TECHNICAL SCIENCES

THE ROLE OF TEST AUTOMATION IN THE DEVELOPMENT OF BANKING SYSTEMS

Bolgov S.
Specialist degree, Murmansk Arctic University
Murmansk, Russia

РОЛЬ АВТОМАТИЗАЦИИ ТЕСТИРОВАНИЯ В РАЗРАБОТКЕ БАНКОВСКИХ СИСТЕМ

Болгов С.Н.
Специалист, Мурманский Арктический Университет
Мурманск, Россия
https://doi.org/10.5281/zenodo.15657632

Abstract

The article discusses the importance and advantages of test automation in the development of banking software products. Test automation significantly accelerates development processes, increases the accuracy of functionality and security testing, and minimizes human error. The work analyzes modern automation tools and methodologies, including Selenium, JMeter, TestNG, and others, as well as their application in the banking sector. Attention is given to security, reliability, and compliance with international standards such as PCI DSS and ISO 27001. The article also highlights the role of automation within DevOps and Agile methodologies.

Аннотация

В статье рассматриваются значение и преимущества автоматизации тестирования в разработке банковских программных продуктов. Автоматизация тестирования значительно ускоряет процессы разработки, повышает точность проверки функциональности и безопасности, минимизируя влияние человеческого фактора. В работе проанализированы современные инструменты и методологии автоматизации, включая Selenium, JMeter, TestNG и другие, а также их применение в банковском секторе. Уделяется внимание вопросам безопасности, надежности и соответствия международным стандартам, таким как PCI DSS и ISO 27001. В статье также освещается роль автоматизации в рамках методологий DevOps и Agile.

Keywords: test automation, banking systems, testing tools, security, DevOps, Agile.

Ключевые слова: автоматизация тестирования, банковские системы, инструменты тестирования, безопасность, DevOps, Agile.

Introduction

Modern banking systems are complex systems, software, and hardware that provide great many financial services. In this regard, in conditions of digital transformation of the banking sector, not only the creation of new solutions but also assurance of their reliability, security, and conformation to regulatory requirements is becoming very important. Bugs in bank software can result in serious financial losses, reputational risks, and threats to data security.

One of the most promising quality assurance tools is automated testing of banking software. It allows not only to speed up the process of identifying errors but also to increase the accuracy of testing, minimizing the influence of the human factor. Modern technologies of automated testing are widely used in the banking sector, providing integration with DevOps (Development Operations) and CI/CD (Continuous Integration, Continuous Delivery) processes. On the other hand, automation does bring some challenges in implementation: high costs of initial implementation, complexity in supporting test scenarios, and a dire need for adaptation in conditions of change within business logics of banking products.

The goal of this study is to examine how automated testing impacts the quality of banking systems. It explores the theoretical foundations of test automation, the unique characteristics of banking software, and the modern tools and methodologies used to ensure the reliability and efficiency of financial applications.

Main part. Theoretical aspects of test automation

Automated testing plays a crucial role in software quality assurance, particularly in highly sensitive fields like banking. It involves using specialized tools to conduct tests without requiring direct human intervention. Unlike in manual testing, automation improves testing efficiency and accuracy during software function testing, reducing the impact of human error and overall reliability.

The basic concept of test automation is to develop some pre-coded test scripts which would run automatically whenever there is a modification in the source code. It is especially relevant to modern software development processes, for instance, **DevOps and CI/CD** technologies, where the updates are deployed at high frequency.

The most significant benefits are increased testing accuracy, reduction of time costs, and the ability to run tests multiple times with no loss in productivity. Industry research indicates that the introduction of automated

testing cuts testing costs by as high and increases its efficiency as high [1]. Besides, automation works much more effectively with processes that require repetition, like regression testing, when, after changing code, there is often a need to recheck already working functions.

However, test automation comes with its own set of challenges that must be addressed. First, creation and maintenance of test automation **require considerable resources,** including skilled professionals and modern testing tools. Second, **not all tests can be automated-for instance**, complex user experience scenarios normally require manual verification. Additionally, automated tests **need regular updates** because even minor changes in the interface or business logic make tests irrelevant.

Automation of the strategy of testing in the banking sector can be effectively performed using modern tools like Selenium, JMeter, TestNG, Appium, and Postman to automate various aspects of the testing of banking applications. The choice of a particular tool depends on certain testing objectives, application architecture, and quality requirements. For example, in general, Selenium automates web applications testing; JMeter provides assessment of banking systems performance and stability under the load (table 1).

Table 1

Automated testing tools in the banking sector

Tool	Type of testing	Description	Application in the banking sector
Selenium	Functional testing (UI)	A framework for automated testing of web applications, supporting multiple programming languages.	Testing online banking, client web interfaces, and internal banking systems.
JMeter	Load and stress testing	A tool for performance testing of web applications and API (Application Programming Interface).	Assessing the scalability and resilience of banking services under high loads.
TestNG	Regression and integration testing	A Java-based testing framework with flexible test scenario configurations.	Regular validation of updates in banking systems, testing module interactions.
Appium	Mobile testing	A tool for testing mobile applications on Android and iOS.	Testing mobile banking applications, verifying UI and functionality.
Postman	API testing	A tool for testing REST and SOAP APIs, enabling request automation.	Validating the integration of banking systems with external services (payment gateways, government services).

