Optimization of testing processes to accelerate software release

Meiram Ismagambetov Aitmagambetovich
Software Quality Assurance Manager at GNC Holdings LLC, USA
Miami FL, USA.

DOI: 10.31364/SCIRJ/v12.i05.2024.P0524981
http://dx.doi.org/10.31364/SCIRJ/v12.i05.2024.P0524981

Abstract: This article is devoted to the study of optimization of software testing processes to accelerate its release. Taking into account the growing complexity of software products and the need for their rapid adaptation to a variety of technological and user requirements, the purpose of this research is to improve the efficiency and reliability of testing. The paper discusses techniques such as automated testing, the application of graph theory and queuing theory to process modeling, as well as the Monte Carlo method for optimizing resources and time. The article suggests approaches to formalizing test processes that can significantly reduce testing time and improve the quality of the final product. These techniques can be applied in various areas of development, from mobile applications to integrated process control systems.

Keywords: software testing, software, software, program, IT.

Introduction

In modern conditions, the software development industry is one of the most capital-intensive sectors, where any miscalculations in the creation process can lead to significant losses. The complexity of the code, in turn, increases the risk of problems during subsequent modifications and maintenance of the software product. Errors undetected at the testing stage may considerably worsen software reliability and slow down the process of its implementation.

Simple but correct code created by a developer is often subject to revision. Usually, it is related to the use of standard algorithms that do not take into account the specifics of the task and automatically translate program instructions without taking into account the context of their application. Such algorithms do not distinguish the specifics of the problem and do not use the potential advantages of specialized solutions. Using such an approach often leads to results that only partially meet performance and resource-saving requirements.

To create code that efficiently uses available machine resources and commands, sophisticated optimization techniques must be applied. These methods include using optimizing compilers that can significantly improve program performance. In addition, it is important to adhere to the 10/90 principle, which states that 10% of time spent on careful planning before starting work can save up to 90% of time in completing tasks.

The architectural design of a system has a significant impact on its performance. However, the choice of algorithm is even more important to performance than any other aspect of design. More complex algorithms and data structures can handle large amounts of data efficiently. In contrast, simple algorithms are better suited to handle small arrays, as the initialization costs of complex algorithms can exceed the benefits of their use [1].

1. Performance Testing

The performance of applications is a determining factor in their success or failure in the market. Poor performance can lead to user frustration, negative feedback, and loss of revenue. For these reasons, the prioritization of performance testing at every stage of the software development lifecycle is undeniable.

http://dx.doi.org/10.31364/SCIRJ/v12.i05.2024.P0524981

Implementing performance testing has many reasons.

First, it allows for identifying and eliminating bottlenecks at the initial stages of development, which saves time, effort, and financial resources that could be spent on fixing these problems later.

Second, it ensures that software meets established service level agreements (SLAs) and other performance requirements, ensuring that the application achieves the minimum required performance.

Third, performance testing provides valuable insights into the behavior of the software under different operating conditions, allowing it to be optimized and improve overall efficiency [2].

Thus, creating quality software is a multi-step and complex process that requires adherence to certain standards and procedures. Testing is one of the key stages of this process, verifying the functionality and compliance of software with the established requirements. With the development of process automation in various spheres, the importance of automated software testing, including in process control systems, becomes obvious.

Software testing has experienced notable transformations over the last few years, evolving together with changes in development methodologies, technologies, and tools that have radically changed the software industry. Initially, testing was limited to rigorously manual methods but is now characterized by the drive towards automation and the integration of artificial intelligence, heralding a new era in the field.

The formalized representation of the automated testing process through graph theory provides an efficient approach to optimize and automate testing, from the creation of test scenarios to the analysis of results (Figure 1). This allows not only to improve product quality but also to significantly reduce the time required for testing in complex multilevel control systems.

