Jovica Đurković
Jelica Trninić
Vuk Vuković

Test Software Functionality, but Test its Performance as Well

Article Info:

Management Information Systems,
Vol. 6 (2011), No. 2,
pp. 003-007

Received 12 September 2010
Accepted 10 February 2011

UDC 004.415.53/.538

Summary

Software product testing has great importance in the detection of errors appearing in the course of software development and reflecting directly on software quality enhancement before its implementation in the working environment. Special priority in the software product testing phase is given to testing software performance. In contrast to functional testing, which should show if software is capable of carrying out planned functions without making errors, performance testing should show if the software will realize planned tasks in accordance with previously defined and expected performance. Software performance testing tools are used for simulating conditions under which software will work.

In this article, the authors point to the importance of the testing phases in the software product development process and give a review of up-to-date testing techniques. The focus of the article is placed on software performance testing and overview of performance testing tools.

Keywords

software testing; software performance testing; performance testing tools

1. Introduction

The requirements of modern business operations speed up business processes to such an extent that their real-time realisation is regarded as standard procedure. Interaction with customers, suppliers, business partners and employees requires immediacy, which is no longer supportable by the classical information structure. The expansion of the Internet and the constantly intensifying migration from physical to virtual sphere lead organisations towards orienting their operations primarily towards e-business and Web technologies.

Developing successful Web and ERP solutions is based on meeting two fundamental criteria – namely, functionality and performance. Functionality implies that the system fully executes the planned user-oriented function, whereas performance refers to the system's ability to execute transactions and deliver information rapidly and precisely, regardless of the number of users using the system simultaneously, or the limitations related to hardware resources. Some of the software problems related to the performance domain can be prevented relatively easily, by way of iterative testing prior to their implementation in the working environment.

2. The Importance of Testing in Software Product Development

In retrospect, as early as 1994, The Walt Disney Company released its first multimedia game for children, *The Lion King Animated Storybook*. Back then, in the 1990s, many companies placed and promoted their own games on the market, while Disney invested additional effort and resources into promoting their product because this game was the pioneer. The initial sales were booming, so that the game very soon became the season blockbuster. However, the subsequent course of events was the last thing Disney would have wanted in commercial terms. Namely, on December 26 the same year, the telephones in Disney's after-sales support department grew red-hot. At the other end of the line, the voices of raging parents and weeping children protested because the software was not functioning. The problem that emerged hit the newspaper headlines as well TV news (Patton, 2006, p.10).

As it turned out, Disney had made a serious oversight in the software testing field, failing to perform tests on a representative sample of PC models currently present on the market. The software did function on systems used by programmers while designing the software, but not on systems with the highest commercial, i.e. public market presence.

Software errors may occur in all software development phases. It is essential to emphasise

that error elimination, i.e. debugging costs in the earlier phases are multiply lower than those incurred at later stages.

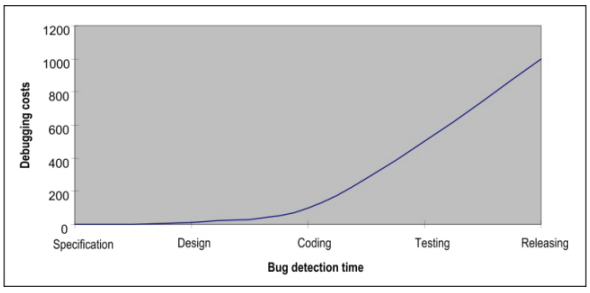

Figure 1 Costs of bug fixing at different stages of SDLC

Detecting and correcting errors in the software specification phase (Figure 1) may be cost-free, or the cost may amount to only one monetary unit (m.u.). The same error detected in the programming or testing stage may cost between 10 and 100 m.u.s, whereas, if the error is detected by the customer, error correction or elimination costs will grow multiply and may amount up to thousands of monetary units.

For better understanding, we shall consider the above mentioned Disney case. The principal cause of the problem may have been the fact that the game was incompatible, and therefore unable to function on every currently existing PC platform. If, perchance, the verification of which PC platforms were the most frequent had been done in the specification writing phase, and if the specification had stated that the software must be tested on existing configurations, the costs would have been minimised. The other option would have been a situation in which software testers would have gathered information on current PC platforms and verified the software on them. The error would have been detected in this stage, but the costs would have been significantly higher, as such situations require debugging and repeated testing. The third option would have been a situation where the preliminary software version could have been sent to a smaller user group, for so-called beta testing (Patton, 2006, p.18).

One of the principal causes of the inadequate software testing method can be sought in the fact that most software testers start from the following views of testing:

- testing is a process demonstrating that errors are not present in the software; or
- the purpose of testing is to demonstrate that the software successfully performs the planned functions; or
- testing is the process of establishing the point that the programme does what is specified, and what it is expected to do.

The above listed interpretations are, in fact, inappropriately applied. In the software development process finalised by testing, an important quality factor is incorporating value added into the software. This refers to enhancing the software's quality i.e. reliability. Enhancing reliability involves detecting and eliminating software errors.

