Performance Testing of New Enterprise Applications using Legacy Load Data
A HIS Case Study

Marek Miłosz
Institute of Computer Science, Lublin University of Technology, Nadbystrzycka 36B, 20-618 Lublin, Poland

Keywords: Performance Testing, Operational Profile, Legacy System Reengineering, Hospital Information System.

Abstract: Legacy information systems are increasingly a subject to reengineering with the progress of ICT. There are developed systems with similar features to replace those which operate. Developed software needs testing. This paper focuses on the importance of performance testing in the early stages of software development. To performance tests be credible, the appropriate load profile of the system must be used. The paper presents a method of using legacy system load of real transaction data to design performance tests for newly developed systems. The example of the hospital information system (HIS) shows the use of this method in the software development using agile methods. Presented approach allowed the early detection of performance problems of the new software.

1 INTRODUCTION

Each methodology of software development has a phase of software testing. Tests can be generally divided into functional and non-functional testing. Normally non-functional tests concern the performance of the system, their usability, interoperability, portability and reliability (Hutcheson, 2003); (Naik and Tripathy, 2008); (Romano et al., 2009). Many authors have indicated the importance of performance testing and the need for appropriate methodologies for their implementation (du Plessis, 2008); (Yao et al., 2006); (Myers, 2004). Unfortunately, during software development, functional and non-functional, especially performance tests, are often split (Perry, 2009). This split is shown in the Figure 1. This situation causes big problems when Configuration Tuning (Figure 1) does not lead to the achievement of the required system performance. In such cases, the return to the Development phase is required (see System Redesign in Figure 1), which usually is related to serious consequences in the aspect of increased costs and delays of the project.

In iterative processes of software development it is possible to include performance testing to the software development process (Figure 2). This allows permanent (in tight cycles) control of created software performance parameters.

There is a need for defining two main groups of data used for the performance tests. The first is the performance requirements. These are the performance acceptance criteria of software which are expected or desired by the customer. They define

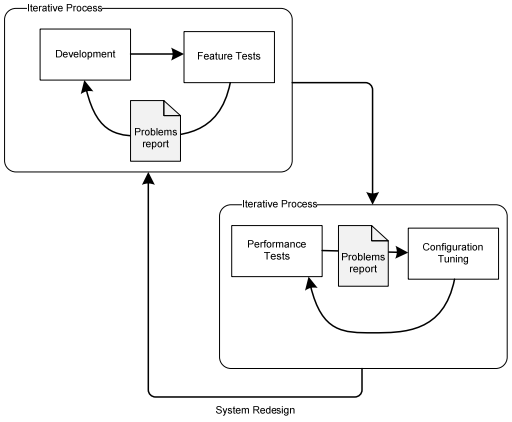

Figure 1: Traditional approach to performance testing.

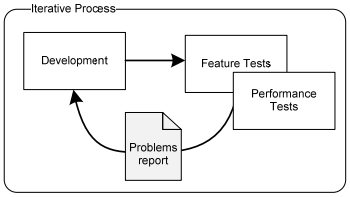

Figure 2: Early performance testing in iterative development (Perry, 2009).

Miłosz M..
Performance Testing of New Enterprise Applications using Legacy Load Data - A HIS Case Study.
DOI: 10.5220/0004433501510156
In *Proceedings of the 15th International Conference on Enterprise Information Systems* (ICEIS-2013), pages 151-156
ISBN: 978-989-8565-60-0

the goals and constrains of performance testing (Meier et al., 2007). They must be measurable and have specific values (Weyuker and Vokolos, 2000). The problem is that the customer is not always able to identify accurately those requirements (Parnas, 1994). It is associated with a lack of understanding of the objectives and metrics of performance parameters measures (e.g. the customer does not fully understand the concepts: end-user response time, processor utilization, the number of transactions per unit of time or the bottleneck), and the importance of their specific values (Hutcheson, 2003). It is also difficult to translate business requirements (such as: "This application will support up to 2,500 concurrent users ") for performance criteria and specific values of parameters (Meier et al., 2007).

