducing static analysis index in the organization help the team in improving their coding skills, finding potential defects upfront and reducing code wastes. After the codes adhere to the organization standard, unit test is performed. Unit test and coverage metrics help in identifying areas that has not been tested and give hints to test group in finding more faults before handing the product officially to customers.

All these reports are transparent and visible thru a centralized dashboard named Sonar. It would be best if this dashboard integrated with Project Management dashboard for real-time monitoring and reporting. The collection of data for measurement and analysis purpose is to identify process and product measures in order to make good management and engineering decisions.

The new finding about collection of code quality best practices can be used to guide the selection and evaluation of the practices under various project profiles for future reference purpose. It will help the practitioners in their planning process in order to meet the project deadline without ignoring the code quality activities and at the end continuously produce high quality products.

5 References

1 Sami Kollanus, Jussi Koskinen.:Survey of Software Inspection Research in The Open Software Engineering Journal (2009)

2 Technobuff, Requirement Coverage-Another dimension to Unit Testing http://www.technobuff.net/webapp/product/showTutorial.do?name=jrequire&tssar

3 Xia Cai, Michael R. Lyu of The Chinese University of Hong Kong: The Effect of Code Coverage on Fault Detection under Different testing profiles. In : 1st international workshop on Advances in model-based testing (2005)

4 Raghu Lingampally, Atul Gupta and Pankaj Jalote from Indian Institute of Technology Kanpur India: A multipurpose Code Coverage Tool for Java in Proceedings of the 40th Hawaii International Conference on System Sciences (2007)

5 Sonar Open source for managing Code quality http://www.sonarsource.org

Improvement and Enhancement in Configuration Management Process Area within the Organization

Nor Izyani Daud, Rudhuwan Abu Bakar, Galoh Rashidah Haron

MIMOS Berhad, Technology Park Malaysia, 57000 Bukit Jalil, Kuala Lumpur.
izyani.daud@mimos.my, duan@mimos.my, rashidah@mimos.my

Abstract. This paper attempts to share the approach on improvement and enchancement in executing processes and activities under Configuration Management (CM) process area in the organization. It begins by describing the overview about CM and the implementation in the organization. Then, it focuses on the problem occurred within the current CM process area. It is based on real case scenarios and mostly rooted from combination of people and tools incompetence factors. The specific practices that are required to strengthen the support of the CM and outline the continuous improvement initiatives taken towards the betterment of CM process will be elaborated. The benefits by having the improvement and enhancement will be described in detailed. The conclusion was made based from the author opinion about the CMMI and specifically CM process area in the organization.

Keywords: configuration management, process improvement, best practices

1 Introduction

CMMI is one of the process improvement approaches that help to improve the organization performance. It is an internationally recognized process improvement model initiated by Software Engineering Institute (SEI) of Carnegie Melon University. Our organization implements Capability Maturity Model Integration-Dev v1.2 (Continuous Representation) practices to the organization in year 2007 and has become the first Malaysian government agency that obtains this prestige achievement on October 2009[1].

CM has been adapted to the software development life cycle process, since our organization started to implement CMMI Level 3. It involved in every phase of the current software development life cycle, Waterfall Model. It is one of the key activity that support both technical and management part in the software development area.

The organization had implemented specific goals (SG) and specific practices (SP) in order to implement the CM process. The SG and SP that has been practiced are as in page 203, from "CMMI: A framework for Building World-Class. Software and Systems Enterprise [2].

The organization had implemented 5 practices to align with the aforementioned SG and SP for the CM process area. There are Software Configuration Management Plan, Baseline and Archiving, Configuration Control, Configuration Status Accounting and Release Management. Several tools have been implemented in our organization. It is to support the CM process in order to manage all the configuration management activity during the project life cycle. The tools are Clear Case (CC), Clear Quest (CQ) and Subversion (SVN).

