ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 7, July 2013

An Approach for assessing the Quality of Software for small and medium sized firms

N. Veeranjaneyulu

Associate Professor, School of Computing, Vignan University, Vadlamudi, India[1]

Abstract: Software quality and flexible development approaches become increasingly important in software engineering to meet the wide range of frequently changing customer requirements in various application domains. Software development and assessment methods have caught the attention of software engineers and researchers worldwide. The quality of a process can be assessed by some external audits but they usually are heavyweight and costly processes. To measure the quality and follow the improvements in a process a lightweight assessment method is desired. This paper reports results from a study, which aims to organize, analyze and make sense out of the dispersed field of agile software development methods. The comparative analysis is performed using the method's life-cycle coverage, project management support, type of practical guidance, fitness-for-use and empirical evidence as the analytical lenses. This paper mainly aims at Software Process Improvement (SPI) paradigm from the perspective of both Reference Process Models and the experience Factory (EF) infrastructures.

Keywords: Process Assessment, Agile, QIP, GQM, SPI

I. INTRODUCTION

The crucial part in developing software is that it does what the user or customer wants. This is also the most challenging part. Before any code can be written, a specification needs to be documented to lay out the tasks at hand. It is equally important to verify that software (or a piece of it) does what was specified. In his paper, Brooks[1] highlights four important properties in software projects that make them difficult to manage; complexity, conformity, changeability, and invisibility. These unwanted properties are still today the core issues that make software projects fail. Today's software systems are enormously complex. Nobody can fully understand every detail of a modern software system . Because we have to trust closed source modules and interfaces written by other members of the team, the importance of correct specifications is even more emphasized. All this adds up to a very large number of logical states which make unwanted behavior of the software likely. Software changes all the time. It is more common to integrate reused code into new use domains than to rewrite a system from scratch. The system requirements may also change over time: support for new operating systems is needed, the system needs to be scaled up to handle more users, the system needs work with new peripherals etc. The most notable difficulty of software development is probably the fact that software is invisible. It is much easier for humans to understand the structure of a building or even of an electric circuit. A software development process is a set of tasks or activities imposed in a given order on the development of the software product. There are several models that describe such software development processes.

The process models help stakeholders to understand the current state of the project, to speak a common language and help ensure stable, capable, and mature processes.

In this four common software development models were introduced. Start with the traditional development models: the Waterfall Model and the Spiral Model. Then, iterative development methods were introduced: the Rational Unified Process and agile process models (eXtreme Programming and Scrum). These more recent models attempt to solve some of the problems of the traditional models but are no "silver bullets" on their own. Software Process Engineering activities are sets of best practices for development process models. When a process model is combined with a process engineering model, it gives tools to measure and compare the maturity of development processes between companies or projects. A CMMI or ISO model is not a process by itself, but allows the project to implement a model that best suits the needs of the project or company standards. In this first a set of widely used software development process models focusing more on them through the Software Process Improvement (SPI) perspective has been presented. Two approaches to the SPI paradigm are described in this; the approaches are termed Process Reference Models and Experience Factory. This will be the background for the process assessment method of agile processes that will combine the reference model approach and software metrics. The work done in this is an introduction to this kind of assessment of agile processes. Further research is needed to refine and develop the model and the assessment.

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 7, July 2013

II. SOFTWARE PROCESS MODELS

The Traditional software development models introduces in this are: Waterfall Model and Spiral Model. Then, the iterative models described are Rational Unified Process and agile process models (eXtreme Programming and Scrum). Agile methods are a family of development techniques designed to deliver products on time, on budget, with high quality and customer satisfaction.

A. Waterfall model

The waterfall model is the most straightforward approach to tackle *ad hoc* development. The waterfall model was first presented by Winston W. Royce [3] although he never used the term "waterfall model". Later Royce proposed a final version of a software development model, the spiral model. At first glance, this model is attractive. It is definitely an improvement over ad hoc development. The waterfall model splits complex tasks into smaller, easily manageable sub-projects that deliver an outcome that can be inspected. The product of the previous step needs to be inspected and verified and each step must be flawless. Unfortunately, this model is too naive. In practice steps overlap each other: during the design phase, problems of specification are identified, during the implementation phase problems of design are identified and thus the waterfall model is not as streamlined as one would wish.