The second group of data used for the performance tests is the load parameters of the system used for planning tests. These parameters should be set in the client requirements and be a part of the performance testing constrains. In large systems, the load is very diverse (many different actions are performed by different users) and changes over time. This leads to difficulties in their determination. It is reduced to identifying and defining so-called "Operational Profile" (Perry, 2009) with it random variation in time. Operational Profiles can be obtained from empirical research on existing applications using statistical methods (Zaman et al., 2012). For non-existing application and the lack of analogues, developing the Operational Profiles is difficult. Some researchers suggest creating mathematical models of applications, and try to analyse them, using computer simulation, in terms of performance (Hao and Mendes, 2006); (Avritzer and Weyuker, 2004). Operational Profiles are used to design test scenarios in a manner corresponding to client requirements (Hao and Mendes, 2006); (Perry, 2009); (Xiao-yang et al., 2010).

Performance tests can be carried out using the real end-users experiment on application installed on production platforms. Such approach is organizationally difficult and very costly. Therefore, in practice, there are used special software tools for performance testing running on special testing platforms (Poston and Sexton, 1992); (Sakamoto et al., 2002); (Zhen et al., 2009); (Romano et al., 2009). These tools enable planning, load generation, execution and analysis of tests' results (Netto et al., 2011); (Myers, 2004). The security aspect should be carefully considered during tests. To ensure security all software should be installed on the test platform and checked before implementing in the production system (Kozieł, 2011). Transferring of the applications from the test platform into production should also be tested.

Another problem is the analysis of the results of performance tests (Zhen et al., 2009) and presenting the results in terms of business requirements, necessary to report the results of the testing process (Perry, 2009).

2 PERFORMANCE TESTS IN AGILE PROJECTS

According to Figure 2 in agile projects (as iterative and incremental styles of software development), performance tests can thus be performed in parallel with the functional tests, immediately after the appearance of first application builds (du Plessis, 2008); (Meier et al., 2007).

Immediately after the functional testing of the first application build, the performance test can be started and provides its results to the developing team. Developers gain in this way up-to-date feedback on performance parameters of software. This also applies to other non-functional tests such as usability tests (Luján-Mora and Masri, 2012). Performance tests are also included in the Test Driven Development as a test first-performance technique to ensure the performance of the system development (Johnson and Maximilien, 2007); (Borys and Skublewska-Paszkowska, 2011).

Effective use of performance testing in agile development processes is conditioned by the possibility of using special software tools to automate it. Without it, repeated testing of applications is too costly. Selection of software tools to automate performance testing usually strongly associated with a type of an application development environment.

Agile methods are used not only for developing of completely new applications, but they are also useful in modernization and reengineering projects (Bin Xu, 2005).

3 TYPICAL PROCEDURE FOR PERFORMANCE TESTING OF ENTERPRISE APPLICATIONS

Methodology of performance testing usually consists

of the following seven core phases (Meier et al., 2007):

1. Identity Test Environment.
2. Identity Performance Acceptance Criteria.
3. Plan and Design Tests.
4. Configure Test Environment.
5. Implement Test Design.
6. Execute Tests.
7. Analyze, Report, and Retest.

Test Environment and Acceptance Criteria should be defined by the client as non-functional requirements.

During the third phase (Plan and Design Tests), it is necessary to determine (Meier et al., 2007) key tests scenarios, representative users (the most demanding in the performance area) and its activities, and to define test data. All these aspects result from the Operational Profile assumed or discovered during researches.

