The usage of system testing for information and control systems

Pavol Tanuska, Lukas Spendla, Alojz Masar, Ondrej Vlkovic

Faculty of Material Science and Technology in Trnava

Slovak University of Technology in Bratislava

Abstract: The main focus of this article is the possibility of the performance and stress testing for information systems. Our proposal focuses on steps, which perform the performance and stress 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.

Key words: system testing; Informationa and Control Systems (ICS); performacne 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.

System testing is often used as a synonym for ñblack-box testingò, because during system testing the test team concerns itself mostly with the applicationôs ñexternalsò [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]. System testing also involves:

1.1. Performance testing

Performance tests verify that the system application meets specific performance efficiency objectives. Performance testing can measure and report on such data as input / output rates,

total number of I/O actions, average database query response time and CPU utilization rates [2].

1.2. Step Stress Testing

As an extension of stress testing we can consider step stress testing, which seeks to improve some of the stress testing features. A step stress test starts the same way a fully censored testing starts. A fixed number of parts run through one live, what is for example represented by 500 hours on Fig. 1. After one live, the stresses applied to the product are elevated in steps in order to precipitate failures. The precision of the test depends on three key factors:

- Number of parts tested
- Number of lives tested
- How the stresses are increased

The grater the number of parts, the more precise (and more accurate) the test is. Increasing the number of lives decreases the number of parts needed, but introduces an assumed distribution in the population. Increasing the stress reduces increases the graduation on the time-to-failure. At low stress, the time to accumulate damage is very long and differences between time-to-failure will also be long.

As stress is increased, the rate of damage increases, the time to failure decreases and the spread between times to failure decrease [1], [3].

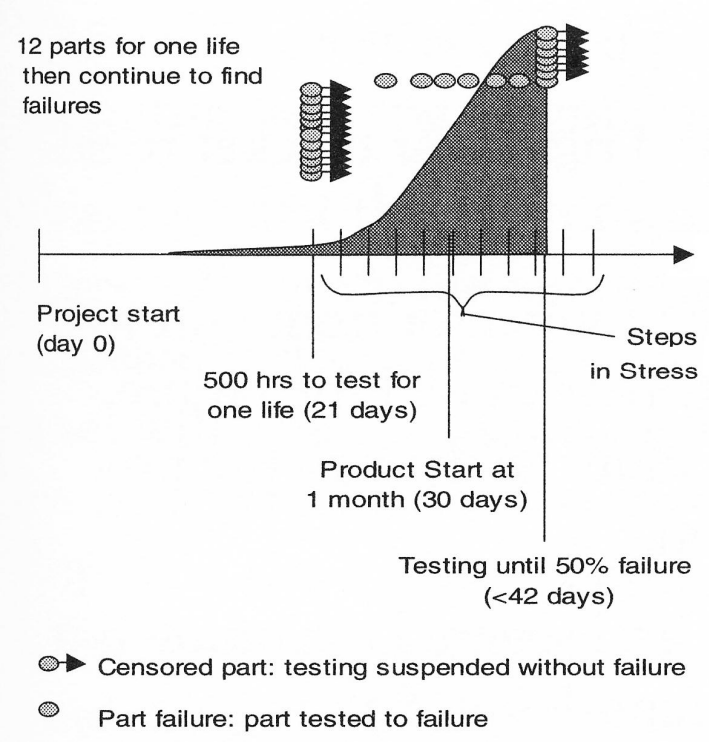

Figure 1: Example of step stress test [1]

2. The proposal of performance testing for ICS

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 result of our proposal is the following sequence diagram (captured in Fig. 2), which captures the steps of automated performance testing in terms of time.

