Software Quality Assurance: The CMM and XP approach

[1]Faheem Ahmad, [2]Agha Urfi Mirza, [3]Burhanuddin Mohammad, [4]Mohammed Abdul Habeeb

[1,2,3,4] Department of Information Technology, Al-Musanna College of Technology
Sultanate of Oman

(Received Jan. 2015 & Published online March. 2015 issue)

Abstract: Software Quality Assurance is an intended and systematic set of activities essential to provide adequate confidence that requirements are properly established and products or services corroborate to specified standards. Successful software engineering strongly depends on the delivery of high quality software. In the present paper, we compare Capability Maturity Model (CMM) and Extreme Programming (XP) regarding their software quality support in terms of software quality development and software quality assurance and also we presented Software Quality Assurance Proposed by ISO 9000-3.

1. Introduction

"Software engineering is the application of science and mathematics by which the capabilities of computer equipment are made useful to man via computer programs, procedures and associated documentation".

Thriving software engineering strongly depends on the deliverance of high quality software. The support of software quality in a software development process may be considered as two facets: one by developing techniques which is used in the development of high quality software and the other by developing techniques which assure the desired quality attributes in the existing software.

The software quality engineering focuses on the processes involved in the development and establishment of software quality. Software quality engineering includes software quality development and software quality assurance. Software quality development consists of requirements engineering, system and software design and implementation. Software quality assurance consists of software quality assurance, quality management and verification and validation. Software quality is achieved by three approaches: testing and static analysis and development approaches. The integration of all three approaches is the most desirable approach.

Software quality assurance is an umbrella activity that is applied at each step in the process of building the software. It is a planned and systematic set of activities necessary to provide adequate confidence that requirements are properly established and products or services confirm to specified standards". Software quality assurance is defined as "A planned and systematic pattern of actions that are required to ensure quality in software [2]."

Different users think differently about the quality of software. The end-user expects the software to help him to do the job faster and easier with adequate help. The buyer expects 2 the software to meet the specifications within the contract terms. The developer attempts to trace defects and focuses faster development as well as higher productivity. The maintainer expects software to be understandable, testable, and modifiable, with all documentation.

The characteristics of software quality in product transition reusability, portability and interoperability. The characteristics of software quality in product revision are maintainability, adaptability and expandability. The characteristics of software quality in product

(DOI: dx.doi.org/14.9831/1444-8939.2015/3-3/MAGNT.167)

operation are usability, security, efficiency, correctness and reliability. The attributes of software quality are manageability, efficiency, safety, expandability, reliability, flexibility and usability.

There are quantitative as well as qualitative benefits in maintaining quality assurance. The Quantitative benefits are reduced costs, greater efficiency, better performance, less unplanned work and fewer disputes. The Qualitative benefits are improved visibility and predictability, better control over contracted products, improved customer confidence, better quality, problems show up earlier and reduced risk.

2. Software quality assurance activities:

- Application of technical methods.
- Software Testing
- Control of change
- Conduct of formal technical reviews
- Enforcement of standards
- Measurement
- Record keeping and reporting

3. Software quality assurance proposed by ISO 9000-3:

ISO 9000-3 is the standard of the ISO 9000 series that is most relevant to software development and maintenance. Organizations typically use ISO 9000 standards to regulate their internal quality systems and assure the quality systems of their suppliers. ISO proposes a quality assurance manual that consists of management responsibilities, a set of measurements, analysis and improvement activities and required documentation. An ISO 9000 organization should have implemented a Quality Management System (QMS) that is continuously maintained for effectiveness and process improvement. The effectiveness of the Quality Management System should be improved by the use of quality, policy, quality objectives, audit results, analysis of data, corrective and preventive actions and management reviews. The organization defines and documents its policy which provides the overall objectives for an effective Quality Management System. The quality policy should be relevant to the organization goals and expectations of its customers. ISO 9000 requires an organization to plan and perform audits. The results of audits are communicated to management and deficiencies found are corrected.

ISO 9000 states that organizations must establish adequate statistical techniques and use them to verify the acceptability of the process capability. This is also called measurement. According to ISO 9000-3 “there are currently no universally accepted measures of software quality”. The auditors can accept the use of statistical tools or any consistently collected and used data.