2 Problem within the current CM process area

There are some issues on the implementation of the current CM process area in the organization, such as:

1. Lack of knowledge on CM practice, tools and tool expert
   A resource is the most important factor that contributes to the successful implementation of the practices. When the CM process was first implemented, most of the project team lacks the knowledge about the CMMI, CM process and practices. The organization had provided internal and external trainings on the CM process and tools, but it was still not enough the support the SCM personnel needs. In addition, there was only 1 Tools Engineer provided by the organization to support the tool usage. With the number of tool expert, user and SCM tools, sometimes it delayed the time taken to solve the issue. The figure below was taken from Process Tools and Six Sigma (PTSS) helpdesk, showing the number of the problem ticket raised regarding CM tools by the user from year 2008 until 2011. The figures showed that the number for the problem was increased drastically. For example in year 2009, there are 665 problem ticket raised and 1 manpower was not enough to support it.

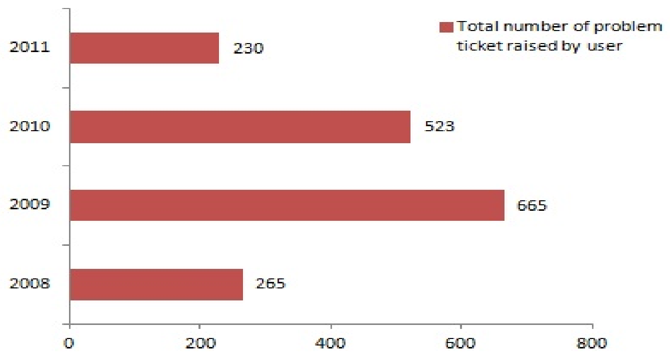


Fig. 1. Total number of problem ticket raised by user for CM tools

2. All privileges access was given to all CC users
   During the installation and configuration of the CC, all the users were given a full privilege access to the Versioned Object Based (VOB) [3]. CC was not properly been setup and configured by the system administrator. As a result, one SCM personnel accidently deleted the project data. The Central Engineering (CE) team

had to restore the backup VOB in order for the project team to get back their previous data.

3. System integration issues
   Developer tends to work in a silo, and will only start to communicate among themselves when system integration phase starts. An issue such as missing dependency was a common occurrence due to different library version used. Sometimes a specific library had missing when the new module was being integrated as it was accidently got erased during the integration. This leads to time-consuming effort to troubleshoot the root cause and end up extending the project dateline. Furthermore, there was no checkpoint or restore point to go back after each round of integration failure. This result in the team needed to scramble to re-build from scratch and frequently a new issue would appear. Usually, it took a few days before the team managed to release a workable integrated package.

4. Cost for tools and maintenance
   CC and CQ are commercial tools and comes with maintenance and support fees. With limited financial resources, the organization was not able to support the user needs.

3 Process improvement and enhancement in CM process area

Our organization improved and enhanced the CM process to align with the needs of the current software development life cycle activity. It was also to solve the problem within the current CM process area. The organization had decided to establish a Configuration Management System as stated in SP 1.2 by:

1. Establish SCM Interest Group
   SCM Interest Group was basically a team of SCM personnel of a project in the organization. The group will have knowledge sharing among all the team members on CM best practices. SCM Interest Group lead had prepared the milestone for the task that the SCM Interest Group plans to do for that particular year. The milestone will be shared among the SCM team members.

2. Establish CM portal
   SCM portal was created as a knowledge based share repository for the SCM team members. All the team members were given the user id and password to access the portal. It is to ensure that only restricted user able to access in order to protect the confidential of the information inside the portal.
   SCM portal contains the information about CM practices in slides format, online video training and minutes of meeting for all the discussion within the group. There was also a section for discussion board where all the SCM team members able to post any questions regarding the problems while performing the CM task. A list of the contact information about the SCM team members was also listed in the portal.

