International Journal of Computer Network and Security(IJCNS)
Vol5. No.1 – Jan-March 2013 Pp. 5 -11
©gopalax Journals, Singapore
available at : www.ijcns.com

CMMI STANDARDS IN SOFTWARE DEVELOPING PROCESS

[1] **C. SenthilMurugan**, [2] **Dr. S. Prakasam.**
PhD Scholar Asst., Professor
[1,2]Dept of Computer Science & Application, SCSVMV University, Kanchipuram
[1]Dept of MCA , Thiruvalluvar College of Engineering and Technology, Vandavasi.

ABSTRACT

Software development and maintenance is the activity that is used to make the error-free Software and also concentrates on time-consuming and complex activity. The software quality management is used to evaluate the quality of a software product and to keep its level high is much more difficult than to do them for the other industrial products. The software quality is maintained to make the software as customer satisfied one. In this paper I have concentrated to keep the quality level of software products high, firstly the Software Quality Assurance activity process like CMMI Maturity levels are defined in the early stages of software development, it may reduce the problem that are occurring at the time of development.

Key words: Software Quality Assurance Software development, SQA process, CMMI Maturity level.

1. INTRODUCTION

The Software has been used for commercial purpose. With every aspect of software development, Software engineers have been tasked to solve the large and complex programs and in a cost effective and efficient manner. Also the development and maintenance of the software product has become an important criterion. The Software quality assurance is implemented in the early stages of software development life phases, this is used to make the error free software and reduce the rework of development.

In the early years, engineers faced many problems, without having a better knowledge in the software fields, such as "Late delivery of software, Development team exceeding the budget, poor quality, user requirement are not completely supported by the software, difficult maintenance and unreliable software and lack of systematic approach, this problem are overcome by the implementation of software quality activities.

To develop a software product, the following criteria has to be satisfied:

- User needs and constraints must be determined and explicitly stated.
- The source code must be tested thoroughly.
- The product must satisfy the user needs
- Supporting documents such as user guide, installation procedures and maintenance documents must be prepared.

Software Engineering

Software engineering is an emerging discipline that focuses on the creation, development, operation and maintenance of cost effective, reliable correct and high quality solution to software problems and it is the application of a systematic, disciplined and quantifiable approach

With the development, operation, and maintenance of software.

The software engineering is useful

- To acquire skills to develop large programs
- Ability to solve complex programming problems
- Learn the techniques
- To acquire skills to be a better programmer

The primary goal of software engineering is to improve the quality of software products and to increase the productivity and job satisfaction of software engineers.

2. SOFTWARE QUALITY ASSURANCE (SQA)

A systematic, planned set of actions necessary to provide adequate confidence that the software development process or the maintenance process of a software system product conforms to

International Journal of Computer Network and Security(IJCNS)
Vol5. No.1 – Jan-March 2013 Pp. 5 -11
©gopalax Journals, Singapore
available at : www.ijcns.com

established functional technical requirements as well as with the managerial requirements of keeping the schedule and operating within the budgetary confines. The difference between the quality control and quality assurance should be recognized. Quality control activities are done to sort the products that do not qualify for the qualified products to not deliver the customer or to not sell in the market.

A Software process is a set of activities and associated results which produces software products. These activities are mostly carried out by software engineers. There are four fundamental process activities which are common to all software process.

These activities are

1. Software Specification:
   The functionality of the software and constraints on its operation must be defined.

2. Software Development:
   The software to meet the specification must be produced.

3. Software Validation
   The Software must be validated to ensure that it does what the customer wants.

4. Software Evaluation
   The software must evolve to meet changing customer needs.

2.1Common Process Framework

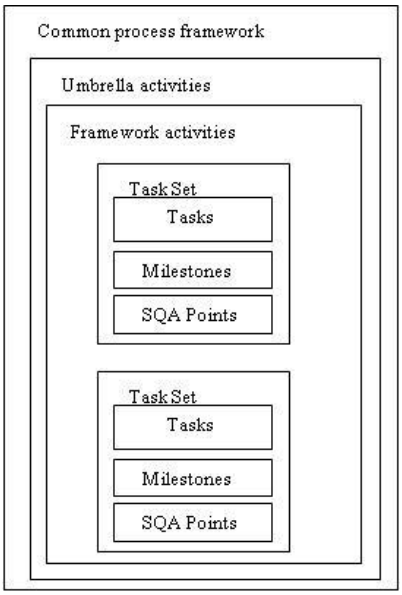

Figure-1.1: Software Process framework

Task Sets: A number of task sets-each a collection of software engineering work tasks, project milestone, software work products and deliverables and software quality assurance points.