Testin must be initiated with the supposition that software is virtually bound to contain some errors, which involves comprehensive approach to software testing, in order to detect the maximum possible number of errors. For this reason, software testing should be understood from the standpoint illustrated by the following definition: "Testing is a process of controlling and verifying software performance aimed at detecting errors." However, regardless of the stated aspects of software testing, one must bear in mind that it is impossible to detect absolutely all software errors (Myers, 2004).

3. Testing Techniques

All the existing techniques in the software testing process can be grouped into two strategies, i.e. black-box testing and white-box testing.

Black-box testing is a strategy where testing is performed exclusively based on software requirements and specification. This strategy does not require knowledge of internal pathways, structures or implementation of the software tested. The black-box testing process includes the following steps:

1. requirement specification analyses;
2. entering prepared correct, i.e. real and correct test data into the system tested based on requirement specifications, so as to verify that it processes them correctly; incorrect test data is also entered into the system so as to verify that the system recognises it and submits a status report to the system user;
3. determining the expected outputs for the selected inputs;
4. creating tests based on the selected inputs;
5. test implementation;
6. comparing the real, i.e. received outputs to the expected ones.

The principal shortcoming of the above approach to testing is the fact that testing and

validation specialists, regardless of the experience they possess, will never be able to know what proportion of the system has been tested. Some of the black-box testing techniques include Equivalence Class Testing, Boundary Value Testing, Decision Table Testing, Pairwise Testing, Domain Analysis Testing etc.

Contrary to black-box testing, white-box testing is a strategy where testing is based on the internal pathways, structure and implementation of the software tested, and which generally requires possessing comprehensive programming skills. Generally, the white-box testing process includes the following steps:

1. analysing the implementation of the software tested;
2. identifying pathways through the software tested;
3. selecting inputs so as the software tested can perform the identified paths;
4. in addition, determining the expected results of these inputs;
5. implementation
6. comparing the real outputs to the expected ones.

The problem with this approach is that larger applications must have a large number of functioning performance paths, which cannot be adequately tested due various factors, such as time and human resources in the first place. Some of the white-box testing techniques include Control Flow Testing and Data Flow Testing (Copeland, 2004).

4. Software Performance Testing

In addition to functionality, some other features are important for software quality, such as performance, availability, safety and security, and maintainability. They must inevitably be given appropriate importance, so as to avoid possible software failures and malfunctions, and consequent unforeseen after-sale costs.

Software performance is an indicator showing the degree of a software product's ability to meet the defined requirements over a given time period. To illustrate the significance of software performance, the subsequent section of the article describes a real-life case. If a sales representative, for instance, communicates by phone with a prospect, i.e. potential customer who is interested in a certain product and requests an offer for a thousand units of the product, the sales rep may not be able to access the system each time due to overload, i.e. the system will not be able to provide connection teach time. The outcome of the situation could include two options – the first, in which the prospect would find another sales outlet to buy the product, and the second, somewhat more favourable, in which the customer would call again after some time. To avoid the occurrence of this and similar situations, it is essential to test software performance before its delivery to the final customers for use.

What is, in fact, software performance testing? Performance testing is defined as technical examination with the basic purpose of determining, i.e. assessing the features in terms of speed, scalability and/or stability of the systems or applications that are tested. Speed refers to whether the application responds fast enough for the planned number of users. Scalability is the system's ability to provide for defined requirements in terms of application's response time and application throughput in the conditions of peak application use. Finally, stability refers to verifying whether the application is stable in the planned and peak application use intensity. Stress and load testing are regular sub-categories of software performance testing.

Stress testing focuses on determining or evaluating the characteristics or performances of the application tested in the conditions, i.e. under stress which is above the levels anticipated in practice. Moreover, this form of testing may involve system or application test in "harder" working conditions such as limited memory and disk space. These tests are specifically designed to determine under which conditions an application may fail, how and which indicators can be monitored preventively so as to point to potential failures. Errors and shortcomings detected by this form of testing may be fully eliminated, but not necessarily, which is a question of assessment of the testing team.

Load testing involves exposing a system to statistically representative, normal or expected load. Normal load includes minimum and maximum load that a system can endure without the need for engaging additional resources, as well as functioning of an application without excessive delays.

Microsoft Corporation has established a total of seven core activities in their approach to software performance testing based on acquired experience, representing an excellent methodological framework for performing this form of testing. These are (Meier, Farre, Bansode, Barber, & Dennis Rea, 2007):

1. Identify Test Environment – involves identifying hardware, software and network resources, as well as tools and resources available to the testing team;
2. Identify Performance Acceptance Criteria – involves defining parameters such as application response time (e.g. a product catalogue must be displayed in not longer than three seconds), throughput (e.g. the application must serve 25 books orders per second), resource utility (i.e. maximum processor utilisation is 75%);
3. Plan and Design Tests – involves identifying key scenarios and defining data for testing;
4. Configure Test Environment – involves preparing the working environment for testing, tools and resources required for execution of all potential testing strategies; in other words, the working environment should provide a framework for testing all system components;
5. Implement Test Design – involves developing performance tests in accordance with earlier designed tests;
6. Execute Tests – involves initiating and monitoring the test execution progress; tests, test data and gathered results must be verified;
7. Analyse, Report and Retest – involves consolidating and distributing test results. After the outcome analysis, some tests can be repeated if required. When all the values match the previously defined acceptance values, and when all the desired information is gathered, testing can be regarded as completed.