The organization should implement and maintain documented procedure to initiate corrective and preventive actions. Corrective action procedures define the requirements for:

- Reviewing non-conformities including customer complaints.
- Determining causes of non-conformities.
- Evaluating the need for action to ensure that non-conformities do not recur.
- Determining and implementing the action needed.
- Records of the results of action implemented.
- Review of corrective action implemented.

The SQA manager is responsible for corrective and preventive actions and a feedback system should be used to provide early warnings of quality problems. Preventive action procedures define requirements for:

- Determining potential non-conformities and their causes.
- Evaluating the need for action to prevent occurrence of non-conformities.
- Determining and implementing the action needed.
- Records of the results of action implemented.
- Reviewing preventive action implemented.

(DOI: dx.doi.org/14.9831/1444-8939.2015/3-3/MAGNT.167)

The QMS documentation structure can be described at five levels:

Level1: is maintained in the form of quality policy. Level 2: documentation is maintained in the form of quality assurance manual.

Level 3: consists of quality procedure. Level 4: contains work instructions.

Level 5: documentation is maintained as records/reports.

4. Capability Maturity Model:

Software process capability describes the range of expected results that can be achieved by the following process [3]. The process capability of an organization determines what can be expected from the organization in terms of quality and productivity. The goal of process improvement is to improve the process capability. A maturity level is a well-defined evolutionary plateau toward achieving a mature software process. Based on the empirical evidence found by examining the processes of many organizations, the CMM suggests that there are five defined maturity levels for software process. These are initial (level 1), repeatable (level 2), defined (level 3), managed (level 4) and optimizing (level 5). The CMM framework says that as process improvement is best incorporated in small increments, processes go from their current levels to the next higher level when they are improved. Hence, during the course of process improvement, a process moves from level to level until reaches level 5.

5. Software quality assurance proposed by CMM:

It is well known the CMM describes an evolutionary improvement path to a mature disciplined process.

CMM defines key practices to improve the ability of the organization to meet goals for cost, functionality and quality. SQA activities are defined at level 2.

According to CMM the purpose of software quality assurance (SQA) is to provide the management with appropriate visibility into the process being used by the software project and of the products being built. It is required that the project follows a return organizational policy for implementing the SQA.

CMM defines eight activities to be performed as follows:

- A SQA plan is prepared for the software project according to documented procedure.
- SQA's group activities includes:
    - Responsibilities and authority of SQA group of Resource requirements of SQA group
    - Schedule and funding of the project.
    - Participation in establishing the software development plan (SDD).
    - Evaluations to be performed.
    - Audits and reviews to be conducted.
    - Projects standards and procedures forming basis for SQA reviews.
    - Procedures for documenting and tracking non-Compliance issues.
    - Documentation to produce.
    - Method and frequency to provide feedback to other related group.
- The SQA group participates in the preparation and review of the project's software development plan, standards and procedures and audit the software project.
- The SQA group audits designated software work products to verify compliance.
- The SQA group periodically reports the result of its activities to the software engineering group.
- Deviations identified in the software activities and software work products are documented and handled according to documented procedure.

(DOI: dx.doi.org/14.9831/1444-8939.2015/3-3/MAGNT.167)

- The SQA group conducts periodic reviews of its activity and findings with customers SQA personnel as appropriate.

CMM levels key process areas and their purpose:

5.1 Initial:

This is the starting point for use of a new or undocumented, repeated process. Little documentation is necessary if any processes and procedures take place. Success is only achieved by the heroic actions of team members.

When to use:

Used for a kind projects of very limited scope.

5. 2 Repeatable:

The process is at least documented sufficiently such that repeating the same steps may be exempted. Enough documentation exists that the QA process is repeatable.

When to use:

This is used for any project that will be done again, whether as an upgrade or somewhat similar variation.

5. 3 Defined:

The process is defined/confirmed as a standard business process, and decomposed to levels 0, 1 and 2 (the latter being Work Instructions).

QA documentation and processes & procedures are standardized. Templates exist for all documentation and a QA "system" exists.

When to use:

This is critical for a QA department that must provide QA for multiple projects. This avoids reinventing the wheel for each project.

5. 4 Managed:

The process is quantitatively managed in accordance with agreed-upon metrics. The exact time & resources required to provide adequate QA for each product is known precisely so that timetables and quality levels are met consistently.

When to use:

This requires an existing data set based on previous QA projects. This level can only be achieved by well documented experience.