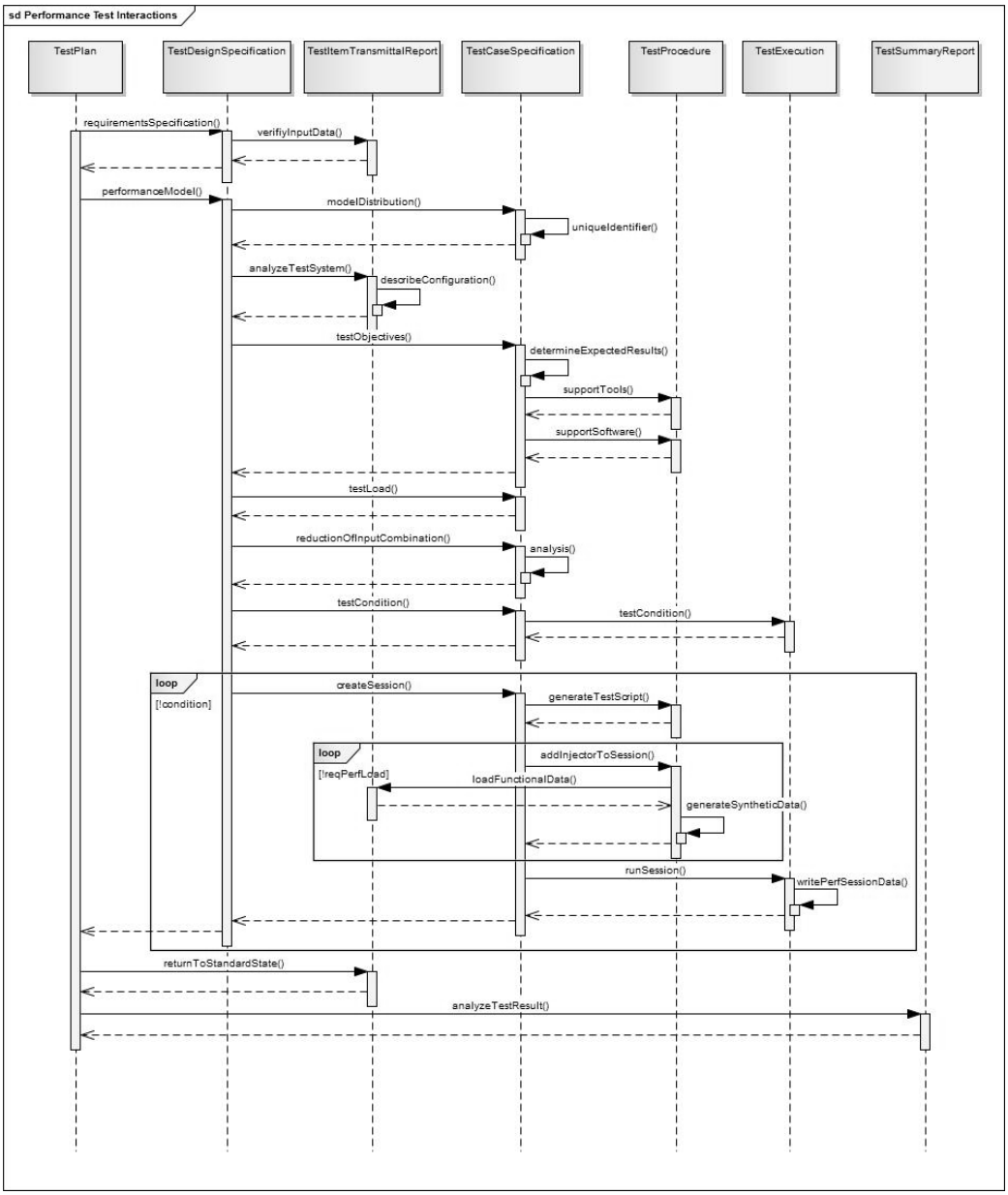

Figure 2: Proposal of performance testing steps

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.

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.

Also the load testing itself, that is used in our proposal (in the individual session), will be modeled in the next iteration of our proposal.

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. 3), 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.

