Software Management and Quality Assurance

1.1.1 Minakshi Dhillon & Happy Sharma

Department of Information Technology, Dronacharya college of engineering,
Khentawas, Farrukhnagar, Gurgaon-123506,India
Email: minakshidhillon@yahoo.in, happy22sharma@gmail.com

ABSTRACT

SOFTWARE MANAGEMENT AND QUALITY ASSURANCE, this idea basically deals with the Software project management is the art and science of planning and leading software projects.[1] It is a sub-discipline of project management in which software projects are planned, implemented, monitored and controlled and Software Quality Assurance (SQA) involves the entireworking on the software development process - monitoring and improving the process. it make sure that any agreed-upon standards and procedures are followed, and ensuring that problems are found and dealt with. It's aimed towards prevention and if followed will result in the production of quality software. To manage new development efforts, companies applied the established project management methods, but project schedules slipped during test runs, especially when confusion occurred in the gray zone between the user specifications and the delivered software. To be able to avoid these problems, software project management methods focused on matching user requirements to delivered products, in a method known now as the waterfall model.

1. INTRODUCTION

There are billions of dollars being spent trying to develop quality software, software bugs are very common. For most computer systems, the cost of software constitutes a major part of the cost of the system. Since software is so important and valuable, if software development process lacks quality, then the software that's developed will surely lack quality and will not also be user interface. Software Quality Assurance and its management involves the entire software development PROCESS - monitoring and improving the process, making sure that any agreed-upon standards and procedures are followed, and ensuring that problems are found and dealt with and managing the whole art of the development. It is oriented towards prevention". Software Quality Assurance is aimed at developing a sound software development methodology that will produce quality software.

2. INDUSTRIAL HISTORY

In the 1970s and 1980s, the software industry grew very quickly, as computer companies quickly recognized the relatively low cost of software production compared to hardware production and circuitry. As the industry has matured, analysis of software project management failures has shown that the following are the most common causes

1. Unrealistic or unarticulated project goals
2. Inaccurate estimates of needed resources
3. Badly defined system requirements
4. Poor reporting of the project's status
5. Unmanaged risks
6. Poor communication among customers, developers, and users
7. Use of immature technology
8. Inability to handle the project's complexity
9. Sloppy development practices
10. Stakeholder politics
11. Commercial pressures

3.SOFTWARE DEVELOPMENT PROCESS

A software development process is concerned primarily with the production aspect of software development, as opposed to the technical aspect, such as software tools. These processes exist primarily for supporting the management of software development, and are generally skewed toward addressing business concerns. Many software development processes can be run in a similar way to general project management processes. Examples are:

- Risk management is the process of measuring or assessing risk and then developing strategies to manage the risk. In general, the strategies employed include transferring the risk to another party, avoiding the risk, reducing the negative effect of the risk, and accepting some or all of the consequences of a particular risk. Risk management in software project management begins with the business case for starting the project, which includes a cost-benefit analysis as well as a list of fallback options for project failure, called a contingency plan.
- Requirements management is the process of identifying, eliciting, documenting, analyzing, tracing, prioritizing and agreeing on requirements and then controlling change and communicating to relevant stakeholders. New or altered computer system[1] Requirements management, which includes Requirements analysis, is an important part of the software engineering process; whereby business analysts or software developers identify the needs or requirements of a client; having identified these requirements they are then in a position to design a solution.
- Change management is the process of identifying, documenting, analyzing, prioritizing and agreeing on changes to scope (project management) and then controlling changes and communicating to relevant stakeholders. Change impact analysis of new or altered scope, which includes Requirements analysis at the change level, is an important part of the software engineering process; whereby business analysts or software developers identify the altered needs or requirements of a client; having identified these requirements they are then in a position to re-design or modify a solution. Theoretically, each change can impact the timeline and budget of a software project, and therefore by definition must include risk-benefit analysis before approval.
- Software configuration management is the process of identifying, and

documenting the scope itself, which is the software product underway, including all sub-products and changes and enabling communication of these to relevant stakeholders. In general, the processes employed include version control, naming convention (programming), and software archival agreements.

- Release management is the process of identifying, documenting, prioritizing and agreeing on releases of software and then controlling the release schedule and communicating to relevant stakeholders. Most software projects have access to three software environments to which software can be released; Development, Test, and Production. In very large projects, where distributed teams need to integrate their work before releasing to users, there will often be more environments for testing, called unit testing, system testing, or integration testing, before release to User acceptance testing (UAT).

4. SOFTWARE BUGS

Microsoft Chief Executive, Steve Ballmer said that any code of significant scope and power will have bugs in it. And only 1% of bugs in MS Software is causing half of all reported errors. Find and fix 1% of your software bugs, and 90% of your system problems go away, say experts. The term "*Software Crisis*" is used in the software industry to emphasize the complexity in developing quality software.

Miscommunication: There is widespread miscommunication of information during all the phases of software development, because humans tend to assume and misinterpret a lot of things when communicating.

Software Complexity: Any software, that's developed to serve some useful purpose, is enormously complex and no single person can fully understand it.

Programming Errors: Software is created by people, and people are inherently prone to making errors. So, software bugs are also created due to programming errors.