B. Spiral model

The problems of the waterfall model have been known for a long time, especially in long, expensive and highrisk projects. As late as in 1988 Barry Boehm explained why the so called spiral development model would be a superior to the Waterfall model [8]. This paper addresses the risks involved in the development process and the changes intrinsic to software development. These concerns are not very well covered by the waterfall process model

2.3 Iterative development

Iterative development methods are developed in response to the weaknesses of the classic waterfall model. The spiral model is an early example of an iterative development model. Today the spiral model has been refined further and its basic principles are currently essential parts of the Rational Unified Process (RUP)[9], Extreme Programming[15] and generally the agile software development frameworks. Historically iterative development may also mean incremental development. But to avoid confusion these two terms were merged into practical use
in the mid-1990s. The authors of the Unified Process (UP)[10] and the RUP[9] selected the term "iterative development" to generally mean any combination of incremental and iterative development.

C. Rational Unified Process

The Rational Unified Process (RUP)[9] is an iterative software development process framework and, at the same time a software process product developed by Rational Software, a division of IBM since 2003. RUP is based on the spiral model by Barry Boehm[8] but is highly modified from the original model. RUP further evolves and defines the principles for iterative development and use of prototypes. It embeds object-oriented techniques and uses the UML as the principal notation for the several models that are built during the development.

The software lifecycle is broken down into subcycles, each subcycle working on a new generation of the product. RUP divides each development cycle into four consecutive phases: inception phase, elaboration phase, construction phase and transition phase.

D. Agile Methods

Extreme Programming, Scrum, Dynamic Systems Development Method (DSDM), Adaptive Software Development, Crystal, Feature Driven Development and Pragmatic Programming are some of the process models implementing agile development principles. In this paper we will cover the two most frequently used models - eXtreme Programming (XP) and Scrum.

Extreme Programming

eXtreme Programming (XP) is probably the best known agile software development methodology. It was introduced in 1999 by Kent Beck [15] as an answer to problems faced by the long development cycles in traditional development. XP emphasizes the facts that systems have vague user requirements, and acknowledges that rapid changes are inevitable. These starting points matched well with the current time of software development where short development cycles, introduction of new technology and emphasized focus on speed-tomarket were considered competitive business factors as a result of the rise of the Internet and the "dot-com boom". The ideas behind XP are not that "extreme" or new. Rather, XP takes traditional development practices "to the extreme", for example by writing all unit test cases before any code is implemented. It also takes to the extreme the fact that the end customer wants working

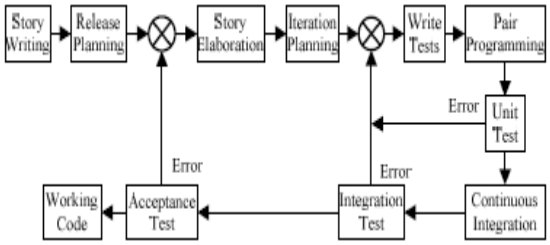

Fig 1: System Representation of XP

software over documentation. In XP, the amount of documentation is limited to what is absolutely necessary; for example, all implementation documentation should be automatically generated from the code and its comments. The code should be the documentation itself, because

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 7, July 2013

eventually when someone has to make changes to the code, it is more important to know what the actual code does over what a possible outdated document says. Coding standards and shared code ownership aid in this matter.

Scrum

Scrum1 is an agile software project management method in contrast to XP which is an agile software development method. It was introduced by Ken Schwaber in 1996 [16]. Scrum will not define in what way the software is developed, what documents are to be produced or how requirements are defined or gathered. Rather, Scrum is a guide on how an agile implementation team should be managed. Schwaber [16] observes that Scrum is probably most successful when it is used for prototyping new technology or for implementing a completely new system with a number of uncertainties. Obviously, Scrum and XP work very well together. As any agile approach, Scrum notices that the development phase is under constant pressure of change and involves several environmental and technical variables (e.g., requirements, time frame, resources and technology) that are likely to change during the process. Scrum goes as far as calling the development environment as being a set of "chaotic circumstances".