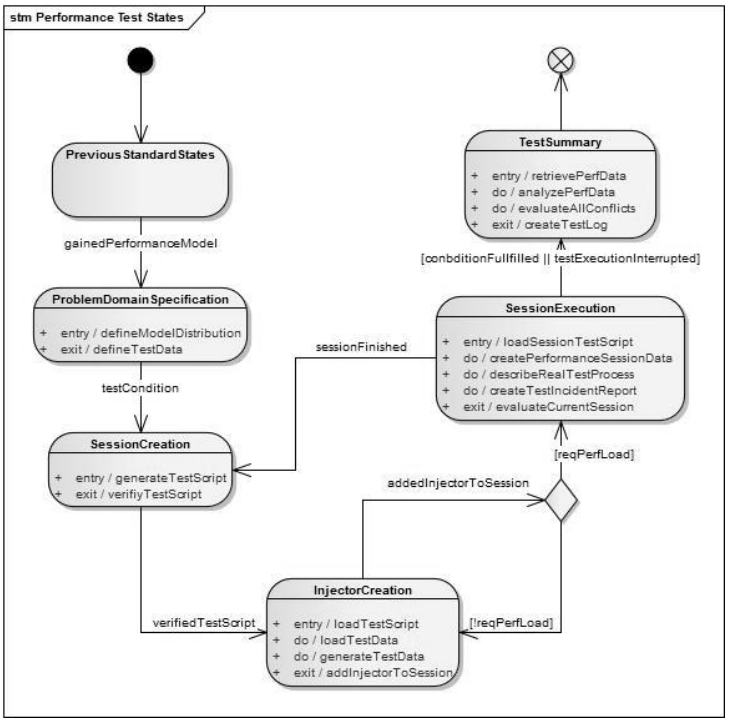

Figure 3: The overview of states in proposed performance testing

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.

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. Proposal of step stress testing for ICS

Design of our automated step stress test model is based on six steps, which are captured in Fig. 4. It should be noted, that this model captures the automated software testing in the phase of regression testing, whereas our model focuses on stress testing. This model served as the basis for our stress testing sequence.

Acceleration of stress testing has been reinforced by using of step stress testing. As captured in Fig 1, it builds upon the original stress testing and adds additional life cycle where stress values are increased by 10% every 10% of the life cycle. This procedure is repeated until it reaches 50% of failed parts.

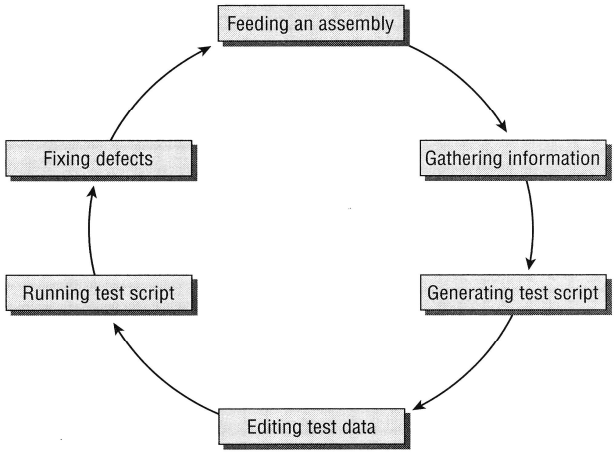

Figure 4: The base six steps of the automated software testing [7]

3.1. Proposal of modified phases of step stress test

The first step of our proposal is captured in Fig. 5 - the model of basic step stress test. This proposal is modeled as an UML Activity diagram with a certain degree of abstraction. It captures the initialization and iterations of step stress test. It should be noted that this diagram includes simple, but also complex activities. The test is brought to the end, if in the iteration are more than 50% of failed parts, or even if there is at least one failed part in a fully censored test.

3.2. Proposal of automated stress testing

The next step is proposal (Fig. 4), which captures the time sequence of the automated step stress testing mapped for each phase of testing methodology IEEE 829. This was slightly modified for greater clearness and simplicity. The phase Test Plan includes a procedure, which describes how the testing will proceed and it is captured by modeled sequence diagram as a whole. In the diagram we use the name Test System that better reflects this phase in terms of process automation. We also merged phases Test Log and Test Incident Report to the Test Execution phase, mainly because of phases modeling difficulty.