3. Establish SCM mailing list
   SCM mailing list was created for all the team members in the SCM Interest Group. The purpose of the mailing list is to create a centralized communication channel between the SCM team members. With the implementation, all the SCM team members will not miss out any latest information or updates regarding the CM process. As for now, there were 18 person includes in the SCM mailing list

4. Establish continuous integration
   To accomplish continuous integration, a tool should be able to bind and integrate all modules at the start of the project. Furthermore, it should be able to store all builds as a restore point. *Hudson* [4] has been deployed to improve the current build management and system integration process. It came with various plug-ins such *Maven 2* for software build, unit testing and dependency management, *Subversion* for code management and *Sonar* for code coverage report, *email notification* and *build scheduling*.

5. Implementation trigger for Clear Case
   As a lesson-learned from the previous incident, the trigger has been implemented to CC. The purpose of having it was as the preventive action to control the user access right or power to the VOB. The trigger that has been implemented are rmelement trigger, rmname trigger, remove empty branch trigger and change owner trigger [5].

6. Use SVN as a main SCM tool
   As an alternative, a case study has been conducted and a report “Cost Benefit Analysis Subversion” has been prepared. The report compared the cost, benefit and features of SVN against various SCM tools. Based from the report, the organization had decided to use SVN. A proper plans need to be in place. Briefing and training sessions were conducted for the SCM Interest Group and project team. Since then SVN has been implemented throughout all projects.

4 Benefit of the improvement and enhancement in CM process area

There are some benefits of the implemented improvement in CM process area in the organization. There are:

1. SCM Interest Group plays a role as a middle-man between the project team and PTSS.
   SCM Interest Group act as a reference point for all projects. PTSS relayed new changes on process or any latest announcement to the SCM Interest Group. The group had updated the changes to all projects through meetings, presentations, SCM portal and emails. The information also will be shared among their project team members. In the other way, these created a proper channel of communication

throughout the organization to ensure all CM practices are in-line with the company policy.

2. Reduce number of problem ticket regarding tool.

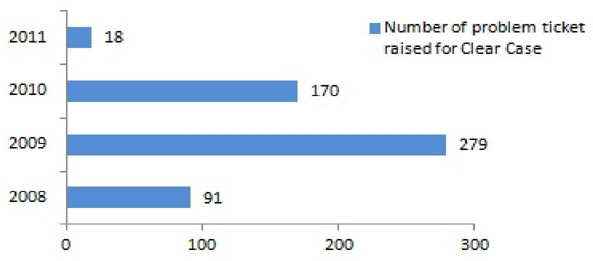
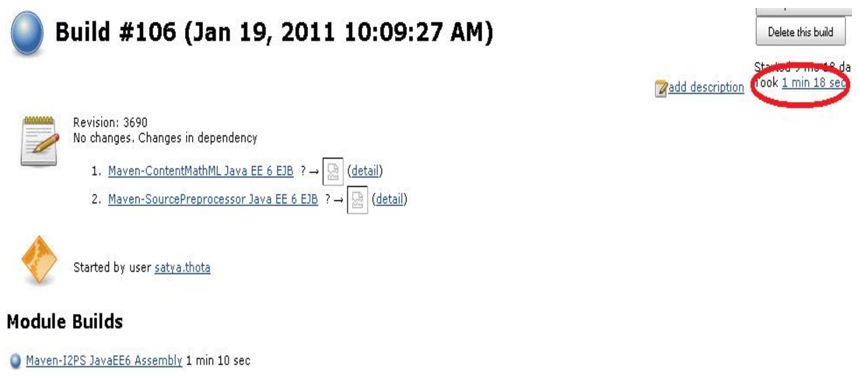


Fig. 2. Number of problem ticket raised for Clear Case

Figure 2 shows that the number of problem ticket raised by the users had decreased since 2009. The data was taken from the PTSS helpdesk. With SCM Interest Group, SCM portal and mailing list setup in 2009, users have more communication channel which indirectly contribute to lesser user complaints. This reduced the number of the problem ticket raise and also the Tool Engineer task.

3. Improve build time for project.
   Continuous Integration eliminates the so-called "integration hell"[6]. Early bug detection reduces rework and the system enabled automated unit testing and rapid integration. All this was done in transparent environment and certainly improve the quality of the product to be delivered to the market. With the setup, integration and rework efforts tremendously reduced; and thus reduced development cost and time.



