Software Quality Assurance

Rishabh Baid
Graduate Student
MATS University
School of Information Technology
Email: rishabhbaid96@gmail.com

Abstract

Software Quality Assurance (SQA) 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. It's aimed towards prevention and if followed will result in the production of quality software. This paper emphasizes the importance of a quality process and also discusses about the ways in which it could be achieved.

1 Introduction

Though billions of dollars are 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. *"Software Quality Assurance (SQA) 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. It is oriented towards prevention [1]".* Software Quality Assurance is aimed at developing a sound software development methodology that will produce quality software.

2 Importance of SQA

There is an increasing use of software, in all walks of life. From electronic devices like watches, and cell phones to applications like ecommerce, banking, medical and what not? Computer Systems are omnipresent and all computers run some software. So, software is omnipresent. Due to the widespread acceptance, and use of software systems, in various areas, software bugs are proving to be costly, and sometimes fatal. The *Sustainable Computing Consortium*, a collaboration of major corporate IT users, university researchers and government agencies, estimates that buggy or flawed software cost businesses $175 billion worldwide in 2001 [3]. Interested readers are referred to [1] for a list of some of the recent, major computer system failures, caused by software bugs, and its consequences. Bugs have affected banking systems, stock exchanges, medical institutions, educational

institutions and even the Social Security Administration. Most bugs, encountered during software development, can be avoided, by adopting a sound software development process, and having strict software quality control using Software Quality Assurance. The process of SQA is comparable to *Software Testing*.

3 Software Quality Assurance VS Software Testing

Software Testing involves operating a system, or an application, under controlled conditions, and evaluating the results. In most cases, software testing will involve the development of a test bed, which tests the given software, upon a set of test cases. The test bed will feed the test input to the software system, get the result that's generated by the software system, and compares the generated result with the expected result. If the generated result is same as the expected result, then the software is bug free else, it has bugs that need to be fixed.

Software testing is normally carried out under controlled conditions. The controlled conditions should include both normal and abnormal conditions. The aim of testing is to try to break the software, and find the bugs in it. Successful testing will discover all the bugs in the software. Developing automated test tools to perform testing is an active area of research. Testing is oriented towards *'detection'* of bugs in the software (An interesting article that discusses about how extensive testing should be can be found in [4]). On the other hand, SQA is aimed at avoiding bugs.

Software Quality Assurance is oriented towards *'prevention'* of bugs in the software, by following a software development methodology. SQA is more concerned with developing a quality *process* for software development, which will prevent the generation of bugs, and will result in the production of quality software. SQA, when practiced, makes sure that all the standards are followed, and that all the problems that arise during development are detected and are dealt with. Both SQA and Software testing are non- trivial tasks.

Software Quality Assurance is more challenging than Software Testing because, solving problems is a high-visibility process; preventing problems is a low-visibility process. During Software Testing, we know what the problem is, and we are trying to fix the problem, which is easier than, preventing the problem before it occurred, or even showed signs of occurrence.

Given the importance of software testing and SQA is one is left wondering why is software so error prone. Why do we always have software bugs?

4 Reasons for 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 [2].

Find and fix 1% of your software bugs, and 90% of your system problems go away, say experts [3].

The term "*Software Crisis*" [10] is used in the software industry to emphasize the complexity in developing quality software. There are five common problems in the software development process. They are miscommunication, software complexity, programming errors, changing requirements and unrealistic schedule [1].

- **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 [2].

- **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. When we have a system with rapidly changing requirements, additional functionality that's added to the system, can affect the already existing modules in unforeseen ways. High level of interdependencies between the modules, makes the system error prone.

- **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.

Given these problems, it's apparent that software bugs are very common. One is surely left wondering, "Did any one do anything to reduce software bugs?" and make software more reliable. The answer is "yes". The next section discusses one such successful attempt.

5 Capability Maturity Model (CMM)

The *'Software Engineering Institute'* (SEI) [5] 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 [1].

The five levels quantify the software development methodology, followed by the organization. The

following subsection will discuss on what ratings at each level mean.

5.1 Level 1 - Initial or chaotic

Level 1 means that the software development methodology, followed by an organization is in its novice stage, and is filled with chaos, and periodic panics. Due to lack of any methodology, heroic effort is required by individuals, to successfully complete projects. No software process is in place, and even if the organization meets with success in a project, successes may not be repeatable in other projects. [1]

5.2 Level 2 – Repeatable

Level 2 in the CMM model means that, some software development process is in place, and is being followed. Software project tracking, requirements management, realistic planning, and configuration management are part of the process in place. The success achieved by the organization in a project is repeatable in other projects. [1]

5.3 Level 3 – Defined

Level 3 in the model signifies that standard software development, and maintenance processes are integrated throughout an organization. It also means that, a Software Engineering Process Group is in place, to oversee software processes, and training programs are used to ensure understanding, and compliance. Any organization that does contracts for the US Department of defense, must reach this level. [1]

5.4 Level 4 – Managed

If an organization reaches level 4 in the CMM model, then it means that metrics are used, to track productivity, processes, and products. Project performance is predictable, and quality is consistently high. [1]

5.5 Level 5 – Optimized

At level 5 of the CMM model, the focus is on continuous process improvement. The impact of new processes, and technologies, can be predicted, and effectively implemented when required. Moreover, as and when required, the software development methodology that's practiced is optimized to suit the changing needs. [1]

Organizations which comply with the CMM process (Level 3 and higher); will surely produce quality software, when compared to organizations at lower levels of the model. Software developed by organizations, that have attained level 3, or higher, is less likely to be error prone. Despite its advantages, CMM also has some disadvantages.

CMM describes what an organization should have, does not say how to get there. Also, a clearly defined process is not equal to a good process. For a discussion on the drawbacks of CMM refer [12].

CMM is not the only methodology, that's in place to improve the software development process. There are also other approaches suggested by IEEE, ANSI and the ISO [1]. But the CMM model is the most popular, and is

an industry standard, with wide spread use and acceptance.

6 Conclusion

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. The second version of CMM (CMMv2) is currently in progress at the Software Engineering Institute at the Carnegie Mellon University.

7 References:

1. Rick Hower's "Software QA and Testing Resource Center" (Source: www.software qatest.com)

2. Any software code will have bugs-Microsoft (Source: http://www.ciol.com/content/ news/repts/102100302.asp)

3. Biting Back (Source: http://www.computerworld.com/softwar etopics/software /appdev/story/0,10801,77381,00.html)

4. Finding Your Sweet Spot (Source: http://www.computerworld.com/softwar etopics/ software/story/0, 10801,77374,00.html)

5. Carnegie Mellon Software Engineering Institute (Source: http://www.sei.cmu.edu/)

6. Resources for Busy Testers (Source: http://www.qacity.com/front.htm)

7. Software Testing Hotlist (Source: http://www.io.com/~wazmo/qa/)

8. Storm (Source: http://www.mtsu.edu/~storm/)

9. Internet/Software Quality Hotlist (Source: http://www.soft.com/Institute/HotList/)

10. The Software Crisis (Source: http://www.unt.edu/benchmarks/archives /1999/july99 /crisis.htm)

11. Software Testing and Quality Assurance (Source: http://www.software-quality-assurance.info/)

12. CMM information (source: http://www.cs.concordia.ca/~faculty/paq uet /teaching/342/CMM.ppt)

13. Bca Notes (source : http://bcaraipur.blogspot.in)