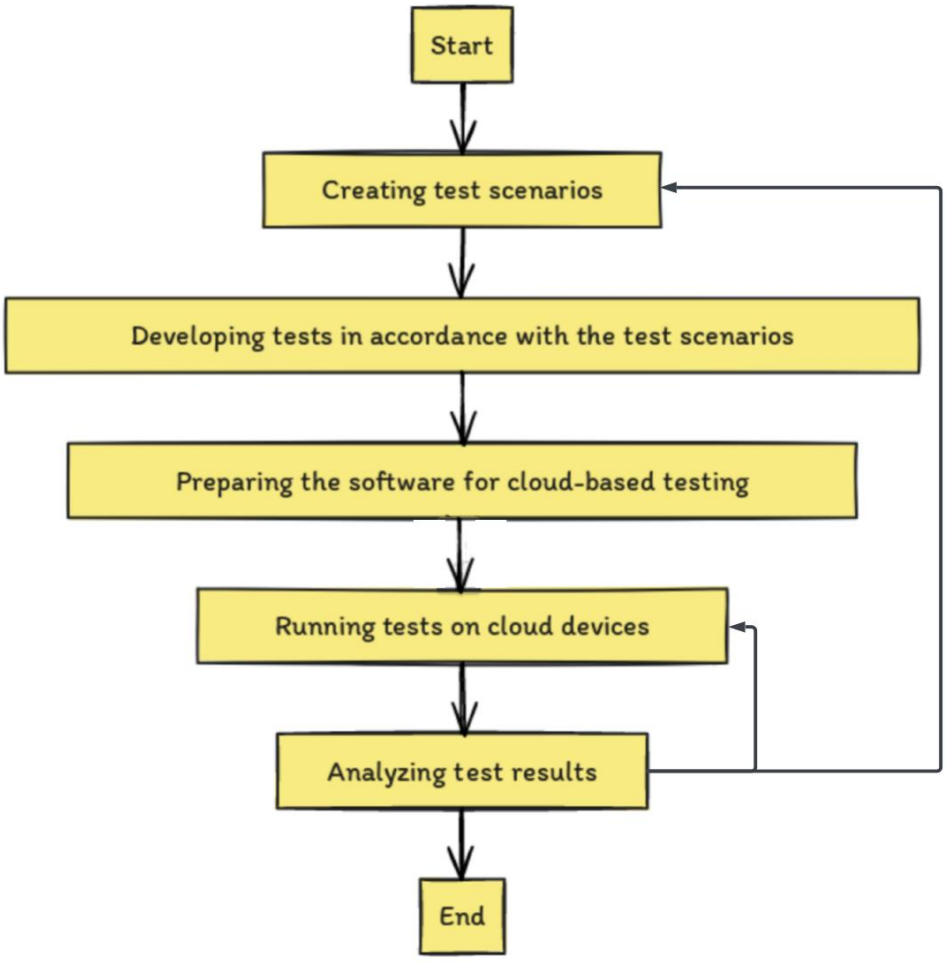

Fig.1. The structure of software testing

http://dx.doi.org/10.31364/SCIRJ/v12.i05.2024.P0524981

Figure 1 shows that the software testing process involves some sequential stages, each of which requires careful planning and implementation.

The initial stage, the creation of test scenarios, involves identifying the functions and aspects of the software to be tested. Each test scenario can span multiple test cases to comprehensively evaluate all possible states of the program. The methods for creating test scenarios can be either formal or informal, depending on the requirements of the project.

The next stage, test development, involves programming the test cases in the chosen programming language. This process covers creating test data, executing the tests on, and capturing the results for later analysis.

The third stage involves preparing the software for testing in the cloud environment. This stage involves uploading the software product to the cloud platform, setting up virtual machines for testing, and configuring the cloud environment to support tests on different devices.

Running the tests in the cloud infrastructure constitutes the fourth stage. This involves executing tests on multiple devices in the cloud, which can take a significant amount of time depending on the size and complexity of the tests.

The fifth and final stage is analyzing the test results. It involves detailed analysis of the collected data, visualizing the results for better understanding, and reporting on the bugs and issues identified.

Graphical representation of the testing process, using graph theory, allows to visualize and optimize each step by displaying the dependencies between the different steps. For better visualization, the data will be presented as a table.

Table 1. Features of software testing [1]

Testing Stage	Service Time (hours)	Queue Waiting Time (hours)
Scenario Creation	2	0
Test Development	4	2
Software Preparation	1	1
Test Execution	8	4
Test Results Analysis	2	2

From the analysis of the table it is possible to see that each step in the testing process is characterized by its execution duration and waiting time. For example, the generation of test scenarios takes two hours and does not require waiting time because it is the initial stage of testing. In turn, the test development phase requires four hours to execute and includes two hours of waiting time in the queue. Applying mass service theory in modeling this process not only allows us to estimate the total time required for testing but also to identify steps that can be optimized to speed things up.

To effectively model the testing process using mass service theory, some parameters must be considered:

- The number of test participants working simultaneously, which affects queue length and waiting time.
- The nature of service time distribution, which can be uniform or variable.
- System performance requirements, including response time and page load speed.

When a test model is developed, it can be used to optimize the process. For example, if modeling results indicate excessively long wait times in the Test Development step, measures can be taken to speed up that step.

