International Journal of Computer (IJC)

ISSN 2307-4523 (Print & Online)

https://ijcjournal.org/index.php/InternationalJournalOfComputer/index

Methods for Reducing Testing Costs through Automation

Igor Volynets*

Senior QA Automation Engineer. Usetech Bel, Minsk, Belarus

Email: ivolynets@usetech.ru

Abstract

Software testing automation is a strategically important method that significantly reduces the cost of quality assurance and increases the efficiency of the development process. The article discusses the main methods and approaches that contribute to the optimization of testing, including the choice of tools and technologies, cost-effectiveness assessment and integration of automated processes. The main attention is paid to the issues of planning, selection of tools, as well as structuring and parameterization of tests. Examples of successful automation implementation are given, demonstrating a significant reduction in testing time and improvement of product quality. An assessment of economic efficiency has shown that automation not only pays back the invested funds, but also brings significant profits, especially in large-scale projects. In conclusion, the importance of a flexible approach to the selection of tools and methodologies that allow automation to be adapted to the specific needs and objectives of the project is emphasized.

Keywords: test automation; cost reduction; choice of tools; economic efficiency; test planning; optimization of test processes; regression testing; ROI automation.

1. Introduction

In the context of rapid information technology development and increasing demands for software quality, testing has become an integral part of the development process. In this regard, test automation serves as an effective tool for optimizing costs and enhancing the efficiency of testing processes. The relevance of this topic is determined not only by the potential to reduce costs but also by the ability to ensure higher software quality. Modern software products are becoming increasingly complex and multifunctional, necessitating a large volume of tests, especially during regression testing and compatibility checks. Manual testing in such conditions becomes labor-intensive and prone to errors, while automation allows tests to be performed faster, more accurately, and with minimal human involvement.

Received: 8/15/2024
Accepted: 10/12/2024
Published: 10/28/2024

----

* Corresponding author.

Furthermore, in the highly competitive software market, companies strive to reduce time-to-market, requiring acceleration across all development stages, including testing. Test automation plays a key role in addressing this challenge by enabling quick responses to code changes and promptly identifying defects.

The literature on "Methods for reducing testing costs through automation" highlights several research directions, each contributing to the understanding of automation's potential in optimizing testing expenses. Studies focusing on communication models and practical recommendations for initiating automated testing provide a foundation for developing cost-reduction strategies in global projects. For instance, the study by Rauf M. A. and his colleagues [1] proposes an effective communication model for identifying requirements, which minimizes costs by improving coordination among development team members. Similarly, the practical guide "How do I start automated testing from scratch?" [2], published on the GeeksforGeeks website, presents a step-by-step approach to implementing automated testing, emphasizing the importance of initial process setup and the use of modern tools.

Another significant research area involves systematic reviews and comparative analyses of existing automated testing tools. Reviews by Gadwal A. S., Prasad L. [5] and Viklund K. and his colleagues [8] compile insights into a wide range of tools, highlighting both their advantages and the challenges organizations face in their implementation. Complementing this direction, the study by Okezi F., Odun-Ayo I., Bogle S. [6] evaluates various tools and provides a comparative analysis, offering an objective assessment of their contribution to reducing testing costs.

Empirical research on methods and frameworks for automated testing, as presented in the works of Umar M. A., Zhanfang S. [4] and Sneha K., Malle G. M. [7], aims to determine the practical effectiveness of different automation approaches. These studies focus on selecting optimal tools and methodological solutions that reduce both time and financial costs in the testing process. Meanwhile, Eckhart M. and his colleagues [9] examine the role of automated testing in industrial automation, emphasizing the need for integrating automated systems into complex production environments to maximize efficiency.

A particularly interesting area is the application of innovative algorithmic methods to optimize testing processes. The study by L. M. and his colleagues [3], despite being centered on adaptive ROI determination in the context of remote photoplethysmography, illustrates the potential of block partitioning algorithms in enhancing the efficiency of automated processes. This interdisciplinary approach suggests the feasibility of adapting methodologies from related fields to software testing, further reducing costs.