Framework Activities: The task sets enable the framework activities to be adapted to the characteristics of the software project and the requirements of the project team. The generic process framework activities are

1. Communication: The customer requirement information gathering is done by communication.
2. Planning: The planning activity defines the engineering work plan, describes technical risks, list resource requirements, work products produced and defines work schedule.
3. Modeling: The modeling activity defines the requirement analysis and design.
4. Construction: The construction activity implements the corresponding coding and testing.
5. Deployment: The software delivered for customer evaluation and feedback is obtained.

2.2 Umbrella Activities

Umbrella activities –such as quality assurance, software configuration management and measurement. Umbrella activities are independent of any one framework activity and occur throughout the process. The umbrella activities are

1. Software project tracking and control: The activity helps to access the software team progress and take corrective action to maintain schedule.
2. Risk management: This activity analysis the risk that may affect the quality.
3. Software quality assurance: This activity maintains the required software quality.
4. Formal technical reviews: This activity helps to analyze the engineering work products to uncover and clear errors before they proceed to the next activity.
5. Software configuration management: This activity manages the configuration process when any change in the software process.
6. Work product preparation and production: This activity defines the model, document forms and lists to be carried out.
7. Reusability management: This activity defines the work product reuse.

International Journal of Computer Network and Security(IJCNS)
Vol5. No.1 – Jan-March 2013 Pp. 5 -11
©gopalax Journals, Singapore
available at : www.ijcns.com

8. Measurement: This activity defines the project and product measure to assist the software team in delivering the required software.

3. PROBLEM FORMULATION

Software Errors: The error resulting from bad code in some program involved in producing the erroneous result.

Software faults: Due to the Software errors the Software fault occurs.

Software failures: Due to Bug/defect/fault consequence of a human error.

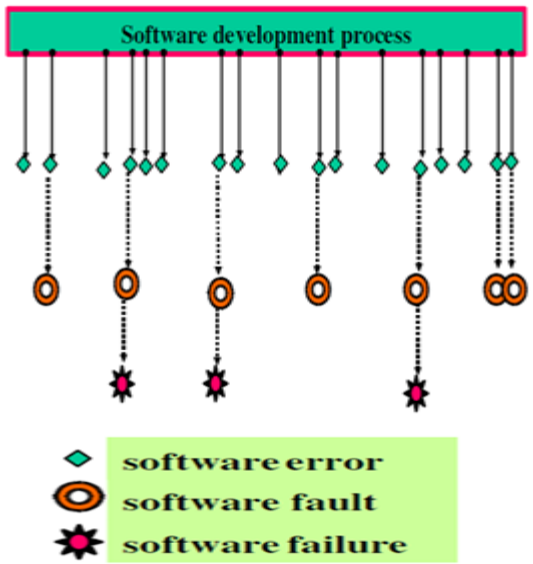

Figure.3.1 Software Errors, software faults and software failures

3.1 Nine Causes of Software Errors

1. Faulty requirements definition
2. Client developer communication failures
3. Deliberate deviations from software requirements
4. Logical design errors
5. Coding errors
6. Noncompliance with documentation and coding instructions
7. Shortcomings of the testing process
8. User interface and procedure errors
9. Documentation errors

3.2 Development process relating to defects

Majority of defects is introduced in earlier phases

Table-1: Majority of defects

Phase	Percentage of defects	Effort to fix defects
Requirements	56	82
Design	27	13
Code	7	1
Others	10	4