The benefits achievable in the software testing process are:

- determining characteristics related to the application's speed, scalability and stability;
- obtaining the answer to the question whether the system user will be satisfied with the application performance features
- identifying differences between the expected and real performance-related features
- verifying the adequacy of the software the application will be used on;
- detecting errors in the domain of functionality occurring in performance testing;
- assistance in determining the number of users the application can serve without questioning the application's performance;
- providing the assessment how much load above the designed the application can endure before errors occur that may result in application slowdown.

Experience has shown that there issues that still remain unresolved in the software performance testing process. Some of them are:

a) if the tests are not carefully designed, performance-related characteristics can be indicators only in a small number of cases in the application's real-life working conditions;
b) until the tests are performed on hardware that will be used by users, there will always be a certain degree of discrepancy in the testing results.

5. Rational Performance Tester – Software Performance Testing Tool

The market offers numerous software performance testing tools, including Microsoft Web Application Stress Tool (WAS), Apache Jmeter, ANTS - Advanced.NET Testing System, Load Manager, etc. In this analysis of tools available on the market, the authors of this article have decided to analyse one of the most popular ones – the IBM Rational Performance Tester.

IBM Rational Performance Tester, or Performance Tester for short, is a tool for creating, implementing and analysing software performance tests providing support to development teams when assessing the scalability and reliability of Web and ERP applications before their implementation in the working environment.

Saving and analysing tests in this tool involves interaction with selected Web applications or ERP solutions by way of browsers or ERP clients. Test results are reported in a summarised and easily viewable editor. The software's dynamic responses are identified and automatically placed in the function of data-guided testing enabling the change of input data, thereby eliminating the need for changing test manually. The tool can display all the pages visited in offline mode, and thereby provide a review of all user's interactions and transactions. Test scripts can be grouped into numerous various combinations, thus enabling the simulation of various user groups and transactions. Test execution is accompanied by reports created in real time, easy to read, and updated as the test is executed. The greatest contribution of Rational Performance Tester is its ability to point to bottlenecks in the entire transaction, and then the specific spots representing the cause of these bottlenecks.

It is essential to note that Rational Performance Tester is an integral component of IBM Rational Software Delivery Platform. Rational Software Delivery Platform is currently the most complete and powerful solution in IT industry, both for software development and information system life cycle management. Practically, this platform provides support for automation of all the phases and aspects in the software development process. When it comes to performance testing, it means that development staff, testing specialists and operative team members can design, share and analyse performance tests from the same user interface used for developing, testing, implementing and monitoring applications.

Rational Performance Tester does not require high processor and memory capacities when performing simulation with a large number of users. Consequently, this may result in reaching high scalability levels, although the testing team does not have strong hardware resource support at their disposal. In other words, test implementation and result review can be performed on Windows, UNIX and Linux devices, thus enabling the use of existing hardware resources, without the need for additional expenditure (IBM, 2007).

6. Conclusion

Software product testing refers to a process developing through each individual stage of the development cycle, so as to secure software efficiency and availability, which confirms the importance of considering this software development aspect. The set of complex activities accompanying software testing is aimed at assessing its compliance with specified requirements, and, at the same time, verify its appropriate and high-quality functioning. Focussing on only one segment due to the complexity of the issue, the article gives priority to system performance testing. This form of testing includes a number of successive phases, which provide a high-quality software product and enhance its quality only as a compatible process.

Undoubtedly, the priority in software performance testing belongs to software testing tools, which provide support in the efficient application of the standard testing procedure and evaluation of the existing techniques. The use of standards and appropriate testing tools as a support can result in high-quality software, with efficient functionality and elimination of any type of errors, i.e. reducing their occurrence to minimum tolerance levels.

References

IBM. (2007). *IBM Rational Performance Tester*. Retrieved August 21, 2010, from IBM: ftp://ftp.software.ibm.com/software/rational/web/datasheets/rpt_ds.pdf

Copeland, L. (2004). *A Practitioner's Guide to Software Test Design*. Boston-London: Artech House Publishers.

Meier, J.D., Farre, C., Bansode, P., Barber, S. & Dennis Rea, D. (2007). *Performance Testing Guidance for Web Applications*. Seattle: Microsoft Corporation.

Myers, G. (2004). *The Art of Software Testing*. Hoboken, New York: John Wiley & Sons.

Patton, R. (2006). *Software Testing*. Indianapolis: Sams Publishing.

Jovica Đurković
University of Novi Sad
Faculty of Economics Subotica
Segedinski put 9-11
24000 Subotica
Serbia
Email: djovica@ef.uns.ac.rs

Jelica Trninić
University of Novi Sad
Faculty of Economics Subotica
Segedinski put 9-11
24000 Subotica
Serbia
Email: trninicj@eunet.rs

Vuk Vuković
University of Novi Sad
Faculty of Economics Subotica
Segedinski put 9-11
24000 Subotica
Serbia
Email: vuk@ef.uns.ac.rs