Thus, a literature review reveals both converging and conflicting approaches to reducing testing costs through automation. On one hand, studies focused on communication models and practical guidelines highlight the potential for significant cost reductions through the adoption of modern automated solutions. On the other hand, systematic reviews and empirical studies point to challenges related to tool integration and the adaptation of methodologies to specific development environments. Furthermore, while modern algorithmic methods show promise, their impact on overall economic efficiency in testing remains insufficiently explored, necessitating further in-depth research.

The aim of this work is to examine methods for reducing testing costs through automation and to analyze the effectiveness of various tools and approaches to automating testing processes.

The limitations of research on cost reduction through automation lie in the need for continuous updates to automated tests due to changes in technology and software architecture, the substantial initial investment required, and the fact that automation is not suitable for certain types of testing, such as usability testing or multimedia file testing.

2. Planning and Cost Estimation for Automation

Software testing plays a key role in ensuring product quality and reliability, but it can be a costly and complex process. For successful testing, it is crucial to plan thoroughly and consider the factors influencing its cost. It is important to clearly define the testing goals and objectives, focusing on the key aspects of the product that require the most attention. This helps to concentrate efforts on the most significant and critical areas, minimizing redundant actions. Additionally, careful planning and development of testing processes, including selecting the most appropriate methodologies and tools, are essential.

One of the most effective ways to reduce testing costs is through the implementation of automation. Automation significantly accelerates the testing process, reduces the likelihood of errors, and increases test accuracy. It also contributes to product quality improvement by providing more thorough checks of functionality and performance. Automated testing is particularly valuable for repetitive tasks, such as regression testing, allowing the team to focus on more complex and creative aspects of software validation [2].

The automated testing process can be divided into several key stages:

- Test Analysis: This stage involves knowledge transfer between business users and analysts, resulting in high-level test cases. It helps identify the main processes that require automation and determine their key characteristics.

- Test Design: In this phase, the scenarios for automated testing are designed, including the definition of essential functions and libraries needed for test implementation.

- Test Development: During this stage, automated tests are created and debugged, making them ready for execution.

- Test Execution and Monitoring: The final stage involves running the tests and monitoring their execution. Tests are grouped according to specified criteria to achieve maximum testing efficiency [2].

To calculate the profitability of automation, the following formulas are applied:

1) The formula for calculating ROIt is as follows:

$$ROIt = S / I$$

where:

- S — time saved through automated tests compared to manual tests (in minutes),

- I — total time spent on development, execution, and maintenance of automation (in minutes).

2) To calculate time savings:

$$S = (V_M - V_A) \times T_N \times T_R$$

where:

- V_M — time taken for manual execution of a single test,

- V_A — time taken for automated execution of a single test,

- T_N — number of tests,

- T_R — number of test runs during the measurement period.

3) To calculate the time spent on investment:

$$I = D_P + (D_A \times T_N) + M_C$$

where:

- D_P — time spent on developing the automation platform,

- D_A — time spent on developing a single automated test,

- M_C — maintenance costs, including time spent on fixing errors and handling failures.

4) The ROIm metric reflects the financial efficiency of test automation and is calculated using the following formula:

$$ROI_m = \frac{T \times W_M - I}{W_A - C}$$

where:

- T — time spent on manual testing during the measurement period,

- W_M — labor cost of a manual tester,

- W_A — labor cost of an automation tester,

- C — other expenses, including licenses, equipment, and server rental [3].

Below, Table 1 presents the interpretation of ROI values.

Table 1: Interpretation of ROI Calculation Results

ROIt	**Result**	**Explanation**
< 1	Poor	The invested costs do not pay off.
1	Good	The invested costs break even. Test automation is beneficial.
> 1	Excellent	The invested costs pay off well. Test automation is highly profitable.

The economic benefits of implementing automation include a reduction in the time specialists spend on routine tasks and a decrease in manual testing costs. By enabling timely defect detection and improving software quality, automation helps optimize software development and maintenance expenses.

Additionally, automated tests provide high accuracy and repeatability of results. Unlike human factors, automated systems eliminate errors related to inattention or fatigue, enhancing the reliability and validity of testing.

