Student Name: Gökay Gülsoy
Student Number: 323078045

MISCONCEPTION THAT SOFTWARE TESTING IS A SUPPLEMENTARY TASK

Software testing has long been seen as a supplementary task in the software development process. Is it really less crucial than other tasks in the software development life-cycle ?. The answer is definitely no; falsification of this statement can be supported by three fundamental facts about the software development process and its history. The first one is that the biggest software failures occur in critical tasks and systems due to a lack of even simple testing, which in turn culminate in catastrophic results. The second one is quantifying the time spent on testing relative to when testing is not involved in the software development life-cycle. The third is a lack of the necessary formal education in software testing and verification within the overall software development process. These three facts, when put together, demonstrate that software testing is not just required in the software development process but indeed vital for developing high-quality software systems by reducing the time spent on the overall procedure and providing professionals with software testing maturity via formal education to better understand its importance in the overall life-cycle.

The first fact that falsifies the view that software testing is optional is reflected in the biggest failures that occurred in critical tasks and missions throughout history, which resulted in the loss of millions or billions of dollars from budgets. For instance, NASA's Mars Climate Orbiter (MCO) and Mars Polar Lander (MPL), both of which were sent to broaden their knowledge of the Red Planet by collecting information about Mars' climate, atmosphere, and surface conditions, failed to do so because of critical yet preventable issues in testing [1]. The MCO was lost due to a unit conversion error between metric and imperial measurements, while MPL likely failed because of an early engine shutdown due to an incorrect signal. Financially, the combined losses from both of these missions exceeded $492.6 million, with additional costs in delayed scientific progress. Another example is the Ariane 501 satellite launch failed catastrophically 40 seconds after initiation on 4 June 1996, incurring a direct cost of approximately $370 million [2]. The approximate cause of the disaster was identified as an outside-expected range value, derived from the launcher's horizontal velocity, which led to an unhandled exception in the Internal Reference System (SRI). These events have proven that software testing is even more necessary than performing the task itself, which may have otherwise caused millions of dollars of loss and a great amount of wasted time.

The second fact is thoroughly studied by different researchers, which disproves the fallacy that effort put into software testing is a waste of time. It is mistakenly considered that if testing isn't done, the project will be released much earlier, but various studies have shown that the cost of fixing bugs increases multiplicatively as the software project progresses through its life-cycle. For example, according to Steven McConnell a distinguished software engineering expert cost of fixing defects increases as the software project moves from the requirement analysis phase towards the release phase [3]. For instance defect introduced in the architectural design phase costs 10 times as much to fix if it is detected in the construction phase, 15 times as much if it's detected during system testing. Similarly, Emeritus Professor of Control and Intelligent Systems engineering in the University of Hull, Ron Patton, gives an estimate that the software cost to fix is increased 10 times per project phase [4]. Overall, all these studies have indicated that if testing is not properly conducted, even though a team may have a quick start at the beginning of the software project, they will struggle much more compared to situations that have systematically carried out tests.

The third fact relates fallacy of software testing being less important than other life-cycle activities to not having a sufficient number of well-educated professionals in the software testing field. Educational research reveals that many computer science students struggle to grasp the fundamental software testing concepts, resulting in incomplete or ineffective test suites, which introduce potential weaknesses that propagate through integrated systems [5]. It means that a better understanding of testing principles is required for breaking the notion that testing is not as important as other software life-cycle activities. Recent research underscores the importance of conceptual feedback in software testing education. Studies have shown that providing students with feedback on missing fundamental testing concepts-such as branch coverage, equivalence class partitioning, and exception-handling, leads to greater improvements in test suite quality than simply identifying missing test cases [6, 7]. Studies indicated that as the number of well-educated professionals in the software testing field increases, better understood principles lead to higher quality software systems, which urge professionals to consider software testing as equally important as any other task in the software development life cycle.

These three facts, when considered together, refute the notion that software testing has a lower priority and can be neglected as compared to other phases of the software development life cycle. As the first fact suggests, if software systems are tested properly, loss of enormous budgets can be prevented and used for further benefits. When the second fact is taken into consideration substantial amount of time can be saved and allocated to other tasks by performing systematic testing in the early stages of the software development life cycle to improve productivity in the whole process. Finally, the third fact suggested that providing all the necessary fundamental principles of testing techniques enables software engineering professionals to gain maturity in testing and better understand its place in the overall software development procedure by integrating systematic testing in their workflows, which also implicitly lets them apply the principles of the first and second facts. In conclusion, a good understanding of these three facts eliminates the perception that software testing is a supplementary task and puts it into a position as equally important as all other phases.

REFERENCES

[1] “(PDF) A Case Study on the Mars Climate Orbiter and Mars Polar Lander failures: What is the cost of underestimating testing?” Accessed: Nov. 15, 2025. [Online]. Available: https://www.researchgate.net/publication/391878998_A_Case_Study_on_the_Mars_Climate_Orbiter_and_Mars_Polar_Lander_failures_What_is_the_cost_of_underestimating_testing

[2] M. Dowson, “The ARIANE 5 Software Failure,” *ACM SIGSOFT Software Engineering Notes*, vol. 22, p. 84, 1997.

[3] S. McConnell, “Code Complete, Second Edition | Guide books,” *2004*, Accessed: Nov. 15, 2025. [Online]. Available: https://dl.acm.org/doi/10.5555/1096143

[4] Ron. Patton, “Software Testing,” 2013.

[5] K. Buffardi and S. H. Edwards, “Responses to adaptive feedback for software testing,” *ITICSE 2014 - Proceedings of the 2014 Innovation and Technology in Computer Science Education Conference*, pp. 165–170, 2014, doi: 10.1145/2591708.2591756.

[6] C. Y. Chung and I. H. Hsiao, “Investigating patterns of study persistence on self-assessment platform of programming problem-solving,” *SIGCSE 2020 - Proceedings of the 51st ACM Technical Symposium on Computer Science Education*, pp. 162–168, Feb. 2020, doi: 10.1145/3328778.3366827;JOURNAL:JOURNAL:ACMCONFERENCES;PAGEGROUP:STRING:PUBLICATION.

[7] Z. Fan, S. H. Tan, and A. Roychoudhury, “Concept-Based Automated Grading of CS-1 Programming Assignments,” *ISSTA 2023 - Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 199–210, Jul. 2023, doi: 10.1145/3597926.3598049;ISSUE:ISSUE:DOI.