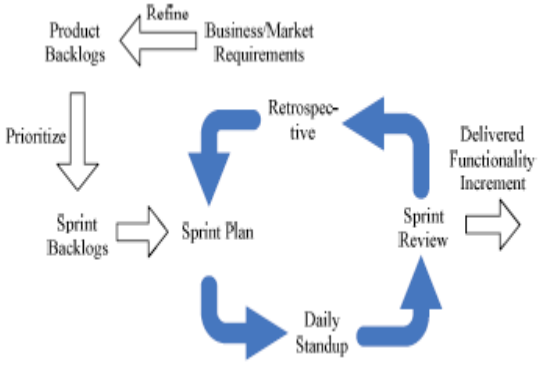

Fig 2: Scrum Model

III. SOFTWARE PROCESS MODELS

A software process model is an abstraction of a software process, which, in turn, is an abstraction of a set of real-life activities. Because of the abstracted, or simplified, nature of software process models, the activities have to be customized and optimized to the actual development environment (business goals, work methods, system requirements, resources etc.).

In this the methodology for improving implemented software development process models were discussed, i.e., *Software Process Improvement* (SPI). SPI is the activity where an implemented software development process is being improved to either meet a reference model or a set of business goals external to the development process. These two approaches have a common goal: to make the implemented software development process better for the organization, but they look at it from two different perspectives. *Software metrics* are closely related to SPI, since they provide a measure of the process improvement and guide to the actions necessary with regard to the goal of the SPI. Next step is to apply the GQM model in a lightweight project assessment method used in Small firms for internal audits. The assessment method is used in conjunction with SPI to characterize and improve the agility of processes.

A. **Software Process Engineering Activities**

The reference models that describe a static set of activities related to process engineering are 1. SW-CMM 1.1 2. Trillium 3.0 model 3. ISO 12207 process standard 4. Bootstrap 3.0 model 5. 15504 process reference model The models have not been presented in any particular order. As these models describe the activities of software engineering in general, not just Software Process Engineering activities.

SW-CMM 1.1

The key process areas (KPA's) in the SW-CMM 1.1 are not functional processes as such but rather characteristics in the overall software development process (Kinnula 1995, 51). However, by studying the activities in the KPA's one can discern process-like entities. The KPA's that relate to process engineering in the SW-CMM 1.1 are: Organisational Process Focus (OPF), Organisation Process Definition (OPD), Training Program (TP), Integrated Software Management (ISM), Quantitative Process Management (QPM), Defect Prevention (DP), Technology Change Management (TCM), Process Change Management (PCM).

Essentially the OPF is about the entire Software Process Engineering system within an organisation. The other key process areas relate to specific activities within the scope of Software Process Engineering. Process definition work (OPD) is about creating and managing process assets, ISM is about refining and deploying those assets, supported by training (TP). Technology and process change management (TCM and PCM respectively) cover the activities that aim at improving the process assets. In addition, through the key process areas of QPM and DP, one can see the activity of process management through metrics in general, although the actual KPA's are about specific processes (project management processes and processes where defects are being injected, respectively), those two having been identified by the model authors as key issues for achieving process stability and quality.

Trillium Model

Trillium model has devoted the capability area “Process" for process engineering activities and practices. The four elements within that capability area address the product development process -related issues, including its development, improvement and maintenance. They are:

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 7, July 2013

Process Definition covers the practices that address the formalization and coverage of the process. It involves activities such as definition, development, documentation and maintaining the processes, and establishing and maintaining process asset repository. *Technology Management* covers the practices that address the monitoring, assessment and introduction of technology into the process. It involves activities such as identifying the need for new technologies, and selection, evaluation, piloting, acquisition and introduction / implementation of the new technologies. A technology can be a method, technique or a tool. *Process Improvement and Engineering* covers practices that address process improvement activities. It involves activities such as process assessments, co-ordination of process improvement and definition activities, planning and tracking process improvement projects, deploying improvements, collecting, recording and analyzing process data, and quantitative process management. *Measurements* covers practices that address the measurement system and its elements. It involves activities such as metrics identification, collecting, analyzing and storing measurement data, communicating process analysis results, and statistical process control.

ISO 12207

The ISO 12207 standard identifies "Improvement" as one of the processes in the "Organizational" process class. These are the basic, top-level activities that are needed to assess, measure, control and improve the organizational life-cycle processes. (ISO/IEC 1995). The activities within the Improvement process are: Process Establishment, Process Assessment and Process Improvement

As the ISO 15504 process reference model has been aligned with the ISO 12207, and ISO 15504 being more recent and more comprehensive as models go, these three processes will be described in section 2.5, under "ORG.2 Improvement processes".

