2012 International Conference on Information and Computer Applications (ICICA 2012)
IPCSIT vol. 24 (2012) © (2012) IACSIT Press, Singapore

The Usage of Performance Testing for Information Systems

Pavol Tanuska, Ondrej Vlkovic, Lukas Spendla

Institute of Applied Informatics, Automation and Mathematics, Slovak University of Technology, Paulinska 16, SK-91724 Trnava, Slovakia, pavol.tanuska@stuba.sk

ABSTRACT: The main focus of this article is the possibility of the performance testing for information systems. Our proposal focuses on steps, which perform the performance testing itself. The individual steps of our proposal have been mapped for each phase of the software testing standard IEEE 829. To visualize the process of testing, the model was outlined using UML diagrams.

KEYWORDS. System testing, Performance testing, UML

1. Introduction

Testing is an essential pre-requisite for successful implementation of software product. It belongs to one of the most important phases of software life cycle and each system must be tested. Testing can be done by either manual or automation testing [1]. One of the most important phase of the testing is system testing, that largely takes advantage of test automation.

1.1 System testing

System testing is often used as a synonym for "black-box testing", because during system testing the test team concerns itself mostly with the application's "externals" [2].

System testing is done to find out those imperfections that were not found in tests conducted earlier. This includes forced system failure and validation of total system as it will be put to use by its user(s) in the actual working environment. It generally starts with low volumes of transaction based on real data. Gradually, the volume is increased until the maximum level for each transaction type is attained [3].

1.2 Performance testing

Performance tests verify that the system application meets specific performance efficiency objectives. Performance testing can measure and report on data as input / output rates, total number of I/O actions, average database query response time and CPU utilization rates [2]. Performance testing measurement and criteria

In order to accurately measure performance there are a number of key indicators that must be taken into account [4]. These indicators are part of the performance requirements, simply we can divide them into two types: service-oriented and efficiency-oriented [5].

Service-oriented indicators are availability and response time; they measure how well (or not) an application is providing a service to the end users. Efficiency-oriented indicators are throughput and utilization; they measure how well (or not) an application makes use of the application landscape [5].

1) Concurrency user

In practical testing, testing engineer pays more attentions to business concurrency users, that is how many concurrency users in the business model is available. In equation (2), the C is the mean concurrency users, n is the amount of login session, L is the mean length of login session, and T is the inspected time. A login session is a time interval defined by a start time and end time[6].

$$C = n * L / T \tag{1}$$

$$Cp = C + 3 * \sqrt{C} \tag{2}$$

2) Throughout

Throughout is the amount of users requests processed within one second. Throughout is the directly load performance, it is normally defined by hits per second or pages per second, there are two aspects role. One is used to design performance testing scenery, verify performance testing scenery achieved test object or not, the other one is to analyze performance bottleneck, the limit of throughout is the mainly aspect of performance bottleneck. Throughout is related to concurrency users when no bottleneck happened, defined as [6]:

$$F = Nvu * R / T \tag{3}$$

1.3 Standards for software testing

Over the years a number of types of document have been invented to allow for the control of testing. They apply to software testing of all kinds from component testing through to release testing. Every organization develops these documents themselves and gives them different names, and in some cases confuses their purpose.To provide a common set of standardized documents the IEEE developed the 829 Standard for Software Test Documentation for any type of software testing, including User Acceptance Testing.

This White Paper outlines each of the types of document in this standard and describes how they work together. There are eight document types in the IEEE 829 standard, which can be used in three distinct phases of software testing [7].

2. The Proposal of Performance Testing

The starting point for our proposal is standard for software testing IEEE 829. Its individual phases were slightly modified for better simplicity and clearness.

The phase Test Plan, describes how the testing will proceed and it is captured by modeled sequence diagram as a whole. Phases Test Log and Test Incident Report were merged to the Test Execution phase, mainly because of phases modeling difficulty [7].

Our proposal focuses on steps, which perform the performance testing itself. The previous identification and analysis as well as the following tuning activities are not captured by our proposed models.

2.1 The sequences of proposed performance testing

The first result of our proposal is the following sequence diagram (captured in Fig. 1), which captures the steps of automated performance testing in terms of time.

Among the first steps of our proposal belong: obtained requirement of the specification for testing system and creation of performance model. The performance model represents the analysis of different functionalities as well as the rates of use for these functionalities which the test system provides. These two initial points are captured with a certain degree of abstraction, because it is more complex and more difficult steps. These steps will be more detailed modeled in the next iteration of proposal.

The next step is the analysis of system that will be tested. The result of this analysis is an initial state of the system, therefore the state in which the system is before testing. The following step is to specify test objectives, system load at which testing is performed as well as condition, which determines when test stops.

These sequences are followed by the testing itself. In this part, the individual sessions are parallel created in such number as is required by definition of distribution. This distribution represents the rates of use for different functionalities as we described above.