Build #106 (Jan 19, 2011 10:09:27 AM)

Delete this build

add description

Took 1 min 18 sec

Revision: 3690
No changes. Changes in dependency

1. Maven-ContentMathML Java EE 6 EJB ? → (detail)
2. Maven-SourcePreprocessor Java EE 6 EJB ? → (detail)

Started by user satya.thota

Module Builds

Maven-I2PS JavaEE6 Assembly 1 min 10 sec

Fig. 3. Hudson build screen shot

Figure 3 captures time taken to complete the integration build. It only took 78 seconds to have a workable integrated package as compared previously which took a few hours and sometimes days to complete. Since it was integrated with SVN, Hudson will pullout only updated codes and builds them. This way, it resulted in a faster compilation and for any failed build; it will notify respective developer immediately, hence early bug detection.

5 Conclusion

CMMI is a good practice that should be implemented in any software development organization. It is to ensure the qualities of the delivered product met customer's expectation. A proper CM practices need to be in place in order to support the other activity in the software development life cycle.

Basically, for organization to be accredited with CMMI Level 3, it must have at least a basic configuration management system in place. It should have a standardized repository to store all their data. A common tool to perform the CM activity such as check in, check out and labeling; needs to be in place to support the process. A dedicated SCM personnel need to be appointed in every project to perform the SCM task.

To implement it, a proper study and planning need to be in place. For example, on the proposed SCM tool such as its cost, features, ease of use and its future maintenance. This is important to avoid spending unnecessary budget on commercial tool if there is free open source software available.

Problems that provided in this paper is not exhaustive; it does provide some indication of the significant of having proper process and supported system in place and ultimately going to lead to a successful and high quality end product. It might be useful to other when they want to implement the CMMI practices to their organization.

The improvement and enhancement that has been implemented here is the best for the organization. As CM process area is a living process and it needs consistent improvement over time.

6 References

1. MIMOS Berhad, http://www.mimos.my/2010/08/capability-maturity-model-integration-cmmi/
2. Raghav S Nandyal.:CMMI:A Framework for Building World-Class Software and Systems Enterprises. New Delhi (2004)
3. Versioned Object Based, http://searchsqlserver.techtarget.com/definition/Versioned-Object-Base
4. John Ferguson Smart : Jenkins The Definitive Guide, Creative Common Edition, Wakaleo Consulting(2011)
5. IBM Rational ClearCase: The ten best trigger, http://www.ibm.com/developerworks/rational/library/4311.html
6. Integration Hell, http://c2.com/cgi/wiki?IntegrationHell

Software Process Improvement Practices – A Pakistani Perspective

Nasir Mehmood Minhas[1], Javed Iqbal[2]

[1]UIIT-PMAS Arid Agriculture University, Rawalpindi, Pakistan
[2]COMSATS Institute of Information Technology, Islamabad, Pakistan
nasirminhas@uaar.edu.pk, javediqbal@comsats.edu.pk

Abstract. It is believed that Software Process Improvement methodologies can lead towards batter quality software. Software development companies are becoming increasingly aware of the importance of an efficient and well-managed software process. In this research we studied the SPI implementation trends among the Pakistani software industry. The research was aimed to produce some analytical information on the SPI implementation within Pakistani software Industry. Issues addressed by the research was to study the overall industry trends for the implementation of SPI practices, the segment wise variations in this implementation, problems in the implementation of SPI programs and the characteristics of people involved in SPI activities.

Key words: Software, Software Process, Software Process Improvement (SPI), Pakistani Software Industry

1 Introduction

Software plays a vital role in today's life and day-to-day business operations. This research purposes to determine the current SPI trends within the Pakistani software industry, the major problems faced by the companies in initiating and implementing an effective SPI program, the variation across different segments of industry, and the demographic attributes of the software practitioners working in different organizations. All these aspects help to analyze the current status of SPI practices in the Pakistani industry on the whole.