The application of mass service theory to represent software testing steps demonstrates its effectiveness in process optimization. Modeling with this theory allows us to accurately determine the duration of testing and identify key points for improvement[3,4].

http://dx.doi.org/10.31364/SCIRJ/v12.i05.2024.P0524981

In turn, modern testing approaches should also be touched upon, which rely on automated testing systems that can execute complex tests with high speed and minimal human intervention. Test automation, supported by AI-based tools, is becoming an integral part of the process, providing opportunities for seamless operations and improved user experience in a rapidly digitalizing world.

Test automation is central to modern software development methodologies, especially in Agile and DevOps. It involves the use of various tools, platforms, and techniques to automate repetitive and time-consuming tasks, which frees testers to perform more complex and meaningful tasks. Table 2, reflects the advantages and disadvantages that are inherent in automated software testing.

Table 2. Advantages and disadvantages of automated software testing

Advantages	Disadvantages
Minimization of human errors through standardization and increased reliability of results.	Automated testing is much more expensive than manual testing.
Acceleration of the testing process and reduction in product release time.	Automated testing also requires additional trained and qualified personnel.
Expansion of testing coverage due to the ability to perform a large number of tests concurrently.	Automated testing eliminates only the mechanical part of the testing process, but test case creation still requires testing specialists.
The ability to reuse test scenarios simplifies regression testing.	
Improved collaboration between different teams through the ability to share automated tests.	

2. Trends

Software testing is continuously transforming, reflecting the impact of technological innovation, evolving methodologies, and changing industry requirements. The following are the key trends that are shaping the future of this field:

- DevOps and Agile methodologies: These approaches are becoming increasingly common in organizations seeking to accelerate development cycles and improve collaboration between development and operations teams. They require integrated and continuous testing, incorporating automation into the development and delivery process.
- IoT testing: With the advent of the Internet of Things, new testing challenges have emerged, including the need to test the connectivity, reliability, and performance of multiple connected devices. IoT testing also includes comprehensive security checks to protect user data and prevent vulnerabilities.
- Performance Engineering: With heightened user expectations for application performance and responsiveness, performance engineering is becoming a key element that surpasses traditional performance testing by incorporating performance considerations into all phases of development.
- Testing in the Cloud: Cloud environments offer scalability, flexibility, and cost-effectiveness, making them ideal for testing. They enable automatic deployment of test environments, parallel execution of tests, and efficient resource management.
- Big Data Analytics for Testing: The application of big data analytics can provide valuable insights into application performance, identify patterns and trends that improve application quality, and enable informed decisions about testing strategies.

http://dx.doi.org/10.31364/SCIRJ/v12.i05.2024.P0524981

These trends emphasize the need for an adaptable, efficient, and scalable approach to software testing, which is combined with innovative tools to enable a more agile and transformational future in testing. [5].

Conclusion

Software testing is subject to continuous evolution, reflecting advances in technology, methodologies, and industry requirements. The study confirmed the significance of optimizing testing processes in the context of accelerating software release. The application of automated testing techniques, including graph and mass theory and Monte Carlo simulation, enables companies to efficiently manage resources, reduce development time, and improve product quality. Optimizing testing not only speeds up the process of getting software to market but also helps improve its competitiveness and reliability. The importance of these methods is emphasized by their applicability in a variety of projects and their ability to adapt to the specific requirements of different types of software. Thus, the integration of the mentioned approaches into standard testing procedures can be considered as a strategic direction for IT industry development.

References

1. Battalova N.I. THEORETICAL FOUNDATIONS OF PROGRAM CODE OPTIMIZATION // Materials of the X International Student Scientific Conference "Student Scientific Forum".[Electronic resource] Access mode: https://scienceforum.ru/2018/article/2018004080 .– (accessed 04/25/2024).

2. Software Performance Optimization: Performance testing strategies. [Electronic resource] – Access mode: https://fastercapital.com/ru/content/%D0%9E%D0%BF%D1%82%D0%B8%D0%BC%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F-%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D0%B8-%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE-%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D1%8F--%D1%81%D1%82%D1%80%D0%B0%D1%82%D0%B5%D0%B3%D0%B8%D0%B8-%D1%82%D0%B5%D1%81%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F-%D0%BF%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%BE%D1%81%D1%82%D0%B8-QAE.html (accessed 04/25/2024).

3. Bukarev A.V. Practical application of the formalized approach to optimize the software testing process in automated process control systems (ACS) // Modern high-tech technologies. 2023. No. 5. pp. 7-12

4. 11 ways to improve the software testing process. [Electronic resource] – Access mode: https://qarocks.ru/software-testing-best-practices/ (accessed 04/25/2024).

5. Kalashnikov E. I. The process of software testing, types and methods of testing //Young Scientist. — 2020. — № 50 (340). Pp. 27-31.

http://dx.doi.org/10.31364/SCIRJ/v12.i05.2024.P0524981