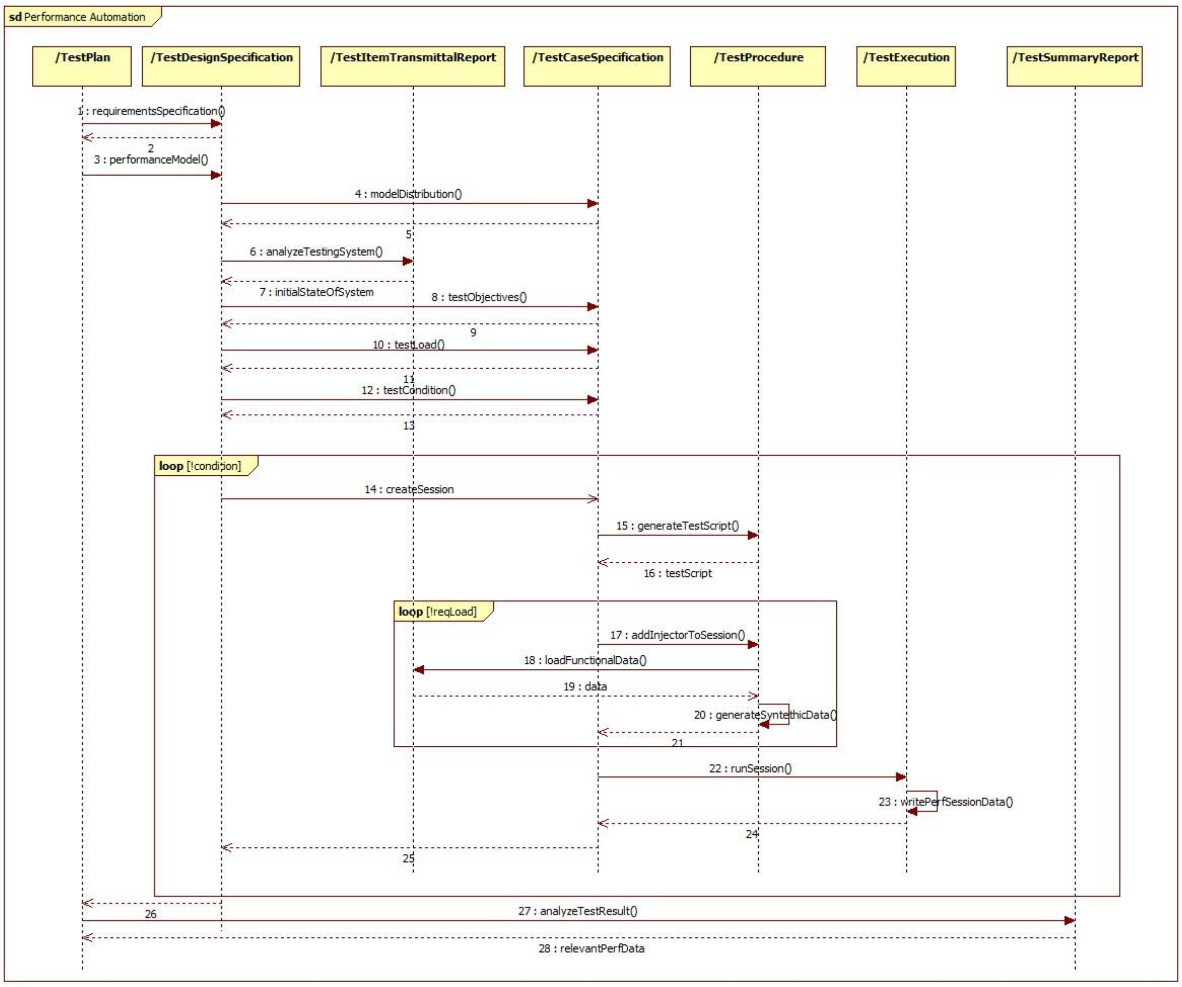

Fig. 1 Proposal of performance testing steps

In the individual sessions there are generated a test scripts, which represent a mix of end-users activities. In the next step, the injectors are assigned to the individual session, in number, which corresponds to the required load. In the injectors there are generated and loaded data, which are used for system load in session. The required data from the session of the performance testing are recorded. If the number of sessions is smaller than is required (e.g. one session ended) a new one is created (based on the rate from the distribution) and the whole cycle is repeated until the condition is fulfilled.

If the condition is not met, the testing itself ends, the results from individual sessions are analyzed and test will provide relevant results with regard to defined objectives.

2.2 The overview of states in proposed performance testing

The second result of our proposal is the following state machine diagram (captured in Fig. 2), which identifies the events that activate transitions. A set of states is captured as a synchronous sequence. This is mainly due to the complexity and clarity over asynchronous model. Because of the degree of abstraction of our model, we have not captured and described internal actions and activities in individual states.

The next step is the analysis of system that will be tested. The result of this analysis is an initial state of the system, therefore the state in which the system is before testing. The following step is to specify test objectives, system load at which testing is performed as well as condition, which determines when test stops.

These sequences are followed by the testing itself. In this part, the individual sessions are parallel created in such number as is required by definition of distribution. This distribution represents the rates of use for different functionalities as we described above.