4 DISCOVERY OPERATIONAL PROFILE USING LEGACY DATA

Data from a functioning and operating application log file (if it contains the appropriate records) can be used to find the Operational Profiles. They must be a subjected to various transformations (Figure 3) as follows:

- Removing Unnecessary Data - log files can contain data unnecessary for analysis (for example IP of users executing operations in applications or ID of users), which should be removed.
- Data Conversion - log files usually contain text records, and therefore they need to be converted into other, appropriate for data analyses software formats (numeric, dates, etc.).
- Additional Classification - log files usually contain very detailed data such as the user who performed the operation, and did not include his or her classification (e.g. type of an employee, the department where he or she is employed, etc.). Therefore, there is a need to add additional data that would later allow performing statistical calculations and generalizations.
- Statistical Analyse - acquisition of load data of different types of users and their activities in the system are used to calculate parameters of short- and long-term time series.

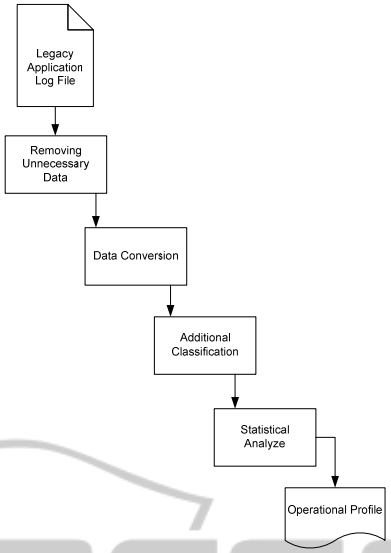

Figure 3: Legacy application log data processing.

5 PERFORMANCE TESTING OF NEW HIS – A CASE STUDY

Legacy Hospital Information System (HIS) is a large, web application with data base, which registers events related to patients's service - from their registration in the hospital to their release, with recording parameters of all the procedures and results. HIS also supports the activities of doctors, nurses and other personnel in the area of patient's service. It also has extensive reporting functions, which are necessary to manage hospital and the settlements with the Polish National Health Fund (PNHF), which pays for medical treatments.

An ICT company has taken actions to develop the new HIS, which would implement the functionality of the used system (with small extensions) with the change of implementation technology. The legacy system was built using old client-server technology. The new system is built as a web application using Java EE technology and an Oracle database server. Reengineering project is implemented using agile methods. The methodology of the creation of new software contained a built-in process performance testing using legacy data of the system load. The legacy data were obtained from log files of HIS, which operated in a large hospital (Ziarno and Zyga, 2012). Source data covers the whole 2011 year. It allowed building end-users Operational Profiles used in performing tests.

HIS systems belong to the class of reliable systems. Therefore, during their operation, all of the

critical hospital actions are recorded. The log file of HIS for 2011 consists of more than 500 thousand records (Ziarno and Zyga, 2012). This data provides important information (from the point of view of the performance tests creation) and unnecessary data. Unnecessary data is the data of session parameters, end-user IP address, etc.

Data from the log files has been processed as it is shown in Figure 3. The primary objective of this data analyse was to determine:

- types of the system users;
- the structure of the actions carried out in the HIS;
- load actions parameters.

Analysis of the data showed that there are two main users (in terms of application load) of HIS: doctors and nurses. Other groups insignificantly load the system. Therefore they were combined into a group "others".

Total HIS load, as it was expected, is variable in time. 24-hours variability (Figure 4) is the result of the hospital schedule, and contains the daily peak (period from 8am to 2pm) and the decrease in activity during the night. From 1 to 5 am HIS load comes down to zero.

System load is characterized by the weekly periodicity with the falling down in weekends (Figure 5). Over the year, there has been an increased of the system load in January (Figure 6). This is related to the settlements with PNHF.

The analysis of legacy HIS usage data also allowed determining what actions were performed by each type of users. Depending on the particular department, distribution of particular actions was variable (Figure 7).

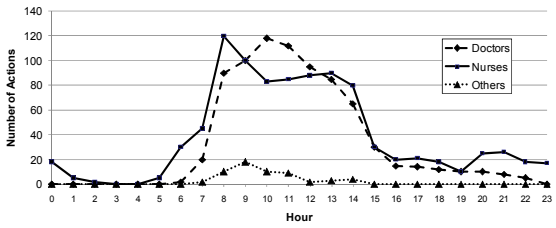

Figure 4: Daily HIS load (work day).