After obtaining the necessary information, the test system passes to the phase Test Design Specification, where are defined stress variables, as well as the length of the life cycle and test sample preparation.

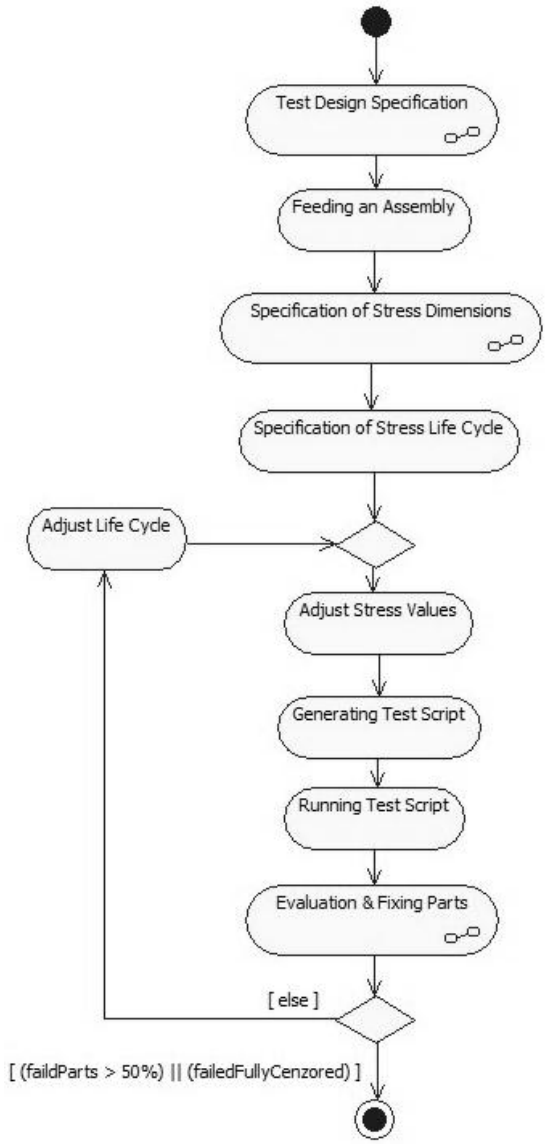

Figure 5: The proposal of modified phases of automated testing

Subsequently, the test system begins performing cycle. The first iteration cycle is fully censored test, in which are the Stress Values initialized to the initial value. Another step is to generate test scripts as well as its subsequent execution. After the execution, the individual results are evaluated. It should be noted, that in the first iteration of the test should not come to the mistakes, in consideration of the fully censored tests character.

Further iterations of step stress testing occur almost identical. They are different in length of the life cycle, which is reduced to 10% of the original fully censored test length. In all following iterations stress values are also increases by 10% of the original value. If there is a failure of individual test parts, they will be corrected and recorded in the Test Report Summary.

The test system will interrupt test proceeding, if more than 50% of the parts fail. In the end, results are evaluated and summarized in the Test Summary Report. The state model

The third step of our proposal is modeling of states together with identification of events that activate transitions (captured in Fig. 6). A set of states is captured as a synchronous sequence. This is mainly due to the complexity and clarity over asynchronous model.