However, despite numerous advantages, test automation has its limitations. One of the primary challenges is the need for ongoing maintenance of test relevance, which requires additional resources. Companies often face the task of updating tests when technologies or software structures change.

It is also necessary to consider the financial aspect: implementing automation involves significant investments, including costs for specialists, software, and infrastructure. Automated tests are limited by the constraints of programmed code and may not detect errors outside predefined scenarios, potentially limiting their effectiveness.

Moreover, there are types of testing that are difficult to automate, such as usability testing, working with multimedia files, or installation testing. In such cases, manual testing may prove to be a more suitable and effective approach [4].

Despite the obvious advantages of automation in reducing time and labor costs, the material highlights its limitations, which are caused by both technical and organizational factors. Automated tests, despite their high accuracy and repeatability, require constant updates and adjustments when software products or technological environments change, which can significantly increase operational costs. Furthermore, there are areas of testing, such as usability assessment, working with multimedia files, or installation testing, where automation encounters methodological difficulties and often falls short compared to manual approaches. Thus, a comprehensive analysis of the presented materials reveals that the successful implementation of test automation requires a balanced approach, where economic efficiency is combined with flexibility and adaptability to changes in the technological landscape.

3. Selection of Tools and Technologies for Test Automation

It is important to begin by examining the existing testing technologies. Unit Testing is a method of testing where individual modules or components of an application are verified. This approach helps detect errors at the early stages of development, ensuring the isolation of the tested parts of the code. The advantages of unit testing include the rapid detection and correction of errors, which reduces the cost of fixing defects in later stages of development. As a result, developers can create more stable and high-quality applications, where each module is tested separately and independently.

Integration Testing focuses on verifying the interaction between different modules of an application. This approach helps identify and resolve issues related to the integration and joint operation of various components. Integration testing ensures the correct data flow between modules and their proper interaction, which is especially important in complex applications with numerous interdependent components [5].

System Testing involves checking the operation of the application as a whole. This approach ensures that all components of the application function correctly and meet the established requirements. System testing covers aspects such as functionality, performance, and security. It helps confirm that the product operates correctly and meets user expectations.

Acceptance Testing is conducted from the perspective of the end user and aims to confirm that the application meets the stated requirements and expectations. Typically, acceptance testing is performed at the final stages of development and includes verification of all functional and non-functional requirements, ensuring the product is ready for release [6].

Below, Tables 2-3 provide a comparison of tools based on key aspects of software test automation.

Table 2: Description of the Tools' Capabilities [7]

Capabilities	**Katalon Studio**	**Selenium**	**UFT**	**TestComplete**
Operating System	Cross-platform	Cross-platform	Windows	Windows
Types of Applications Tested	Web, mobile apps, API/Web services	Web applications	Windows desktop, Web, mobile apps, API/Web services	Windows desktop, Web, mobile apps, API/Web services
Supported Programming Languages	Java/Groovy	Java, C#, Perl, Python, JavaScript, Ruby, PHP	VBScript	JavaScript, Python, VBScript, JScript, Delphi, C++, C#
User Programming Skill Level	Not required; recommended for advanced test scripting	High; skills needed for tool integration	Not required; recommended for advanced test scripting	Not required; recommended for advanced test scripting
Tool Learning Complexity	Medium	High	Medium	Medium
Ease of Installation and Use	Easy to install and launch	Requires setup and integration of various tools	Easy to install and launch	Easy to install and launch
Test Script Creation Speed	High	Low	High	High
Object Storage and Maintenance	Built-in repository, XPath, object property modification for identification	XPath, UI Maps	Built-in repository, smart object identification and correction	Built-in repository, smart identification of common objects
Image-Based Testing	Built-in feature	Requires additional libraries	Built-in, object recognition via images	Built-in feature
Integration with DevOps/ALM Tools	Extensive integration	None (requires additional libraries)	Extensive integration	Extensive integration
Continuous Integration	Popular CI tools, e.g., Jenkins, TeamCity	Various CI tools, e.g., Jenkins, Cruise Control	Various CI tools, e.g., Jenkins, HP Quality Center	Various CI tools, e.g., Jenkins, HP Quality Center
Test Results Analysis	Katalon Analytics	None	None	None
Support	Community, business support, specialized personnel	Open-source community	Community, specialized personnel	Community, specialized personnel
License Type	Free software	Open-source (Apache 2.0)	Proprietary	Proprietary
Cost	Free	Free	License fee and maintenance	License fee and maintenance

The comparative table above highlights the key features of test automation tools.