By implementing test automation, banks can significantly enhance the quality of their software. Automation decreases the number of potential bugs, accelerates testing, and maintains systems at optimal levels of security and reliability. For automation to succeed, however, each step has to be thought out carefully, and the correct tools have to be selected, as well as regularly updated test scenarios in order to remain compatible with evolving market requirements.

Features of banking software

Modern banking software is a complex ecosystem that merges services responsible for financial transactions, customer data management, and adherence to regulatory requirements. Unlike other corporate systems, banks have higher demands on security, reliability, and performance due to the possible financial losses and reputational risks caused by failures [2].

The key feature of banking software is a **multilayer architecture** that includes a user interface, server part, databases, and integration with external payment systems. Such an approach gives much more flexibility and ability for scaling but complicates testing. The introduction of containerization and microservice architecture allows updating components independently but requires effective methods for testing interactions between services [3].

Another unique feature of banking software is **high load and fault tolerance**. Systems should work 24/7, process thousands of transactions per second, and survive peak loads-for example, during periods of mass payments. Load testing helps to simulate such scenarios and identify system bottlenecks [4].

One of the key aspects of modern banking applications is security, which requires strict compliance with international standards such as **PCI DSS, ISO 27001, and CCPA** (California Consumer Privacy Act). These regulations mandate rigorous security testing throughout the development and operational stages to protect sensitive financial data [5]. Failure to comply with these requirements can lead to serious sanctions, e.g., fines or payment processing suspension. In order to achieve compliance and minimize risk, banking systems undergo automated security testing, vulnerability scanning, and simulated cyber-attacks, enabling potential threats to be identified early and the overall system's robustness to be enhanced.

Automated testing methods and tools

Automated banking system testing requires a comprehensive approach with various methodologies and specialized software tools. The selection of the test methodology is determined by the objectives, system architecture, and business process peculiarities. Unlike many other areas, banking software testing must encompass not only functionality testing but also security, load tolerance, and compliance with regulatory requirements. With these expectations in mind, the utilization of efficient testing tools is indispensable for the provision of reliability and optimization of development flows.

Vite, Storybook, and Vitest are among the tools that can significantly enhance and accelerate the development and testing of banking applications. Vite is a fast build tool that improves frontend development efficiency by offering instant reloading and optimized module bundling, reducing build time for large-scale applications. Storybook enables developers to build and test UI components in isolation, allowing for faster verification and higher-quality user interfaces without the need for full application deployment. Vitest, a testing framework optimized for Vite, facilitates efficient test writing and execution, seamlessly integrating into CI/CD pipelines to ensure continuous and automated testing processes [6].

Because banking systems can be so complex, it is just as important to select the proper testing practice as it is to employ the correct tools. **Behavior-Driven Development (BDD)** is one of the most utilized practices, meaning test cases are written in a business-friendly way. This approach facilitates testing integration during early development cycles, especially in the case of banking systems since business logic mistakes in such systems can lead to tremendous financial losses. A great example of such a tool is **Cucumber**, where you are able to specify the tests in natural language and provide full transparency of interactions between analysts, developers, and testers.

Regression testing, aimed at preventing errors after changes to the code, occupies a key place in testing banking systems. This approach is especially applicable to banking software that is regularly updated because of the modification of legislative demands, introduction of new functionality, or optimization of current processes. Automated regression testing can be performed with the help of Selenium and TestNG tools that allow you to create repeatable test cases and incorporate them into CI/CD.

Also, **load and stress testing** is performed in order to make the banking system stable under heavy load. This kind of testing allows us to test how well the system will withstand peak loads, such as during large transactions. JMeter is one of the most widely used load testing tools, which has the capability to simulate thousands of users at a time and assist in analyzing various performance metrics. This is even more important in the banking sector, where there is a need to test online banking system, payment gateway, and processing center. Even small lag times in handling transactions can result in catastrophic system failure and monetary losses, making correct load testing a key part of financial application quality assurance.

As mentioned earlier, **security testing** plays a crucial role in the banking sector, as these systems handle confidential information and are constantly exposed to cyber threats [7]. In this regard, tools such as OWASP ZAP and Burp Suite are being used intensively to find web application vulnerabilities, test the security of API (Application Programming Interface), and examine authentication mechanisms for strength. These tools have particular significance in testing PCI DSS and ISO 27001 compliant systems since regulatory requirements necessitate the performance of regular security audits and penetration testing as a way to ensure data security and system integrity.

In such a way, automatic testing methods and tools of the banking sector provide a broad area of application: from functional tests to performance and security analyses. Their usage enables not only enhancement of software quality but also observance of regulations, decrease of financial risks, and strengthening of user confidence in banking systems [8].