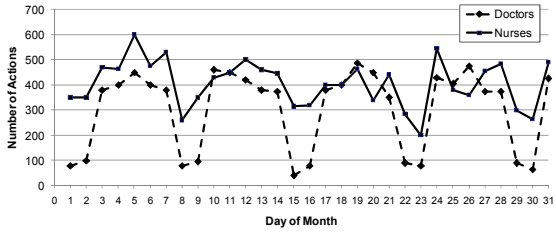

Figure 5: Monthly HIS load made by doctors and nurses.

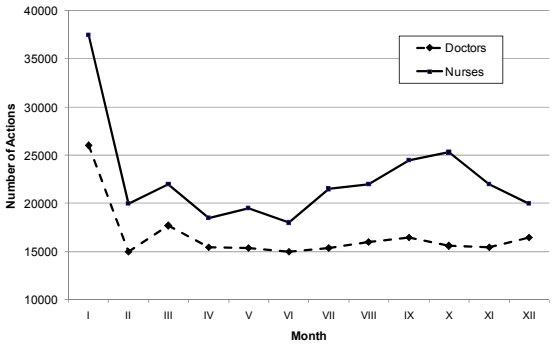

Figure 6: Yearly HIS load.

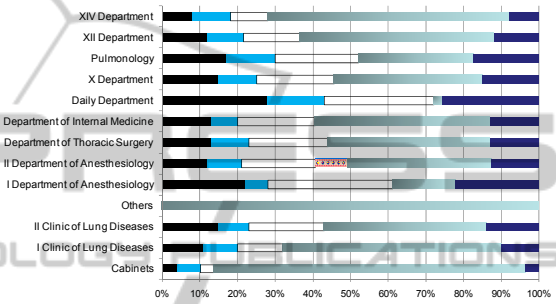

Figure 7: Distribution of doctor actions in different hospital departments (an example).

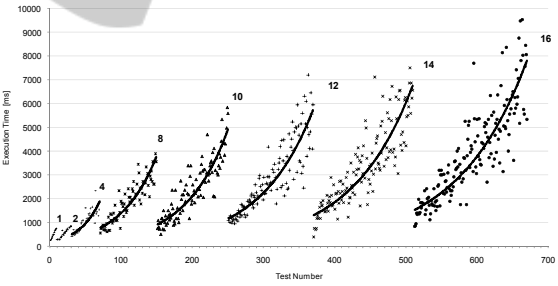

Figure 8: Test results for scenario: adding new item to the patient drugs list for different number of concurrent users (for 1, 2, 4,…, 16 users).

Obtained data was used to carry out the performance tests of the new system. The tests were performed for the most common actions and for different numbers of concurrent users, realizing typical OFAT (one-factor-at-a-time) approach (Perry, 2009). An example of the results is shown in Figure 8. Execution time increases with the number of drugs in the list (Figure 8). Unfortunately, this increase is not linear but exponential, and can cause big performance problems, which must be fixed as soon as possible. It was one of cases proving usefulness of early performance testing in iterative development. After this, there was completed Load

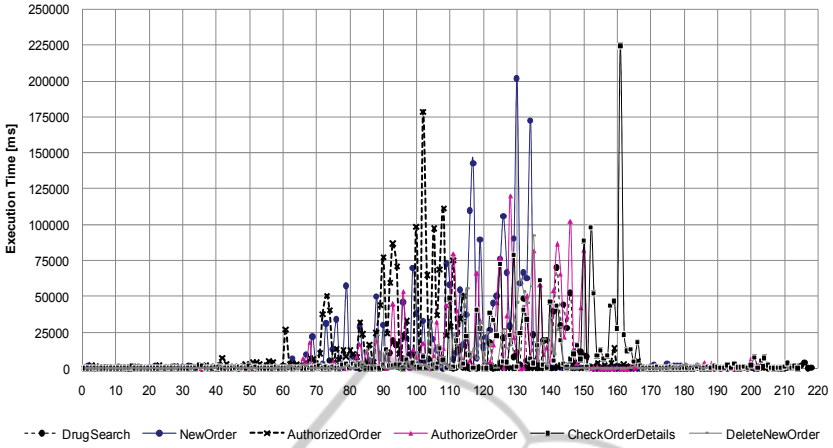

Figure 9: Load Test for doctor's Operational Profile (scenario: daily order operations).

