Software engineering has become one of the pillars of the modern world’s economy. The necessity of quality software is essential as a malfunctioning system might cause disastrous consequences. This essay discusses the definition and the methods to achieve 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 is very difﬁcult to write down complete and unambiguous software requirements. Software speciﬁcations come from various stakeholders in which case some of the requirements are compromised and may not fulfill all the requirements set by the stakeholders. The quality of the software is reflected in the user experience. Software quality is not 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).

The Institute of Electrical and Electronics Engineers (IEEE) deﬁnes software quality in two aspects: veriﬁcation and validation. Veriﬁcation is the degree to which software meets its speciﬁed requirements whereas validation concerns whether the software meets the needs and expectations of the users. The ISO 9126 standards have classiﬁed the quality characteristics into six main categories, which are: portability, reliability, efﬁciency, usability, maintainability, and functionality. Portability checks if the software can be transferred to a different environment. Reliability makes the software capable of maintaining its performance in stated conditions. Efficiency balances the performance of software and the resources it consumes. Usability deals with the ‘easiness’ of using the software. Maintainability ensures software is easy to maintain and the rest of the categories are self-explanatory. Functionality keeps the functions within the software secured and accurate (Lassenius, 2013). These standards set a guideline for quality software, but the deﬁnition of software quality remains equivocal and depends on the kind of software being developed and its development process (Sommerville, 2016).

Achieving software quality is one of the most important aspects of development. The ﬁrst and foremost step is to have a well-deﬁned requirements engineering phase where we interact with the customers as much as possible to eliminate the “ambiguous” and “inconsistent” behavior of software requirements. An unobstructed vision of what and how to develop contributes to building software that accommodates the needs of the users (Sommerville, 2016). To improve software quality, developers should be cautious and attentive during the development and testing phase when the system is most vulnerable to malfunctions. The concept of pair programming could potentially reduce the chance of errors much more than in the traditional standalone-development technique, where the quality assessment and development teams are separate. The team members resolve the defects while the code is still fresh. This practice brings various beneﬁts such as shortened development time (Dubinsky et al., 2006). While software testing is an indispensable part of discovering errors, it remains necessary to have software inspection. Software inspection helps in surfacing the errors which could be hidden during the testing process. It also keeps a check on broader quality characteristics, enables efficiency improvement, and fixes subsequent flaws that might interfere with maintenance. Once the complete system is implemented, a performance test in a real environment should be carried out to check emergent properties. This not only guarantees that the system meets its requirement but also helps in repairing random weaknesses. Environment inﬂuences the use of a system in ways the developers cannot imitate in their testing process (Sommerville, 2016).

Software design is a creative process, and the skills and experience the developers possess have a signiﬁcant role in developing quality software. Information technology enterprises need to maintain a flexible approach to the standards mentioned, as strict regulations could have an adverse effect on software quality. A manager should aim to develop a workplace culture in which everyone is committed to developing quality software. While certain concepts define the basis of quality in management, managers should understand that many intangible aspects cannot be encapsulated (Sommerville, 2016).

To improve software quality, everyone related to the project should have a full understanding of the progress and maintains an effective communication channel. It is vital to engage the stakeholders in the development process and not to isolate the developers from the end-users. Furthermore, software quality cannot be achieved by applying standards or rules, but by a continuous effort to implement appropriate techniques 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 6 November 2021]. DOI: 10.1109/MS.2006.93

LASSENIUS, C., 2013. *Software Quality Assurance and Testing* [lecture online]. 19 February. [viewed 6 November 2021]. Available from: <http://docplayer.net/197663329-Software-quality-assurance-and-testing-casper-lassenius.html>

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