Study provides an evidence of extensive research ([2], [11]) that has been carried out in this regard around the globe. A number of studies have investigated the current status of SPI practices in various companies ([6], [7], [9], [20]). Some of these studies focused on various factors that affect software process improvement program ([1], [3], [8], [10], [14], [19]). There are also many reports from practitioners, reporting on successes and failures of software process improvement efforts at organizations of various sizes ([15], [16]).

2 Literature Review

While reviewing the literature we learnt that there are number of studies that have been carried out to investigate the issues related to SPI. One group of studies we found was about to define SPI, and they have given the definitions of SPI in their own ways [5], [17], [21]. We found couple of studies where motivators for SPI has been investigated [14], [15] in these studies authors indentified some of the motivators as to-down commitment, visible success and resources, sense of empowerment and process ownership. We encountered with the number of studies where authors were investigating the problems and de-motivators regarding the SPI program [8], [16], [12], the de-motivational factors identified by the authors were mainly lack of management support, organizational politics, lack of resources, commercial pressures, lack of evidence of direct benefits, cultural problems, and misunderstood goals. In some of studies organizational issues and factors which can affect the SPI program have been identified [13] following organizational factors were identified business orientation, involved leadership, employee participation, and concern for measurement, exploitation, and exploration. Some of the authors have drawn a correlation between SPI success and organizational size [18], [11], [4] according to authors, success of SPI program is directly related with the size of organization. The findings from the studies showed that there are fundamental differences between small and large software organizations. With respect to organizational performance in general, large software organizations reported a higher level of overall SPI success than small organizations.

3 Data Analysis

We formulated a questionnaire to investigate the SPI trends according to CMMI requirements, we divided the sections of questionnaire according to Key Process Areas (i.e. one section for each process area) questions in each section were targeting to the key activities of that particular process area. Three options (i.e. routinely practiced, occasionally practiced, not applicable) were given against each question. Routinely practiced means that activity is being practiced on regular basis, occasionally practiced means that activity is practiced but not on regular basis, while not applicable means that activity is taken as necessary or the activity is not being practiced at all. We distributed our questionnaire among 80 practitioners of 25 different companies, with a diversified background and experience. We selected the feedback of 25 respondents i.e. one from each company on the basis of their profile (i.e. Experience and involvement in process related activities) and because of the fact that the respondents who were belonging to a same company gave similar response. The interesting thing that we collected during this survey was that most of the practitioners have not received any training regarding Process Improvement or Quality Management, and only a few are trained for the purpose. In fact 71% of our respondents were not having any training and only 29% of our respondents claimed to be trained in Software Quality Management, ISO, CSQP, ISO 9001 - 2000, and Inhouse training regarding quality assurance. In our sample of selected companies not even a single company has achieved any level of CMM / CMMI. According to official sources of Pakistan Software Export Board (PSEB) there are 25 IT companies who are achieving the various levels of CMMI, we didn't selected these companies as they are already engaged in SPI practices.

We also collected the necessary data about the participating companies with the help of this data we made a division of organizations as small, medium, and large. 68% companies are small, 24% are medium and 8% are large. We made this division on the basis of no of professionals working in a company and the volume of work they are undertaking in a cycle of week. Major share in our sample is of small companies so the results coming from this segment would be of more importance.

In the subsequent sections we'll present the Overall SPI trends in Pakistan, followed by the SPI trends in different segments of industry.

3.1 Industry Trends

Fig. 1 shows the overall trends regarding the adoption of SPI practices in Pakistani software industry. It is evident from the data that the most focused Key process areas in our industry are the requirements management and software quality assurance as more than 50% companies claimed to practice these areas. According to our finding the most negligible key process area is Subcontract Management, as only 27% companies are practicing it. Imparting different training programs is an important activity for the growth of the individuals and companies as well. Just 30% companies of our sample believe in training program.