Bootstrap

The Bootstrap model v. 3.0 has been structured to correspond with ISO 15504 v. 2.0 process architecture, which has been largely carried over to the newer, 1998 version of ISO 15504. Those processes, which can be found both from Bootstrap and the ISO 15504 model will be described in section 2.5. Those process engineering -related processes that can be found only from Bootstrap are discussed in this chapter. To help the reader to find the relevant process descriptions, a mapping of names between the elements in Bootstrap, ISO 15504 v.2.0 and the 1998 version of ISO 15505 has been provided.

The Bootstrap model divides processes into three main categories: Organization, Methodology and Technology. The Methodology category is further divided into Life cycle dependent, Life cycle independent and Process-related subcategories. Processes that are related to Software Process Engineering can be found from the Organization category, Methodology/Process-related –category and Technology category.

ISO 15504-2 and ISO 15504-5

The ISO 15504 reference model is the latest and arguably the most comprehensive software process-oriented reference model currently available for Software Engineering. The authors of CMM, Trillium, and Bootstrap have all participated in the definition effort, suggesting that this model is close to being a superset of all those three. In this section the focus is on the newer, 1998 version of the ISO 15504, but a brief comparison to the version 2.0 is provided as well.

The previous version (version 2.0) of the ISO 15504 reference model is quite different compared to the 1998 version, as far as process engineering –related processes are concerned. The earlier version identifies only two processes that fall within the scope of SPE. These are (ISO/IEC 1996a, b).

B. **SPI Based on Experience and Business Goals**

Software engineering offers a framework called *Quality Improvement Paradigm* (QIP) to improve the quality of the software development process. This paradigm works in strong cooperation with other paradigm, the *Goal/Question/Metrics Paradigm* (GQM), which supports the establishment of project and business goals and a mechanism for measuring against those goals. These two paradigms are usually used inside an infrastructure called Experience Factory (EF) which defines a set of practices to create packages of experience collected from past projects and reuse them in an organization. The three paradigms (QIP, GQM and EF) provide a unified framework for software process improvement based on experience and business goals. **Experience Factory**

An important asset of any company is the business knowledge that has accumulated during years of experience. Higher quality at lower cost is usually achieved by reusing processes, knowledge and experience from similar projects that have been successful in the past. In his paper Victor Basili presents an infrastructure called *Experience Factory* (EF) for improving the quality of software processes by systematically saving and reusing experience from previous projects. It is important to distinguish the experience factory infrastructure from process reference models: the former improves the development process by analyzing business goals, while the latter assesses the process against a given predefined process model that needs to be evaluated against the business needs of an organization. The experience factory infrastructure defines two distinct organizations: the development organization and the experience factory. The experience factory is a logical organization that supports project development by collecting and analyzing experiences from previous projects, by acting as a repository for such a knowledge and by packaging experience into reusable

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 7, July 2013

knowledge packages. The development organization represents the R&D part of the main organization, the ones that actually uses and works by the processes. They also provide the experience factory with all project and environment characteristics, development data, resource usage information, quality records and general feedback from the performance of the models and tools in use.

C. **Quality Improvement Paradigm (QIP)**

The basic tool for successful implementation of experience factory is the methodology called Quality Improvement Paradigm (QIP). There are several other process improvement paradigms but since QIP evolved from the lessons learned in the SEL project at the same time as EF. The QIP provides two iterative feedback loops.