Relative cost of fixing defects

Table-2: Cost of fixing defects

Phase in which found	Cost Ratio
Requirements	1
Design	3-6
Coding	10
Unit/Integration testing	15-40
System/Acceptance testing	30-70
Production	40-1000

4. PROCESS OF SQA SYSTEM

The processes are classified into two main groups:

1. Organizational processes, and
2. Departmental/Project processes.

For example,

1. Galin explains an SQA system as dividing its components five major classes
2. Pre-project quality components,
3. Project life cycle quality components,
4. Infrastructure error preventive and improvement components,
5. Software quality management components,
6. Standardization, certification and SQA assessment components,
7. Organizing for SQA-the human components.

International Journal of Computer Network and Security(IJCNS)
Vol5. No.1 – Jan-March 2013 Pp. 5 -11
©gopalax Journals, Singapore
available at : www.ijcns.com

4.1 Life-Cycle Phases of SQA System Development:

The implementation order, start point, duration and end point of the SQA processes are analyzed.

First of all, the phases of Software Development Lifecycle and SQA System Development Lifecycle should be explained. Software Development Lifecycle is a subset of the SQA System Development Lifecycle. Here, the classical waterfall model is chosen as Software Development Lifecycle model

1. *Requirements Definition/Analysis*: In this phase, software engineers gather customer requirements by defining them with the help of customer and domain experts. Most of the time, a developed software must have interfaces with existing hardware and software. Therefore the information about them helps to define the interface requirements. In this phase, software engineers must understand the nature of the program to be built. Therefore, understanding the information domain, system's required functions, behaviors, performance and interface are musts.

2. *Design:* In this phase, software designer identifies the system inputs, outputs and processes. Processing algorithms, data structures, databases and software structures are also defined.

3. *Code Generation:* In this phase, software engineers and programmers transform the design into a code by using a selected programming language. Code review, unit tests, unit integration tests is part of this phase.

4. *System Testing:* In this phase, the system is tested as a whole and system integration is realized. Activities are performed by the testing team.

5. *Installation and Conversion:* After customer approval, the software is installed to serve the customer. If the new software will be used to replace the existing software, a suitable conversion process must be performed to prevent the interruption in the organization's services.

6. *Operation and Maintenance:* Operation phase begins after the installation and conversion is completed. Maintenance activities are performed during the normal operation period which generally continues a few years.

SQA System Development Lifecycle includes the above software development lifecycle phases, but it has a few phases prior to software development lifecycle phases:

1. *Pre-Project Phase:* In this phase, organizational level activities will be performed. None of the activities of this phase are related to a specific project. During this phase, the SQA system of the organization is initiated, organization quality policy and QA methodology are defined, QA staff is assigned an initial SQA component are developed.

2. *Proposal/Contract Phase:* Activities of this phase are performed by the proposal team and legal department for a contract between the customer and the organization. During this phase, firstly proposal team develops a proposal draft from the customer requirements document. After reviewing a proposal draft with a customer, a contract draft is developed from a final approved proposal document. After reviewing the contract draft with the customer, a mutually agreed contract which defines source, timetable and cost estimation for the project is achieved.

3. *Project Preparation Phase:* In this phase, project products, project interfaces, development methodology, development tools, standards, procedures, project schedule, resource and cost estimation, project milestones, staff organization, quality goals, QA activities are identified.

5. CMMI FRAMEWORK

The CMMI framework is the current stage of work on process assessment and improvement that started at the software engineering institute in the 1980s. A capability level is a well-defined evolutionary plateau describing the organization's capability relative to a process area. A capability level consists of related specific and generic practices for a process area that can improve the organization's processes associated with that process area. Each level is a layer in the foundation for continuous process improvement.

The CMMI model is divided into five maturity levels:

1. Initial
2. Managed
3. Defined
4. Quantitatively Managed
5. Optimizing

International Journal of Computer Network and Security(IJCNS)
Vol5. No.1 – Jan-March 2013 Pp. 5 -11
©gopalax Journals, Singapore
available at : www.ijcns.com

Level 1 – Initial: In Level 1, processes are usually ad hoc and chaotic. The success of this organization depends on the competence and heroics of the people in the organization and not on the use of proven processes. In spite of this ad hoc, chaotic environment, maturity level 1 organizations often produce products and services that work, however, they frequently exceed the budget and schedule of their projects.