This is an era of rapid technological changes, every day new technologies are coming in front, and especially in software industry this change is more brisk. Technology Change Management is another process area that is being practiced less than one third of the industry; the results show that only 31% companies claimed that they are managing the technological changes properly. On average almost 39% companies claimed the implementation of software process improvement practices, 43% companies are those who have shown a casual behavior regarding SPI practices, while 18% companies are those where SPI practices are totally ignored.

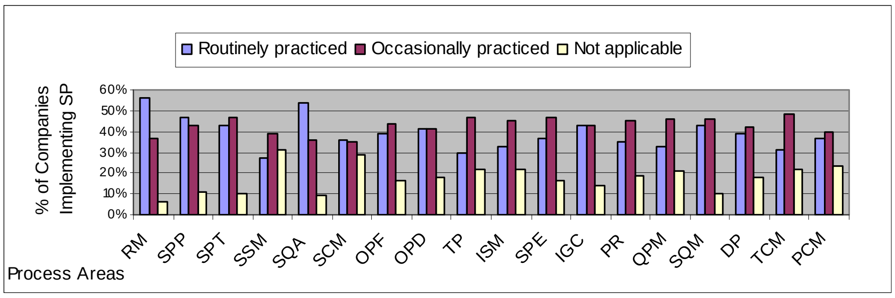


Fig. 1. SPI Trends in Pakistan

3.2 SPI Trends in Small Companies

Our sample shows that majority Pakistani companies come under the category of small size, so SPI trends depicted by this segment are of more importance. Data about this segment of industry shows, that 33% of the participating companies are those who have claimed the implementation of SPI practices, but the detailed results show a varying trend. Among the remaining 67% participating companies 47% are those who have shown a casual behavior while 20% companies are totally ignoring this fact. The conclusion can be drawn from the given results that small companies are not perusing the software process improvement program. As we have mentioned earlier that this segment of industry represent the majority, so final conclusion could be drawn the majority of Pakistani software industry is not adhering the SPI program. Key process area wise detail can be seen in Fig. 2.

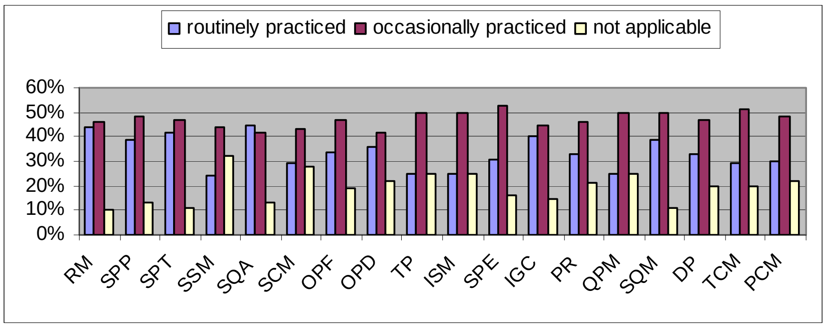


Fig. 2. SPI Trends in Small Companies

3.3 SPI Trends in Medium Sized Companies

From the sample of participating companies it has been learnt that medium sized companies are quite less in numbers as compared to small companies, so that result shown by this segment of industry cannot be considered as influential on the overall industry trends.

Overall results are quite encouraging as compared to small companies, 50% companies claimed the implementation of SPI practices, 38% showed an ad-hoc behavior and, only 12% companies are those who were not imparting SPI practices. Results for the implementation of key process areas in the medium sized companies are presented in Fig. 3.