Test, using a one-day system load parameters (Figure 9). All tests were performed using Apache JMeter.

In addition to the charts, the results of the tests were averages, minimum and maximum execution times of individual actions. They have allowed indicating operations that the new system performs significantly slower. For example, in the first release of the new software, download operation of a disease history record performed almost 10 times slower than in the legacy application. Results of testing has been used to improve of performance during the second release development.

6 CONCLUSIONS

Nowadays, old legacy information systems are gradually replaced with new ones. In the process of creating new systems it is possible to use data and knowledge acquired from the operation of the old systems. This mainly concerns the functionality of the software, but also it is possible to use the system's operation data to create the test procedures including the load tests. Load data of the old system, collected for years, is a source of knowledge that allows for precise determination of the parameters of load testing. Analysis of log files allow specifying who, when and how significantly load the old system. It results in the Operational Profiles, which can be used to test plan accurately and fully dimensioned.

Including the performance tests in an iterative process of developing software enables early detection of performance problems and correction them in subsequent iterations. This approach is reasonable because it discharges developers from the work on improving the software part that meets the performance requirements. This allows them to focus on the most important fragments (in terms of performance) of the code.

The case study presented in this paper uses this approach and shows the success of simultaneous use of legacy load data and early performance testing in the big information system developing. This tests reflected the load of old system and allowed check the performance of each module in more realistic conditions. This is the primary advantage of the proposed method.

The major problem in the use of legacy data load is the need for labour-intensive, manual processing of the data. Due to a big diversity of log files structure acquire the necessary data becomes nontrivial problem. During the building of the Operational Profiles it is necessary to clean and remove unnecessary data, partition and classification it and statistical analyse. It increases the cost of construction the Operational Profiles. This is the primary disadvantage of the proposed method and the area of a major challenge to solve it in the future.

ACKNOWLEDGEMENTS

All practical data presented in this paper have been obtained by my Master students during work on their master's diploma (Ziarno and Zyga, 2012). Thank them for their contribution to this paper.

REFERENCES

Avritzer, A., Weyuker, E. J. 2004. The role of modeling in

the performance testing of e-commerce applications. *IEEE Transactions on Software Engineering*, vol. 30, no. 12, pp. 1072- 1083.

Bertolino, A., 2007. Software Testing Research: Achievements, Challenges, Dreams. In *IEEE Computer Society Conference Future of Software Engineering (FOSE '07),* pp. 85-103.

Bin Xu, 2005. Extreme programming for distributed legacy system reengineering. In *29th Annual International Computer Software and Applications Conference*, COMPSAC 2005, 26-28 July 2005, vol. 2, pp. 160- 165.

Borys, M., Skublewska-Paszkowska, M., 2011. Experience of Test-Driven Development adoption in software industry, *Applied Methods of Computer Science*, Polish Academy of Science, vol. 4/2011, pp. 5-13.

du Plessis, J., 2008. *Performance Testing Methodology. White Paper*, Micro to Mainframe (www.mtom.co.za, [20.01.2013]).

Hao, J., Mendes, E., 2006. Usage-based statistical testing of web applications. In *Proceedings of the 6th international conference on Web engineering (ICWE '06)*, ACM, New York, pp. 17-24.

Hutcheson, M., 2003. *Software Testing Fundamentals - Methods and Metrics*, Wiley Publishing Inc., Indianapolis, Indiana.