A set of previous standard states, according to standard IEEE 829, is captured as initial state, named PreviousStandardStates. This complex state occurs as the first state in our proposed testing process. When the event gainedPerformanceModel occurs, testing process passes to the state ProblemDomainSpecification, where the input action is invoked. The role of this action is to define the rate of use described in the distribution for every possible functionality. The state has also the task to specify the test load, and analyze the test system. At signalizing the output event, the output activity defines test objectives required for the test.

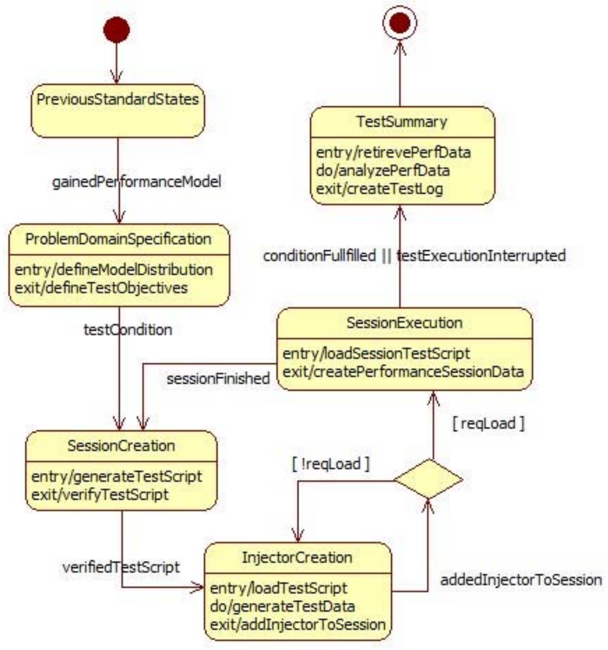

Fig. 2 Proposed state machine diagram

By the event testCondition the process passes into the state SessionCreation. The role of the input action is to generate the unique test script for current session, which is carried out by one or more Injectors. It should be noted, that as an output action, the verification of the generated test script must be carried out. After the verification of the test script the process moves to InjectorCreation state.

After reaching this state, the loadTestScript action is executed, which means that the previously created test script is loaded. In this state there are also loaded functional test data and generated the synthetic test data, which represent random actions in the injector script approach. Assigning a created injector to current session is an output action of this state.

When the event addedInjectorToSession occurs, testing process passes to the state SessionExecution, only if the required load is achieved. If the actual load is under required load, a new injector is created. Since our model is captured as synchronous sequence, the individual injectors in our model are created one after another.

The role of input action in SessionExecution is to load previously created session script. The main task of this state, is to run created injectors for current session. The role of output action is to gather the data and from them create session performance data.

If the condition is not fulfilled and the test execution is not interrupted, a new session is created with the probability described in distribution. Otherwise, the process passes to state TestSummary. Task of its input action is to retrieve performance data from all previous sessions. After that, the performance data are analyzed, and the test log is created, taking into account the test objectives.

3. Conclusion

The aim of this article was to design automation of performance testing. In our proposal we have focused on performance testing itself and therefore we don't present a proposal of load testing. Our proposal is based on the basic steps of performance testing, which are captured in our modeled diagram. It should be noted, that our proposal is based on the software testing standard IEEE 829. Our proposal was captured by sequence and state machine diagram in UML 2.0.

4. References

[1] TANUSKA Pavol; SPENDLA Lukas; VLKOVIC Ondrej. “Accelerated stress testing of control systems,” Annals of DAAAM and Proceedings of DAAAM Symposium. - ISSN 1726-9679. - Vol. 21, No 1. Annals of DAAAM for 2010 & Proceedings of the 21st International DAAAM Symposium 20-23rd October 2010, Zadar, Croatia, DAAAM International Vienna, 2010, PP409-410

[2] DUSTIN Elfredie; RASHKA Jeff; PAUL John. “Automated Software Testing: Introduction, Management, and Performance,” Addison-Wesley Professional, 1999, ISBN 978-0201432879

[3] Isrd Group, “Structured System Analysis And Design”, Tata McGraw-Hill Education, 2006, P432, ISBN 007-0612048

[4] SKAMLA Michal; HAMERNÍK Peter; VAZAN Pavol. “The evaluation of influence of priority rules in job shop scheduling,” Process Control 2010: 9th International Conference. Kouty nad Desnou, 7.-10. 6. 2010, University of Pardubice, 2010, ISBN 978-80-7399-951-3

[5] MOLYNEAUX Ian. “The Art of Application Performance Testing”, O’Reilly Media, Sebastopol, CA, 2009, ISBN 978-0-596-52066-3

[6] YUNMING P.; MINGNA X. “Load Testing for Web Applications,” The 1st International Conference on Information Science and Engineering, Dec 18th to 20th, 2009, PP2954-2957, 2009 ISBN 978-0-7695-3887-7

[7] TANUSKA Pavol; VLKOVIC Ondrej; SPENDLA Lukas. “The Proposal of Step Stress Testing Automation,” International Journal of Computer Theory and Engineering, Vol. 2, No 6, 2010, P860, ISSN 1793-8201