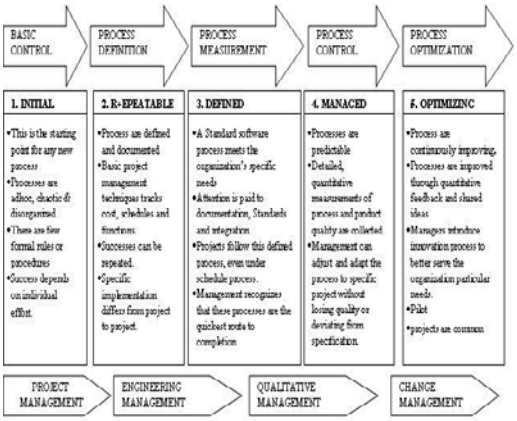

Figure. 5.1: Activities of CMMI

Level 2 – Repeatable: In Level 2, software development successes are repeatable. The organization may use some basic project management to track cost and schedule. Process discipline helps ensure that existing practices are retained during stressful times. As a result, projects are executed and managed according to techniques their documented plans.

Level 3 – Defined: In Level 3, processes are well characterized and understood, and are described in standards, procedures, tools, and methods. The organization's set of standard processes, which is the basis for level 3, is established and improved over time. These standard processes are used to establish consistency across the organization. Projects establish their defined processes by the organization's set of standard processes according
for tailoring guidelines.

Level 4 – Managed: In Level 4, management using precise measurements, can effectively control the software development effort. In particular, management can identify ways to adjust and adapt the process to particular projects without measurable losses of quality or deviations from specifications. Subprocesses are selected that significantly contribute to overall process performance. These selected subprocesses are controlled using statistical and other quantitative

Level 5 – Optimizing: In Level 5, the focus is on continually improving process performance through both incremental and innovative technological improvements. Quantitative process improvement objectives for the organization are established, continually revised to reflect changing business objectives, and used as criteria in managing process improvement. Process improvements to address common causes of process variation and measurably improve the organization's processes are identified, evaluated, and deployed. The organization's ability to rapidly respond to changes and opportunities is enhanced by finding ways to accelerate and share learning.

Audacious Inquiry is actively working to incorporate and embrace the CMMI methodology because we believe it will provide our organization with the essential elements to create improvement and effectiveness across projects, divisions, and the entire organization. As Audacious Inquiry continues to grow in size and technical expertise, CMMI will help integrate our separate organizational functions, set process improvement goals and priorities, and provide guidance for quality processes.

6. CMMI KEY PROCESS AREAS

A Process Area is a cluster of related practices in an area that, when implemented collectively, satisfy a set of goals considered important for making significant improvement in that area. All CMMI process areas are common to both continuous and staged representations.

The continuous representation enables the organization to choose the focus of its process improvement efforts by choosing those process areas, or sets of interrelated process areas, that best benefit the organization and its business objectives. Although there are some limits on what an organization can choose because of the dependencies among process areas, the organization has considerable freedom in its selection.

Once you select the process areas, you must also select how much you would like to improve the processes associated with those process areas (i.e., select the appropriate capability level). Capability levels and generic goals and practices.

International Journal of Computer Network and Security(IJCNS)
Vol5. No.1 – Jan-March 2013 Pp. 5 -11
©gopalax Journals, Singapore
available at : www.ijcns.com

The CMMI Process Areas (PAs) can be grouped into the following four categories

- Process Management
- Project Management
- Engineering
- Support

Table-3: Organization of Process Areas

Category	Process area
Process Management	• Organizational process definition • Organizational process focus • Organizational training • Organizational process performance • Organizational innovation and development
Project management	• Project planning • Project monitoring and control supplier agreement management • Integrated project management • Risk management • Integrated testing • Quantitative project management
Engineering	• Requirements management • Requirement development • Technical solution • Product integration • Verification • Validation
Support	• Configuration management • Process and product quality management • Measurement and analysis • Decision analysis and resolution • Organizational environment for integration • Causal analysis and resolution

6. 1 Goals of CMMI

The Goals are descriptions of desirable organization states. Each process area has associated goals.