Johnson, M. J., Maximilien, E. M., 2007. Incorporating Performance Testing in Test-Driven Development, *IEEE Software*, vol. 24, no. 3, pp. 67-73.

Kozieł, G., 2011. Information security policy creating. *Actual Problems of Economics*, vol. 126, no. 12, pp. 367-380.

Luján-Mora, S., Masri, F., 2012. Integration of Web Accessibility into Agile Methods. In *Proceedings of ICEIS 2012 Conference*, pp. 123-127.

Meier, J. D., Farre, C., Bansode, P., 2007. *Performance Testing Guidance for Web Applications. Patterns & Practices*, Microsoft Corporation.

Meyers, G., 2004. *The Art of Software Testing*, John Wiley & Sons Inc., Hoboken, New Jersey, 2nd edition.

Naik, K., Tripathy, P., 2008. *Software Testing and Quality Assurance. Theory and Practice*, John Wiley & Sons Inc., Hoboken, New Jersey.

Netto, M. A. S., Menon, S., Vieira, H.V., Costa, L. T., de Oliveira, F. M., Saad, R., Zorzo, A., 2011. Evaluating Load Generation in Virtualized Environments for Software Performance Testing. In *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW),* 16-20 May 2011, pp. 993-1000.

Parnas, D. L., 1994. Software Aging. In *Proceedings of the 16th international conference on Software engineering, ICSE '94*, Los Alamitos, CA, pp. 280-287.

Perry, D., 2009. Understanding Software Performance Testing, *Better Software Magazine*, April 2009 (http://www.stickyminds.com/BetterSoftware/magazine.asp?fn=cifea&id=118, [20.01.2013])

Poston, R. M., Sexton, M. P., 1992. Evaluating and selecting testing tools, *IEEE Software*, vol. 9, no. 3, pp. 33-42.

Romano, B. L., Braga e Silva, G., de Campos, H. F., Vieira, R. G., da Cunha, A. M., Silveira, F. F., Ramos, A., 2009. Software Testing for Web-Applications Non-Functional Requirements. In *Sixth International Conference on Information Technology: New Generations, ITNG '09*, 27-29 April 2009, pp. 1674-1675.

Sakamoto, M., Brisson, L., Katsuno, A., Inoue, A., Kimura, Y., 2002. Reverse Tracer: a software tool for generating realistic performance test programs. In *Eighth International Symposium on High-Performance Computer Architecture*, 2-6 Feb. 2002, pp. 81- 91.

Weyuker, E. J., Vokolos, F. I., 2000. Experience with performance testing of software systems: issues, an approach, and case study, *IEEE Transactions on Software Engineering*, vol. 26, no. 12, pp. 1147-1156.

Xiao-yang Guo, Xue-song Qiu, Ying-hui Chen. Fan Tang, 2010. Design and implementation of performance testing model for Web Services. In *2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR),* vol. 1, 6-7 March 2010, pp. 353-356.

Yao Jun-Feng, Ying Shi, Luo Ju-Bo, Xie Dan, Jia Xiang-Yang, 2006. Reflective Architecture Based Software Testing Management Model. In *IEEE International Conference on Management of Innovation and Technology*, 21-23 June 2006, vol. 2, pp.821-825.

Zaman, S., Adams, B., Hassan, A. E., 2012. A Large Scale Empirical Study on User-Centric Performance Analysis. In *IEEE Fifth International Conference on Software Testing, Verification and Validation (ICST),* 17-21 April 2012, pp. 410.419,

Zhen Ming Jiang, Hassan, A. E., Hamann, G., Flora, P., 2009. Automated performance analysis of load tests. In *IEEE International Conference on Software Maintenance, ICSM 2009,* 20-26 Sept., pp. 125-134.

Ziarno, A., Zyga, E., 2012. *Performance testing of corporate applications*, M.Sc. thesis under M. Miłosz supervising, Lublin Technical University, 120 p.