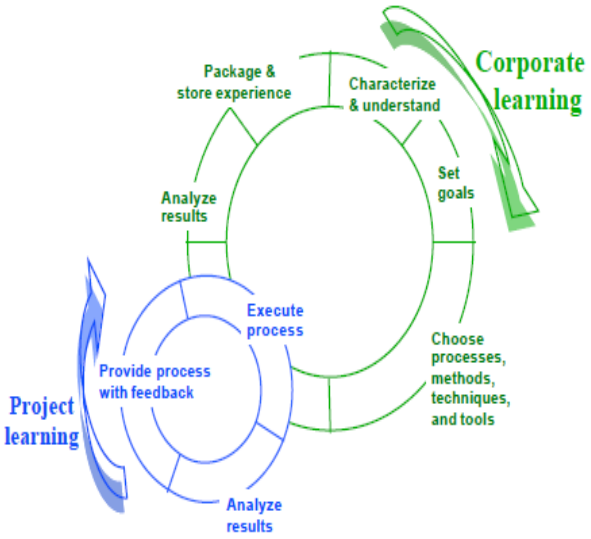

Fig.3 Quality improvement paradigm feedback loop

The QIP cycle is broken into two closed loop cycles – the corporate (larger) and the project (smaller) cycle. The project specific feedback cycle is to provide feedback to the project during the execution phase in order to prevent and solve problems, monitor and support the project and to realign chosen processes with defined goals. The Corporate feedback cycle provides feedback to the organization after the completion of the project. The purpose of the Corporate feedback is to analyze the concordance and discrepancy of the collected data against previous experiences and models. This helps to increase the understanding of the concluded experience and to capture some of that experience, and to accumulate reusable experience in the form that can be used by other projects. The Corporate cycle represents how organization learns. It is divided into following six phases: 1. Characterize and Understand 2. Set Goals 3. Choose Processes, methods, techniques and tools 4. Execute the Processes (run the project cycle) 5. Analyze Results 6. Package and store experience

The QIP cycle can be used both as a tool to learn more of already existing packaged experiences, or to create completely new, packaged experiences. The QIP cycle itself does not change, but if the goal is to produce a new experience and package it for future reuse, the fourth phase (Execute the Processes) requires several iterations. The reason for this is that the experience should not be packaged based on one single case, but requires several experimentations until there is sufficient knowledge of where it works and what it requires to work. The *Characterize and Understand* is the starting phase for the cycle. The aim is to describe and comprehend the current project and its environment with respect to available process/product/ quality models, data, intuition, etc. The phase also establishes quantifiable baselines based on past experiences and characterizes their criticality. The second phase is to *Set Goals* for successful project performance (covering both processes and products) and improvement results. The aim is to be able to get reliable, measurable data of the improvement and for this reason the goals need to be quantifiable and defined from a variety of perspectives, including customer, project and organization viewpoints. The objective of the third phase is to *Choose Processes, Methods, Techniques and Tools* that are appropriate for this project. The decision is based on the characterization of the environment and product and on the goals that have been set. It is important to make sure that the selection is consistent with the goals set for products and processes, since otherwise there is little point in doing the measurements derived from the goals. The fourth phase of the Corporate cycle is where the selected project(s) *Execute the Processes*. From organization point of view, this phase is where the project cycle runs. The project cycle, which represents how project learns and guides itself, is divided into three activities: Process Execution, Analyze Results and Provide process with feedback.

The fifth phase is to *Analyze Results* to evaluate the practices, determine problems, record findings and make recommendations for future project improvements. The data is analyzed against the goals and used to achieve better characterization and understanding of the context, evaluate and analyze the experiments (improvements), determine problems, gain more information to be used for better prediction and control and to motivate future improvements. The sixth and final phase on the Corporate cycle is to *Package Experiences* and to store them in the experience base for future reuse. It should be noted that if the QIP cycle is used for improving processes through experimenting with new procedures, methods or tools, it may require several cycle iterations and projects before there is sufficient information for packaging the experiences.

D. **Goal/Question/Metric paradigm (GQM)**

Feedback is an essential part of any improvement, and software process improvement does not make any

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 7, July 2013

exceptions. Software metrics makes up for the feedback needed for software process improvement. Only with correctly chosen metrics and valid data can a process be assessed with regards to its progress and to help us support project planning in upcoming projects. But more importantly, metrics helps us to determine the strengths and weaknesses of the current processes and products (to form a baseline) and it provides a rationale for adopting and refining the techniques needed to determine if a process has improved or not.

All measurements of software process improvement must be done in a top-down fashion, since there are very many metrics to measure in a software process and since process improvement and business goals must determine which ones are relevant. In his paper Victor Basili describes the *Goals/Question/Metric Paradigm* (GQM) as a method for defining and interpreting operational and measurable software. In the Experience Factory infrastructure, GQM is the method for defining the business goals and the data to measure.

IV. Approach for Process Assessment

A need for quantitative process assessment emerged in Small firms from internal assessments of a part of the regular internal quality initiatives. The goal of the assessment approach in to assess the agility of the processes, but in a lightweight fashion to minimize project overhead expenses and also to collect quantitative metrics for SPI of those processes. The process is kept lightweight by formulating a set of questions with quantitative metrics as answers. The metrics should be easy to obtain from a process management tool. The development process is broken down into three groups with questions in each of them; *project and requirement management*, *development* and *testing*. Each question has a point scale that the answers can be compared against; the overall agility is reflected by the sum of the points. The metrics in each of the groups give indications for the SPI initiatives to spot bottlenecks.