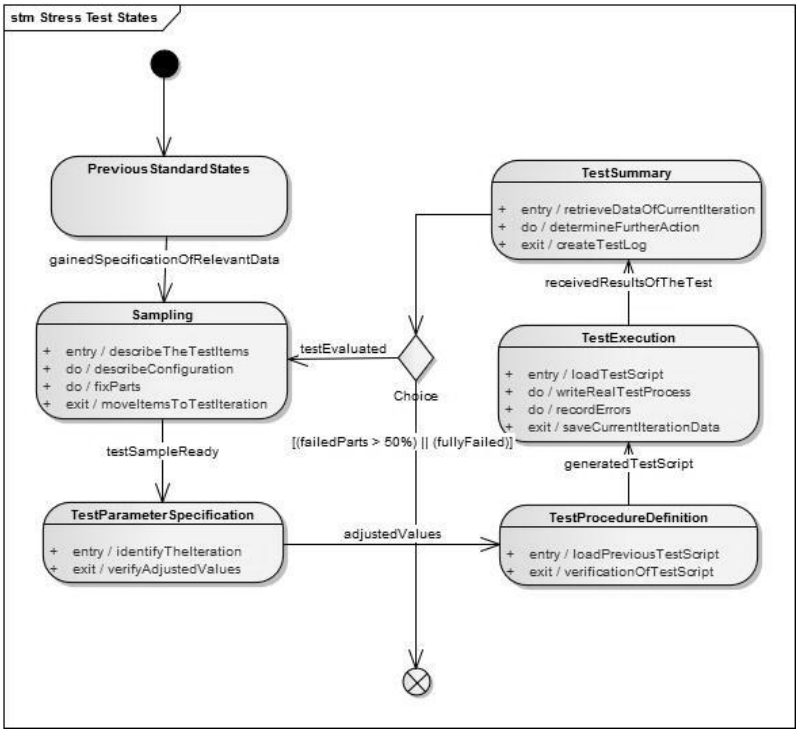

Figure 6: The overview of states in proposed stress testing

4. Conclusion

The aim of this article was to design automation of system testing. In our proposal we have focused on performance and stress testing itself. Our proposal was captured by sequence and state machine diagrams in UML 2.0.

References

[1] PORTER, A. 2004. *Accelerated Testing and Validation*. Burlington, USA : Newnes, 2004. 242 pp. ISBN 0-7506-7653-1.

[2] E. Dustin, J. Rashka, J. Paul, ñAutomated Software Testing: Introduction, Management, and Performance,ò Addison-Wesley Professional, 1999, ISBN 978-0201432879

[3] Isrd Group, ñStructured System Analysis And Designò, Tata McGraw-Hill Education, 2006, p. 432, ISBN 007-0612048

[4] M. Skamla, P. Hamerník, P. Vazan, ñThe evaluation of influence of priority rules in job shop scheduling,ò Process Control 2010: 9th International Conference. Kouty nad Desnou, 7.-10. 6. 2010, University of Pardubice, 2010, ISBN 978-80-7399-951-3

[5] I. Molyneaux, ñThe Art of Application Performance Testingò, OôReilly Media, Sebastopol, CA, 2009, ISBN 978-0-596-52066-3

[6] P. Yunming, X. Mingna, ñLoad Testing for Web Applications,ò The 1st International Conference on Information Science and Engineering, Dec 18th to 20th, 2009, pp. 2954-2957, 2009 ISBN 978-0-7695-3887-7

[7] L. Kanglin, W. Mengqi, ñEffective Software Test Automationò. Alameda, USA : SYBEX, 2004. 408 pp. ISBN 0-7821-4320-2

Authorôs addresses:

Pavol Tanuska, Assoc. prof
Slovak University of Technology in Bratislava, Faculty of Material Science and Technology in Trnava
Paulinska 16
SK-91701 Trnava, Slovakia
pavol.tanuska@stuba.sk

Lukas Spendla, MSc.
Slovak University of Technology in Bratislava, Faculty of Material Science and Technology in Trnava
Paulinska 16
SK-91701 Trnava, Slovakia
lukas.spendla@stuba.sk

Alojz Masar, MSc.
Slovak University of Technology in Bratislava, Faculty of Material Science and Technology in Trnava
Paulinska 16
SK-91701 Trnava, Slovakia
masara@stonline.sk

Ondrej Vlkovic, MSc.
Slovak University of Technology in Bratislava, Faculty of Material Science and Technology in Trnava
Paulinska 16
SK-91701 Trnava, Slovakia
ondrej.vlkovic@stuba.sk