- Corrective actions are managed to closure when the project's performance or results deviated significantly from the plan.
- Actual performance and progress of the project is monitored against the project plan.
- The requirements are analyses and validated and a definition of the required functionality is developed.
- Root causes of defects and other problems are systematically determined.
- The process is institutionalized as a defined process.

7. CONCLUSION

The SQA components were explained by considering the SQA system processes, and their inputs, outputs and sub-processes. They were classified as organizational level and department/project level and also the formulation of errors and the defects that are raised at the time of software development phases are explained with CMMI framework and its goals. As a result, the aim was to make a better software that is fully satisfying the requirements of customers with the help of SQA system components.

REFERENCES

1. Galin, Daniel (2004), Software Quality Assurance – From theory to implementation, Pearson –Addison Wesley, England.
2. Abdel-Hamid, T.; Madnick, S. E. (1991), Software Project dynamics: an Integrated Approach, Prentice-Hall Software Series, Englewood Cliffs, New Jersey, USA.
3. Ambler, Scott W. (2002), Examining the Agile Cost of Change Curve, Ambysoft Inc.
4. ARENA Rockwell Automation (2007), Forward VisibilityFor Your Business, Balan, S. (2003), A Composite Model for Software Quality Assurance
5. Bass, Len; Paul Clements; Rick Kazman (2003), Software Architecture in Practice, Second Edition. Boston: Addison-Wesley, p. 21-24. ISBN 0-321-15495-9.
6. Bertsekas, Tsitsiklis (1996), Neuro-Dynamic Programming, Athena Scientific.
7. Boehm, B. W. (1981), Software Engineering Economics, Prentice-Hall, NJ, USA.
8. Brennecke, Andreas; Keil-Slawik, Reinhard (1996), Position Papers from Dagstuhl seminar 9635 on History of Software Engineering August 26-30, 1996, Germany.

9. Chikofsky, E.J.; J.H. Cross II (January 1990), Reverse Engineering and Design Recovery: Taxonomy in IEEE Software, IEEE Computer Society: 13–17.
10. Pressman, Roger S. (2000), Software Engineering - A Practitioner's Approach, European Adaptation by D. Ince, 5th Edition, McGraw Hill International, London
11. The Linux Information Project (2004 – 2007), Bellevue Linux, Bellevue, WA, USA.
12. Warden, R. (1992), Software Reuse and Reverse Engineering in Practice. London, England: Chapman & Hall, 283–305.
13. Zave, P. (1997), Classification of Research Efforts in Requirements Engineering, ACM Computing Surveys, 29 (4): 315-321
14. Padberg, Frank (1999), A Probabilistic Model for Software Projects, Proceedings ESEC/FSE 7 109-126, Lecture Notes in Computer Science 1687, Springer 1999.
15. Padberg, Frank (2000), Estimating the Impact of the Programming Language on the Development Time of a Software Project, Proceedings International Software Development and Management Conference ISDM/AP-SEPG 287-2
16. N. Gross; M. Stepanek; O. Port; J. Carey (December, 1999), Software Hell: Glitches cost billions of dollars and jeopardize human lives. How can we kill the bugs? Business Week Online –International, NITS (2006), NIST/SEMATECH e-Handbook of Statistical Methods,
17. Ntourntoufis, Panos (2002), From Software Quality Control to Quality Assurance, UPSPRING Software, UK.
18. Nuseibeh, Bashar; Easterbrook, Steve (2000), Requirements Engineering: A Roadmap, Future of Software Engineering, Limerick Ireland ACM 2000 1-58113-253-0/00/6
19. Warden, R. (1992), Software Reuse and Reverse Engineering in Practice. London, England: Chapman & Hall, 283–305.