5. 5 Optimizing:

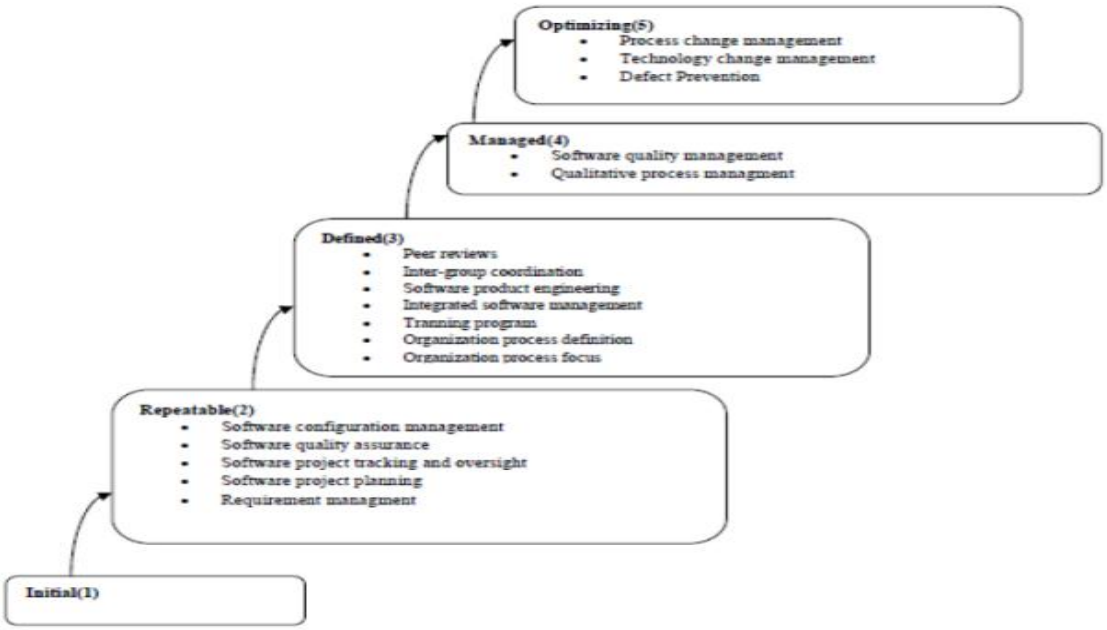
Process management includes deliberate process optimization/improvement. QA processes and procedures are understood well enough to be refined and streamlined.

When to use:

This should be actually used in every stage. In Level

(DOI: dx.doi.org/14.9831/1444-8939.2015/3-3/MAGNT.167)

5, this is the only thing left to work on.



It would be enlightening to conduct a CMM assessment of a team successfully practicing XP. In fact, XP team would achieve a maturity level 2 or better. CMM level 2 is about managing project requirements and schedules effectively and repeatedly. XP claims to do just that, using story cards and a planning game. Thus, the software engineering goals are worthy and they can even be implemented with lightweight methodologies where appropriate. XP is compatible to CMM as well. Software quality assurance consists of Software quality assurance, quality management and verification and validation. Software quality is achieved by three approaches: Testing, Static analysis and development approach. The integration of all the three approaches is the most desirable approach. A different categorization of approaches towards software quality regards four ways to establish software quality: Software quality via better quality evaluation, better measurement, better processes and better tools.

Large- scale quality models like Capability Maturity Model (CMM) or ISO-9001 tend to form a SQA in terms of a "process police". SQA takes care only that the process requirements are met but does not consider the quality of the process itself. Instead of SQA in terms of CMM or ISO 9001 a better solution is to embed quality evaluation in the development process.

XP require certain adaptations in order to fulfill CMM requirements specialized maturity models for XP are introduced by combining Capability Maturity Model (CMM) with Personal Software Process (PSP).Therefore, instead of eliciting SQA in terms of CMM a better solution can be embedded for quality evaluation in XP .

6. Software quality assurance proposed by XP:

6.1. Iterative software development:

To establish higher software quality, a software development process has to use an iterative and incremental development approach. By using iterative approach a process can gain more flexibility in dealing with changing requirements or scope. The Short Releases of the product force early feedback from the customer as well as stakeholders which is important for improvement of overall quality of the software. XP builds on a very strict iterative approach limiting the time needed to encounter errors and forces developers to fix the problem as soon as possible.

(DOI: dx.doi.org/14.9831/1444-8939.2015/3-3/MAGNT.167)

6.2. Quality as a primary objective:

XP software development process defines quality as a major objective to improve overall quality of the software. Quality targets have to be defined by involving project team members and customer (On-Site Customer). Thus the quality goals become achievable and measurable.