Below, Table 3 provides a comparison of tools from the perspective of their key strengths and weaknesses.

Table 3: Comparison of Tools Through the Prism of Strengths and Weaknesses [7]

Tool	**Strengths**	**Weaknesses**
Katalon Studio	No license or maintenance fee (optional paid support with qualified personnel available)	New solution with a relatively small but rapidly growing community
	Integrates with essential frameworks and features for quick test script creation and execution	Evolving library of methods and toolset
	Built on the Selenium framework but does not require advanced skills, unlike Selenium	Limited choice of programming languages for scripting: only supports Java/Groovy
Selenium	Open-source, no licensing or maintenance fee	High programming skills required for the testing team, along with experience in setup and integration
	Large user and developer community promotes rapid tool development	Significant time investment needed to set up and integrate for automation from scratch
	Integrates well with many tools and platforms for extended capabilities	Slow response from community support
UFT	Comprehensive and well-thought-out automated testing functionality integrated into a single system	High licensing and maintenance fees
	Professional support and a large user community	Potential additional costs for upgrades and extra modules
	Only basic programming skills required to start	Supports only VBScript
TestComplete	Comprehensive automated testing functionality integrated into a single system	High licensing and maintenance fees
	Supports multiple programming languages	Additional fees for extra modules and extensions
	Only basic programming skills required to start	

Thus, it can be concluded that there is no universal tool for test automation that suits every case. The testing team must carefully evaluate available solutions to select the one that best fits their specific needs. As programming languages and technologies used in software development continuously evolve, test automation tools are also subject to changes. Consequently, the cost of the tool becomes a critical factor in the selection process. Commercial solutions often require fees for updates, which can become a significant barrier, especially if the software relies on rapidly changing technologies.

Open-source tools, on the other hand, do not involve financial costs for purchase or updates, but their implementation and maintenance may demand substantial effort and a high level of expertise. Updating such tools is often challenging, and finding the necessary information and knowledge for integrating various platforms and tools can be a complex task.

In this context, new tools like Katalon Studio, which offer integration with open-source frameworks, present promising alternatives to both commercial and traditional open-source test automation solutions. These tools can provide an optimal balance of functionality, cost, and ease of integration into existing infrastructure [7].

The following are the author's recommendations for selecting tools and technologies for test automation, among which the following can be highlighted:

- The choice of tools should be viewed as part of the overall quality assurance strategy. This includes not only technical aspects but also organizational management, communication between teams, and project development strategy.
- It is important to use pilot projects to test hypotheses and empirically validate the chosen solutions. This approach helps to identify hidden issues and optimize the automation process before scaling it.
- The selected tools should be flexible enough to integrate into the existing ecosystem and capable of evolving alongside changes in project requirements.
- The implementation of new technologies should be accompanied by the development of training programs and the documentation of processes, which significantly reduces the learning curve and increases team efficiency.
- The technological environment is dynamic, and regular reevaluation of selected tools is necessary to identify new opportunities, updates, and changes in industry standards.

4. Implementation and Optimization of Automation Processes

The implementation of full-scale automated testing for an application is a complex task. It begins with assessing the feasibility of testing various levels, such as API, user interface, or a combination of both. Subsequently, it is necessary to organize test data and select the optimal set of tools [8].

Consider the example of automating a project for a client specializing in antivirus solutions. The client approached an organization with the aim of optimizing quality assurance processes through test automation. A dedicated team of specialists was assembled to achieve the following objectives:

- Reducing the time required for the deployment and execution of smoke tests;

- Decreasing the number of hours spent on manual testing;

- Ensuring the proper functioning of the client portal in 31 languages.

These goals focused on supporting the development of a new web portal, through which users could install antivirus software on various devices and manage licenses remotely. Since the portal was intended to become a key product for the company, a decision was made to make significant investments in test automation.