The two agile development models, XP and Scrum, were described as a response to changing customer needs and requirement prioritizing. They work in short iterations where the customer needs and working, tested, software is valued over anything else. SPI is described which introduces methodologies for improving development models set up by organizations. The two approaches of SPI are considered; Process Reference Models and Experience Factory. In the first, a development process was assessed against a given reference models that contains best practices for the process, while the latter uses business goals and metrics to create experience packages of best practices best suitable for the given organization (QIP and GQM). After looking at both of these development models and how to improve them, Lightweight assessment method for agile processes is preferable. The goal of this method is to assess the agility of a process using quantitative metrics and a checklist with predefined questions that are divided into three project areas (project and requirement management, development and testing).

V. Conclusion

The assessment method presented in this paper has been developed to assess a single small agile team with one project manager, may be one software architect and a few developers and testers. It is certainly possible to refine the method by collecting assessment metrics results from several projects to measure the overall agility of the development in an organization, but the variability of agile development processes should be then taken into account since there is no official method of doing agile development. The presented assessment method is based on the assumption that the development process uses Scrum and XP to introduce agility into development. Probably a different set of questions should be developed for projects using Crystal Clear or Agile Unified Process (AUP). Also the same methodology could be used for assessing different kinds of processes and a completely different set of questions and points should be considered for assessing a process based on RUP. Similarly, it could also be possible to quantify the usability of an user interface of an application. Then it would be interesting to examine how to integrate the results of the assessment of two such different aspects of a software engineering project into an overall result. The assessment method presented in this is a good starting point for future research and work. The demand for lightweight process assessment used internally by companies as a part of SPI is huge. We have seen that this method fits well with Experience Factory and SPI methods, but more work is required to adopt this assessment method on a corporate level and also to combine GQM into this method to measure business goals in cooperation with agile process models

References

[1] Brooks Jr., F. P. : *No Silver Bullet - Essence and Accidents of Software Engineering*, Computer Magazine, Volume 20 , Issue 4 (April 1987)

[2] International Software Benchmarking Standards Group: *Practical Project Estimation 2nd Edition*, http://www.isbsg.org (2005)

[3] Ivar Jacobson, Grady Booch and James Rumbaugh: *The Unified Software Development Process*, ISBN-13: 978-0201571691, Addison-Wesley Professional, February 4, 1999

[4] *IBM Rational Process Composer*, http://www- 306.ibm.com/software/ awdtools/rmc/features/index.html - checked December 2007.

[5] *Eclipse Process Framework Project (EPF)*, http://www.eclipse.org/epf/ - checked December 2007.

[6] Beck, Kent : *Extreme Programming Explained*, Addison-Wesley, 2000

[7] Schwaber, Ken : *SCRUM Development Process*, Advanced Development Methods, 1996.

[8]ISO9001:2000,http://www.iso.org/iso/iso_catalogue/management_standards.htm, checked March 2008.

[9] Mutafelija, Boris and Stromberg, Harvey: *Mappings of ISO 9001:2000 and CMMI Version 1.1*, Software Engineering Institute, July 2003, http://www.sei.cmu.edu/cmmi/adoption/pdf/iso-mapping.pdf, checked March 2008

ISSN (Print) : 2319-5940
ISSN (Online) : 2278-1021

International Journal of Advanced Research in Computer and Communication Engineering
Vol. 2, Issue 7, July 2013

[10] Software Engineering Institute (SEI): *CMMI for Development, Version 1.2*, August 2006.
[11] *The CMMI website*, http://www.sei.cmu.edu/cmmi/ - checked January 2008.
[12] Gerard O'Regan, *A Practical Approach to Software Quality*, Springer-Verlag, New York, 2002.
[13] *SPICE specification - Part 2 : A model for process management, Version 1.00*, ISO/IEC, July 1995.
[14] *IT Infrastructure Library (ITIL)*, http://www.itil-officialsite.com/home/home.asp, checked March 2008.
[15] V. Basili, G. Caldiera and D. Rombach: *The Experience Factory*, Encyclopedia of Software Engineering. Wiley 1994.
[16] V. Basili, F. McGarry, R. Pajerski, M. Zelkowitz: *Lessons learned from 25 years of process improvement: the rise and fall of the NASA software engineering laboratory*, Proceedings of the 24th International Conference on Software Engineering, May 19- 25, 2002, Orlando, Florida.