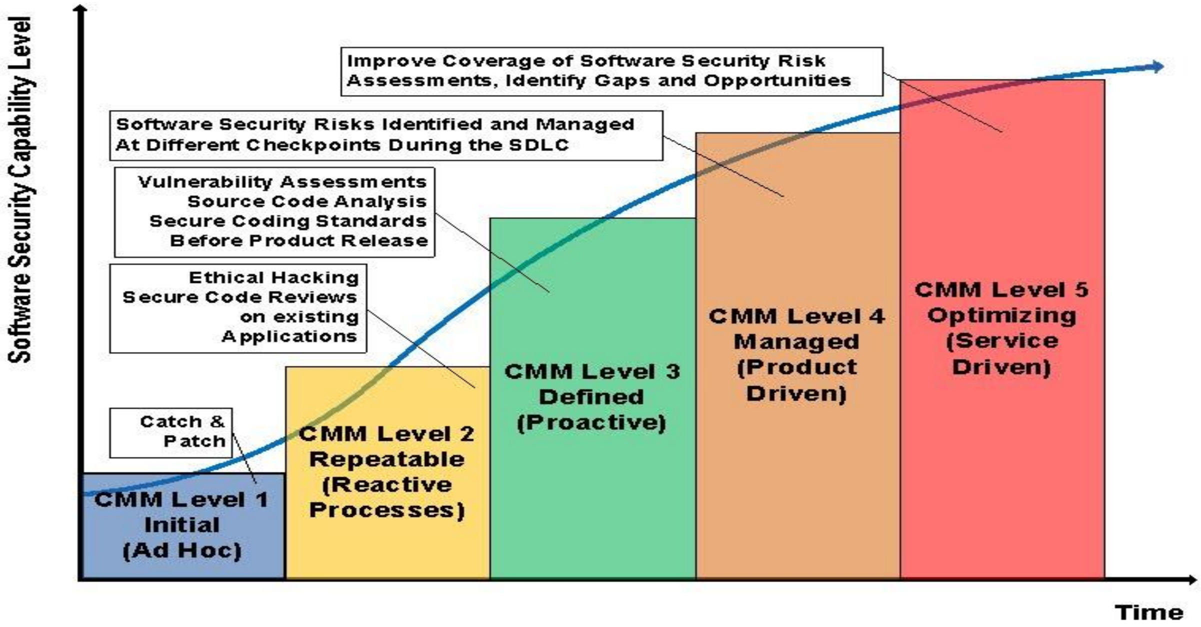
Changing requirements: Software functionality changes, when the requirements change.

Time pressure and deadlines: The software development industry is highly competitive, and schedule slippages are not acceptable. Some projects have unrealistic schedules, which make the development methodology far from perfect and the developed software lacks quality.

5. CAPABILITY MATURITY MODEL(CMM)

The *'Software Engineering Institute'* (SEI) at Carnegie-Mellon University, was initiated by the U.S. Defense Department, to help improve the software development processes. The SEI came up with a model with five levels. These levels are used to gauge the maturity of a software development organization. The CMM model was mainly aimed at making sure that organizations, which bid for contracts with the US Department of Defense (DOD), followed a good process, and developed quality software. Organizations receive CMM rankings, by undergoing assessment by qualified auditors. Any organization, that does a contract for the DOD, must reach at least level 3 in the CMM model .

The five levels quantify the software development methodology, followed by the organization. following subsection will discuss on what ratings at each level mean.


Software Security Capability Level
Time
Improve Coverage of Software Security Risk Assessments, Identify Gaps and Opportunities
Software Security Risks Identified and Managed At Different Checkpoints During the SDLC
Vulnerability Assessments Source Code Analysis Secure Coding Standards Before Product Release
Ethical Hacking Secure Code Reviews on existing Applications
Catch & Patch
CMM Level 1 Initial (Ad Hoc)
CMM Level 2 Repeatable (Reactive Processes)
CMM Level 3 Defined (Proactive)
CMM Level 4 Managed (Product Driven)
CMM Level 5 Optimizing (Service Driven)

CONCLUSION

As a subdiscipline of project management, some regard the management of software development akin to the management of manufacturing, which can be performed by someone with management skills, but no programming skills. Software development is complex, and is error prone. Many problems that are faced during software development can be tackled, by adopting a good software development process. From our discussion, it’s apparent that good processes are essential. The software industry is still learning, about good processes for software development. CMM was developed, to assess, and to give organizations, a framework to improve. Despite some flaws, CMM is a significant contribution to the software industry.

REFERANCES

1. Stellman, Andrew; Greene, Jennifer (2005). Applied Software Project Management. O'Reilly Media. ISBN 978-0-596-00948-9.

2. John C. Reynolds, Some thoughts on teaching programming and programming languages, SIGPLAN Notices, Volume 43, Issue 11, November 2008, p.108:

3. Boehm, B. W. (1981). Software engineering economics.

4. Humphrey, W. S. (1995). A discipline for software engineering.
Pressman, R. S., & Jawadekar, W. S. (1987). Software engineering. *New York 1992*.

5. Schulmeyer, G., & McManus, J. I. (1987). *Handbook of software quality assurance*. Van Nostrand Reinhold Co..

6. Galin, D. (2004). *Software quality assurance: from theory to implementation*. Pearson education.