6.3. Continuous verification of quality:

This includes extensive testing. Besides internal unit testing, external acceptance tests with the customer are needed too, in order to verify that the product fulfills the needs and requirements of the customer (Test-Driven Development).

6.4. Customer requirements:

The requirements of the customer who normally does not have a deep technical knowledge have to be considered, so that developers are able to build an application based on that information. Thus it is necessary that the project team understands the customer and his business. Otherwise it is not possible to implement the customer needs accurately. XP teams focuses on the customer needs and requirements throughout the entire project by means of communication and by framing user stories.

6.5. Architecture driven:

Architecture of a system has a major impact on the overall quality of the product. Using a simple well-designed architecture allows easy integration and reuse (Simple Design and Continuous Integration).

6.6. Focus on teams:

Focusing on team work also effects the motivation of project members. Seeing everyone as an equally important part of the project leads to a high identification of the team members with the product. Hence the project code is not owned by any single programmer but owned by the team collectively (Collective Code Ownership).

6.7. Pair programming:

Better solutions are more likely with Pair Programming since two persons most likely have different perspectives of the same problem and therefore they complement each other in solving it. This approach saves time and minimizes the number of errors. This is an explicit practice of XP.

6.8. Tailoring with restrictions:

Software development process should rely on core elements. Building on these core elements the process should adapt practices (tailoring) according to the project type and project size (e.g. RDP)

6.9. Risk management:

Risk management enables early risk mitigation and the possibility to act instead of to react to problems and risks. A well-defined risk awareness and mitigation management form together an effective risk management and is a key factor in achieving high product quality.

7. Conclusion:

Thus, Practices of XP support software quality development as well as software quality assurance. XP require certain adaptations in order to fulfill CMM requirements specialized maturity models for XP are introduced by combining Capability Maturity Model (CMM) with Personal Software Process. However, much software quality support is implicitly present in XP principles.

(DOI: dx.doi.org/14.9831/1444-8939.2015/3-3/MAGNT.167)

References:

1. B.W.Boehm. ***Software Engineering Economics***. Prentice Hall, Englewood Cliffs, NJ, 1981. Ward, W.A., and Venkataraman.B, ***Some observations on Software quality***, in proceedings of the 37th annual southeast regional conference (CD-ROM), ACM, 1999, Article No.2.
2. Microsoft Cooperation: ***Microsoft Solutions Framework White Paper***, Microsoft Press, 1999.
3. Huo, M., Verner, J., Zhu, L., Babar, M.A: ***Software quality and agile methods.*** In proceedings of COMPSAC 04, IEEE Computer Soc., 2004, pp.520-25.
4. Paulk, N.C: ***Extreme Programming from a CMM Perspective.*** IEEE software, vol. 18, no.6, IEEE, Nov-Dec.2001, pp.19-26.
5. Nawrocki,J., Walter, B.,and Wojciechowski, A.: ***Toward maturity model for Extreme Programming:*** In proceedings Euromicro Conference, 2001.IEEE,2001,pp. 233-9.
6. Baker, E.B., ***Which way, SQA?*** .IEEE-Software, vol.18, no.1; Jan.-Feb. 2001; pp. 16-18 [8] ManZoni, L.V.; Price, R.T.: ***identifying extensions required by***
7. ***RUP (Rational Unified Process) to comply with CMM (Capability Maturity Model) level 2 and 3.*** IEEE Transaction on Software Engineering, Vol 29, no.2, IEEE, Feb.2003, pp.181-192.
8. Pollice, G.: ***Using Rational Unified Process for small Projects: Expanding Upon Extreme Programming.*** A Rational Software White Paper, Rational, 2001.
9. Runeson, P., Isacsson, P.: ***Software Quality Assurance Concepts and Misconceptions,*** In Proceedings of the 24th **EUROMICRO Conference, IEEE Computer Soc, 1998, pp.853-9.**
10. Osterweil, L.J.: ***Improving the quality of Software quality determination processes***, In, The Proceedings of the IFIP TC2/WG2.5 Working Conference on Quality of Numerical Software, Assessment, and Enhancement, Chapman & Hall, London, 1997, pp.90-105.

(DOI: dx.doi.org/14.9831/1444-8939.2015/3-3/MAGNT.167)