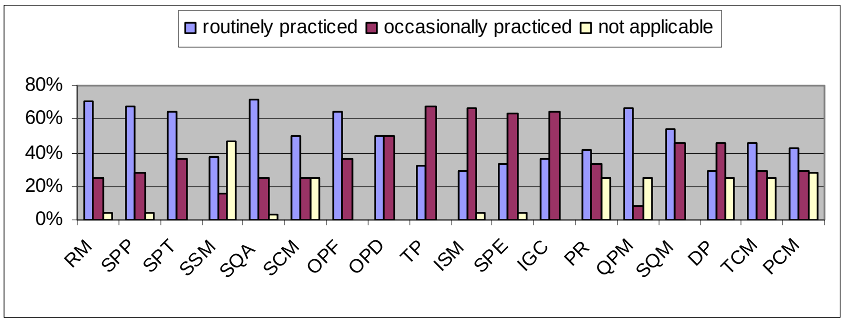


Fig. 3. SPI Trends in Medium Sized Companies

3.4 SPI Trends in Large Companies

Large companies exhibit a batter implementation trend of SPI practices, as we can see in Fig. 4 routinely practiced bar is more prominent. Participation of the large companies in our sample is at the lowest side as only 8% companies come under this category. Results in this section are not the representative of the industry.

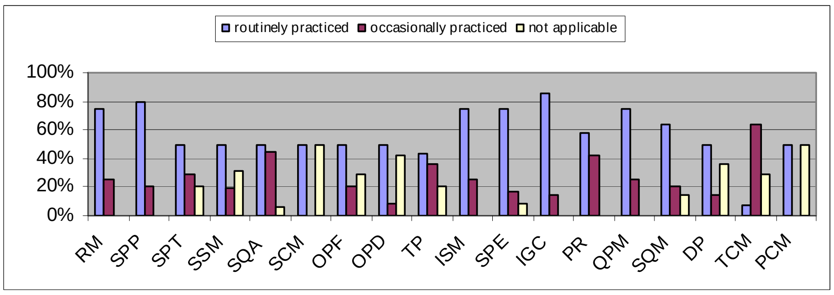


Fig. 4. SPI Trends in Large Companies

In the category x of the Figure 1, 2, 3, & 4 we used the abbreviations of key process areas that are as under

RM : Requirement Management, SPP Software Project Planning, SPT: Software Project Tracking and oversight, SSM: Software Subcontract Management, SQA: Software Quality Assurance, SCM: Software Configuration Management, OPF: Organizational Process Focus, OPD: Organizational Process Definition, TP: Training Program, ISM: Integrated Software Management, SPE: Software Product Engineering, IGC: Inter-group Coordination, PR: Peer Review, QPM: Quantitative Process Management, SQM: Software Quality Management, DP: Defect Prevention, TCM: Technology Change Management, PCM: Process Change Management.

4 Problems in SPI Implementation

To identify the problems regarding the implementation of SPI practices was another key focus of our study, after the analysis of the data gathered through questionnaire we conducted the interviews of our some selected respondents to investigate that why the response was so poor regarding the SPI program. During these sessions with the practitioners we discovered some problems faced by the industry in the implementation of software process improvement. These problems hinder the effective exercise of SPI related practices.

Our industry has depicted a poor implementation of some major key process areas including the Subcontract management, Training program, Technology change management and so on. The major problems faced by our organizations in SPI implementation include the following:

1. Lack of training programs
2. Our burdened practitioners
3. Poor inter group Coordination
4. Budget constraints, specifically in small organizations
5. Lack of management support
6. Lack of focus on incorporation of new technologies.
7. Lack of organizational level drift for SPI implementation.

5 Conclusions and Future Work

In this paper we presented the study of Pakistani IT based industry regarding the software process improvement practices. We selected a sample of 25 such companies which are not having any official certification of CMMI. The total number of IT companies in Pakistan is nearly 1500 and only 25 companies are achieving various levels of CMMI. Our sample was the representative of the majority companies in Pakistan. We tried to make mix of all segments in our sample, just to have a comparative view of SPI trends. We noted that small and medium enterprises are facing more problems in this regard. So it is needed that governmental agencies like PSEB provide more support to small and medium enterprises beside this, a flexible framework for the adoption of SPI in SMEs that best suites our local needs and culture is required. In the extension of this study we are working on such a framework.

6 Acknowledgement

We would like to acknowledge the efforts of Mr. Imran Hashim from National Database Registration Authority (NADRA) Pakistan who provided us with great support during the data collection phase, we also thankful to Mr. Imran ul Haq of Gulf Net Pakistan for his cooperation during the survey phase. Here we would like to express our gratitude to all the respondents of our study for their patience and time they spared for putting their response on the lengthy questionnaire. Finally we are grateful to Mrs. Nayla for helping us in data automation process.