Initially, the task was to automate labor-intensive regression testing. Upon completing this stage, the team moved on to developing tests for the most critical features of the portal. The next step involved automating the monitoring of third-party services, whose performance directly impacted the portal's functionality.

The specialists created automated tests to monitor the performance of external services and notify project participants in case of issues. Following this, they proceeded with automating the localization testing of the portal across 31 languages, which accelerated the testing process by five times.

A well-designed and executed approach to test automation led to the following results:

- The time required for deployment and execution of smoke tests was reduced by five times;

- Test coverage included 900 hours of manual checks daily, with an additional 200 hours per week;

- The downtime of the test environment was reduced 26-fold on a monthly basis;

- Investments in automation were fully recouped, yielding profit within six months of implementation [9].

These outcomes illustrate the effectiveness of a well-planned approach to test automation, significantly enhancing the quality and stability of the final product. The following recommendations focus on the process of implementing and optimizing automation processes:

1. The development of automated systems should be based on a modular principle, which allows for easy adaptation and scaling of the solution in response to changes in business processes and technological requirements. A modular architecture provides the ability to gradually replace outdated components without the need for a complete system overhaul.
2. It is recommended to invest in specialized training programs, seminars, and workshops to help staff recognize the advantages of new technologies and develop the necessary competencies to work with them.
3. Synchronization of automated processes with the current business processes of the organization should be ensured. This includes close collaboration between departments, the development of unified standards and data exchange protocols, as well as the implementation of centralized quality management systems.
4. The iterative testing and analysis methodology is recommended, where each iteration is accompanied by the collection and analysis of data to adjust the strategy. This allows for a quick response to changes and adapts the automation system in accordance with identified issues and new opportunities.

5. Conclusion

In summary, test automation is a powerful tool for reducing development costs and improving software quality.

Implementing automation requires careful planning, selection of appropriate tools and methods, as well as regular assessment of its economic efficiency. A well-structured approach to automation not only reduces time and financial expenses but also significantly enhances the reliability and stability of software products. It is important to emphasize that automation is not a one-size-fits-all solution and needs to be adapted to the specific conditions and requirements of the project. Ultimately, comprehensive utilization of automation in testing contributes to the successful execution of projects and enhances their competitiveness.

References

[1]. Rauf M. A. et al. A cost-effective communication model for identifying requirements in global software development //Scientific reports. – 2023. – vol. 13. – No. 1. – p. 18730.

[2]. How do I start automated testing from scratch? [Electronic resource] Access mode: https://translated.turbopages.org/proxy_u/en-ru.ru.ce469748-66b91996-c08497f3-74722d776562/https/www.geeksforgeeks.org/how-to-start-automation-testing-from-scratch / (accessed 08/09/2024).

[3]. L. M. et al. Adaptive ROI based on blocks for remote photoplethysmography //Multimedia tools and applications. - 2018. – vol. 77. – pp. 6503-6529.

[4]. Umar M. A., Zhanfang S. Research of automated software testing: automation tools and frameworks //International Journal of Computer Science Engineering (IJCSE). – 2019. – Vol. 6. – No. 217-225. – pp. 47-48.

[5]. Gadwal A. S., Prasad L. Comparative literature review on automated testing tools //Researchgate. – 2020. – Vol. 10. – p. 13140.

[6]. Okezi F., Odun-Ayo I., Bogle S. Critical analysis of software testing tools //Physical Journal: A series of conferences. – VGD Publishing House, 2019. – vol. 1378. – No. 4. – p. 042030.

[7]. Sneha K., Malle G. M. Research of software testing methods and software testing automation tools //The 2017 International Conference on Energy, Communication, Data Analysis and Software Computing (ICECDS). – IEEE, 2017. – pp. 77-81.

[8]. Viklund K. et al. Obstacles to software testing automation: a systematic review of the literature //Software testing, verification and reliability. – 2017. – vol. 27. – No. 8. – p. 1639.

[9]. Eckhart M. et al. Ensuring the process of testing software for industrial automation //Computers and security. – 2019. – Vol. 85. – pp. 156-180.