Test automation in the context of DevOps and Agile

New banking software development requires high deployment rate and flexibility, and thus traditional testing loses its effectiveness. Under these conditions, test automation becomes a key part of Agile and DevOps practices, allowing for constant monitoring of software quality and releasing new versions in a shorter time span.

Agile methodology is all about short cycles of development that repeat iteratively, with fresh functional changes being input into the product on a regular basis. Manual testing is a bottleneck in such an environment because it takes a long time. Automated testing, however, allows you to test changes virtually immediately, which is especially important with frequent releases. With methodologies such as **Test-Driven Development (TDD)** and BDD, developers can actually write the tests first before writing the core code, which greatly reduces the risk of defects during the early developmental stages (fig. 1).

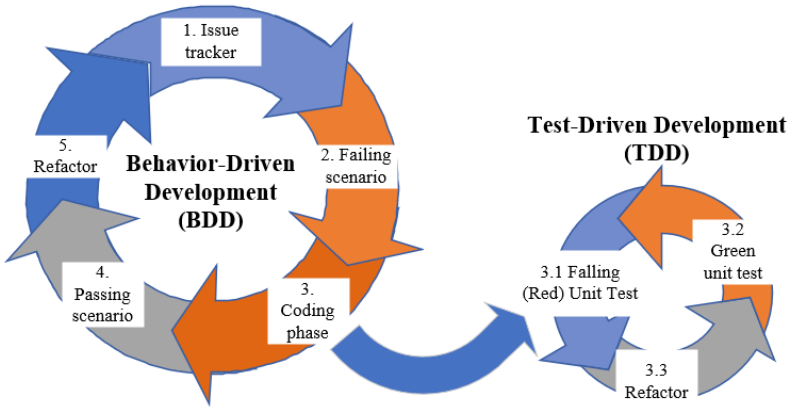

Figure 1. The BDD and TDD process structure

The TDD methodology is based on a three-step cycle. First, a test is written that will initially fail (Red). Then, the minimum code is created that allows the test to pass (Green). After that, the code is optimized: duplications are removed and the structure is improved without changing the functionality (Refactoring). This process is repeated, ensuring the reliability and testability of the code [9].

Automated testing is integrated into CI/CD pipelines in **DevOps**, thus enabling testing of every new code commit in real time. This is achieved through tools like Jenkins, GitLab CI/CD, and Azure DevOps, which perform automating the execution of tests on every code change to check for system stability. For instance, in huge financial organizations, testing becomes an obligatory process before deploying updates to prevent failure and errors in deploying new features.

Therefore, test automation in Agile and DevOps is an integral component of effective bank software development. It helps speed up the development cycle, enhance reliability for software solutions, and decrease risks when releasing new versions of banking systems.

Conclusion

The research done shows that automated testing is an essential element of banking software quality assurance. During high-load environments, high security needs, and constant updates, the use of automated tests increases the process efficiency and reduces the error risks in the code. The scalability and flexibility of testing using the help of modern tools, such as Selenium, JMeter, TestNG, and others, allow you to effectively cooperate with different levels and dimensions of testing: performance, security, and functionality. Automated testing implementation in DevOps and Agile methodologies reduces time costs and improves the quality of the product. Yet, successful automation requires careful planning, choice of the appropriate tool, and updating test scenarios, which remains topical for banking organizations in view of constant changes within legislative and technical requirements.

References:

1. Kumar M. The Design and Implementation of Automated Deployment Pipelines for Amazon Web Services: GitOps practices in the context of CI/CD pipelines using GitLab and Infrastructure as Code. 2024.

2. Zharmagambetov Y. Interest rate risk management and its impact on investment portfolios: the effect of interest rate fluctuations on financial assets and hedging instruments // Financial Markets and Banks. 2024. №. 12. P. 570-574.

3. Kuznetcov I.A. Optimization of distributed systems for mobile applications: improving performance and scalability // Innovative Science. 2024. №. 5-1. P. 52-57.

4. Tonoy M. Mitigating E-Commerce Security Risks with Blockchain: A Multi-Layered Architecture. // 2024 International Conference on Intelligent Systems for Cybersecurity (ISCS), 2024. IEEE. P. 1-6. DOI: 10.1109/ISCS61804.2024.10581253

5. Solomon G. Optimizing Full-Stack Development for Fintech Applications: Balancing User Experience and Backend Performance in High-Stakes Financial Environments. 2024.

6. Dudak A.A. Optimize development and testing processes with vite, storybook and vitest // Innovacionnaya nauka. 2024. №. 9-1. P. 21-25.

7. Balgimbayev A. The evolution of digital payments in the global economy: the role of fintech startups and national initiatives // Cold Science. 2024. №. 12. P. 38-47.

8. Zakharov A.D., Grepan V.N., Blagova I.Y., Ponomarev E.V., Financial technologies as a tool for managing the population's debt burden // First Economic Journal. 2024. № 9(351). P. 87-96. DOI: 10.58551/20728115_2024_9_87

9. Mafi Z., Mirian-Hosseinabadi S.H. Regression test selection in test-driven development // Automated Software Engineering. 2024. Vol. 31. №. 1. P. 9.