7 References

1 Austen Rainer and Tracy Hall; “Key success factors for implementing software Process improvement: a maturity-based analysis”, Department of Computer Science University of Hertfordshire, Hatfield, Herfordshire, U.K.

2 Austin, R. D., & Paulish, D. J. (1993). A survey of commonly applied methods for software process improvement. Pittsburg, PA: Carnegie-Mellon University. (NTIS No. ADA 278595).

3 Biro, M. & Messnarz, R.: Key Success Factors for Business Based Improvement, Proceedings of the EuroSPI´99, Pori, Finland, 1999

4 Batista J. and de Figueiredo, A.D. (2000), SPI in a Very Small Team: a Case with CMM, Software Process – Improvement and Practice, Vol. 5, No. 4, pp. 243-250.

5 Zahran, Sami Software Process Improvement: Practical Guidelines for Business Success. Reading, Mass.: Addison- Wesley, 1998.

6 Diaz, M. and Sligo, J. (1997). “How Software Process Improvement Helped Motorola.” IEEE Software 14(5): 75-81.

7 Gediminas Mikaliūnas, Martynas Reingardtas, software process improvement in lithuania - Ab Alna case study

8 Goldenson, Dennis.R. and Herbsleb, James. D (August 1995), “After the Appraisal: A Systematic Survey of Process Improvement, its Benefits, and Factors that Influence Success”, CMU/SEI-95-TR-009, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania 15213

9 Haley, T., J. (1996). “Software Process Improvement at Raytheon.” IEEE Software 13(6): 33-41.

10 Hammock, N. (1999). Critical success factors in SPI. 4th Annual European Software Engineering Process Group (SEPG99) Conference.

11 Brodman Judith.G.and Donna L. Johnson (1994), What Small Businesses and Small Organizations Say About the CMM, Proceedings of the Sixteenth International Conference on Software Engineering, IEEE Computer Society Press, pp. 331-340.

12 Sarah Beecham, Tracy Hall, and Austen Rainer. Software process improvement problems in twelve software companies: An empirical analysis. Empirical Software Engineering, 8(1):7–42, 2003

13 Tore Dyba, Enabling Software Process Improvement: An Investigation of the Importance of Organizational Issues, SINTEF Telecom and Informatics, Norway. Empirical Software Engineering, 7, 387-390, 2002.

14 Mahmood Niazi and Mark Staples, Systematic Review of Organizational motivations for adopting CMM-based SPI, NICTA Technical Report PA005957 February, 2006

15 Nathan Baddoo and Tracy Hall (2002), Motivators of Software Process Improvement: an analysis of practitioners’ views, Department of Computer Science, University of Hertfordshire, The Journal of Systems and Software 62 (2002) 85–96.

16 Nathan Baddoo and Tracy Hall (2002), De-motivators for software process improvement: an analysis of practitioners views, Department of Computer Science, University of Hertfordshire, The Journal of Systems and Software 66 (2003) 23–33

17 Paulk, Mark C., Bill Curtis, Mary Beth Chrissis and Charles V. Weber, 1993, "The Capability Maturity Model for Software, Version 1.1", Technical Report SEI-93-TR-24, Software Engineering Institute, Carnegie Mellon University, U.S.A.

18 Laitinen, M., Fayad, M. and Ward, R. (2000), Software Engineering in the Small, IEEE Software, Vol. 17, No. 5, pp. 75-77.

19 Sarah Beecham, Tracy Hall, and Austen Rainer. Software process improvement problems in twelve software companies: An empirical analysis. Empirical Software Engineering, 8(1):7–42, 2003

20 Wellisch J.P. Software Process Improvement in CMS - are we different?, CERN, 1211 Geneva, Switzerland

21 Humphrey, Watts S., 1989. Managing the Software Process, Addison-Wesley, Reading, M.A. U.S.A.