Software Quality

CS-C3150 Software Engineering

What is Software Quality?

Quality in simple terms means the ability of a product to meet its required specification. A quality product should meet all the requirements, perform efficiently and reliably, and should be delivered on time within the budget. This definition of "quality" holds in many scenarios, but it becomes very problematic when it comes to software quality. This is because it's very difficult to write down complete and unambiguous software requirements. The specifications come from various stakeholders in which case some of the requirements are compromised and may not fulfill the requirements of all the stakeholders. The quality of software reflects in the user experience. Software quality isn't just about the functional requirements of the software, but it is more about fulfilling the quality requirements because these attributes reflect software usability, dependability, efficiency, and maintainability. (Sommerville, 2016)

Existing Standards

The IEEE defines software quality in two components which are verification and validation. Verification is the degree to which software meets its specified requirements and validation is the degree to which software meets the needs and expectations of the user. The ISO 9126 standards have classified the quality characteristics into six main categories which are Portability, Reliability, Efficiency, Usability, Maintainability, and Functionality. Portability checks whether the software can be transferred to a different environment. Maintainability checks if the software is easy to maintain and the rest of the categories are self-explanatory. (Lassenius, 2013) Both of these standards do give a picture of the considerations of quality software, but it is very important to understand that the definition of software quality is variable and depends a lot on the kind of software being developed and its development process. (Sommerville, 2016)

Achieving Software Quality

Achieving software quality is one of the most important aspects of developing software. To achieve software quality, the first and foremost step is to have a well-defined requirements engineering phase where we try to interact with the customers as much as possible to eliminate the "ambiguous" and "inconsistent" behavior of software requirements. A clear picture of what to develop and how to development automatically helps in building a software that is closer to the needs of the user. (Sommerville, 2016) To improve software quality, it is very important to be very cautious and careful during the development testing phase because the system is most vulnerable to an error during this. To avoid bugs and errors, some development processes use the concept of pair programming. In pair programming environment the overhead of error-management is lower than in traditional environments where the quality assessment and development teams are separate. The team members fix the defects as soon as possible while the code is still fresh in the developer's mind. This has various benefits such as working on a highly stable and clean code makes the development process faster. (Dubinsky et al., 2006) While software testing is an important part of discovering errors, it is essential to have a software inspection. Software inspection helps in surfacing the errors which could be masked during the testing process. It also keeps a check on broader quality attributes and it enables us to look for inefficiencies and other loopholes which could cause a problem during maintenance of the software. Once the complete system has been integrated, it is essential to carry out a performance test to check emergent properties like performance and reliability. This not only demonstrates whether the system meets its requirement but also helps in discovering problems and defects. It is always a good option to test the software in the real environment in which it is meant to work. Environment influences the use of a system in ways the developers cannot include in their testing environment. (Sommerville, 2016)

Software design is a creative process and the skills and experience the developer holds have a significant role in developing quality software. The company needs to make the process standards adaptable because stiff standards can stifle creativity which can lead to poorer software quality. A quality manager should aim to develop a culture in which everyone is committed to developing high-quality software. While standards define the basis of quality management, quality managers should understand that certain intangible aspects cannot be encapsulated by standards. (Sommerville, 2016)

To improve software quality it is important that everyone related to the project has full information on the project status and has a communication channel. It is vital to keep the stakeholders in the loop and not to isolate the developers from the end-users. Isolation can steer the project in another direction and also delay its delivery.

Thus we see that software quality cannot be achieved by some standards or rules. It is a continuous effort of implementing different good techniques appropriate for the project throughout the development cycle.

References

DUBINSKY, Y., HAZZAN, O., KEREN, A., TALBY, D. 2006. *Agile Software Testing in a Large-Scale Project. IEEE Software* [online]. **23**(4), pp. 30-37. [viewed 8 November 2019] Available from: https://mycourses.aalto.fi/pluginfile.php/1088065/mod_resource/content/4/Talby_2005.pdf

LASSENIUS, C., 2013. *Software Quality Assurance and Testing* [lecture online]. 19 February. [viewed 8 November 2019]. Available from: https://mycourses.aalto.fi/pluginfile.php/1088063/mod_resource/content/3/QualityTesting.pdf

SOMMERVILLE, I., 2016. *Software Engineering.* 10th ed. Boston: Pearson