Probabilistic Networks as a Means of Testing Web-Based Applications

Anton Bykau
Department of Informatics
Belarusian State University of Informatics and Radiotechnics
Minsk, Republic of Belarus
anton.bukov@gmail.com

Abstract— **The article describes the mechanism used to control GUI tests coverage and the technique of GUI application under test model building using probabilistic networks. The technology of combining GUI tests into the common network has been developed. The mechanism to report defects is proposed.**

Keywords— probabilistic network testing; web interfaces; automation

I. Introduction

Testing is a process of execution of the program to detect defects [1]. The generally accepted methodology for the iterative software development Rational Unified Process presupposes the performance of a complete test on each iteration of development. The testing process of not only new but also earlier code written during the previous iterations of development, is called regression testing. It's advisable to use the automated tools when performing this type of testing to simplify the tester work. "Automation is a set of measures aimed at increasing the productivity of human labor by replacing part of this work, the work of machines". [2] The process of automation of software testing becomes part of the testing process.

The requirements formulation process is the most important process for software developed. The V-Model is a convenient model for information systems developing. It's become government and defense projects standard in Germany. [3] The basic principle of V-model is that the task of testing the application that is being developed should be in correspondence with each stage of application development and refinement of the requirements. One of the development model challenges is the system and acceptance testing. Typically, this type of testing is performed according to the black box strategy and is difficult for automation because automated tests have to use the application interface rather than API.

"Capture and replay" is the one of the most widely used technologies for web application test automation according to the black box strategies today [4]. In accordance with this technology the testing tool records the user's actions in the internal language and generates automated tests.

Practice shows that the development of automated tests is most effective if it is carried out using modern methods of software development: it is necessary to analyze the quality of the code, merge into the library the duplicate code of tests, which must be documented and tested. All this requires a significant investment of time and the tester should have the skills of the developer.

Thus, the question arises of how to combine the user actions recording technology and the manually automated tests development, how to organize the automated tests verification, and whether it is possible to develop an application and automated tests in parallel according to the methodology of the test-driven development (TDD).

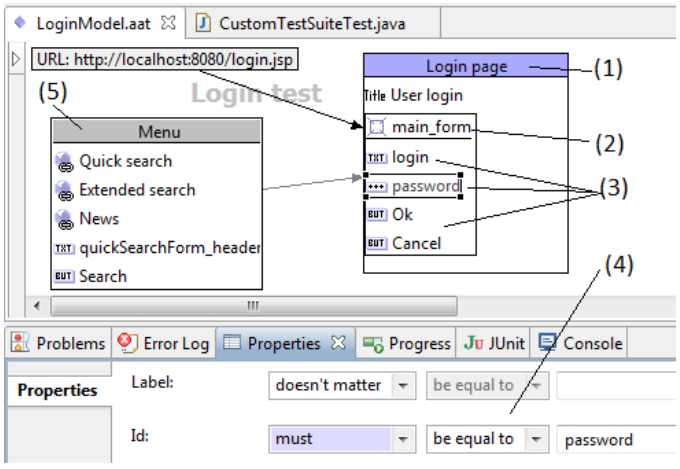
There are systems capable of determining the set of tests that must be performed first. Such systems offer manually associate automated tests with the changes in the source files of application under test. However, the connection between the source and the tests can be expressed in terms of conditional probabilities. The probabilistic networks used in the artificial intelligence, could also be useful when defining the relations automatically based on the statistics of tests results. By using probabilistic networks we can link interface operations and test data and this will allow reducing the complexity of automation.

II. KEY ELEMENTS OF PROPOSED TESTING TECHNOLOGY

For tests automation we could use a probabilistic network that has the following structure:

The first level network shown in Fig. 1, consists of two layers, which determine the location of graphical controls on the web page. Top-level nodes Fig. 1.1 are either pages or the condition of the tested application page such as a page of the user authentication. Lower-level units are templates used to identify GUI elements Fig. 1.3. Some nodes are GUI container templates Fig. 1.3. Fig. 1.4 shows the properties of the selected node, like the template for the password field. Graphic elements that occur more than on one page can be transferred to a general unit for multiple pages, such as Fig. 1.5 that shows the menu items. Fig. 1 shows only the network connection between the unit and the common elements of the page to simplify the visualization of the network for testers.

The availability of GUI templates and states of the web interface allows monitoring the test coverage for interface of application with tests; it also allows to effectively adapt automated tests to new versions of the tested application.



Properties			
Label:	doesn't matter	be equal to	
Id:	must	be equal to	password

Fig. 1 GUI elements composition

The main goal of the second level network is to describe the workflow of the program in the form of interconnected rules, describing the program states and GUI interface actions (see Fig. 2). The network consists of two layers and two types of nodes that include the nodes of all possible states of the program (see Fig. 2.1) and the nodes of all possible program actions (see Fig. 2.2). The communication network describes the state transitions as a result of GUI activities. The page can be linked to the data (see Fig. 2.3) to describe the state of the page containing dynamic elements, for example, a table with a date. The data layer consists of nodes storing the state of the tested application and the operations that modify the data. Fig. 2.3 describes the results table which is used in Fig. 2.4. Each table row should include a reference to additional information; the lower part of the table should contain additional 3 references (see Fig. 2.4) while the search box should include the search phrase (see Fig. 2.5). The state of some graphical elements is not preserved in the data layer (Fig. 2.6) to simplify the automation process.

The system of tests automation constantly analyzes the state of the application interface during the tests recording time. If the same sequence of actions is repeated many times, the system offers to merge this sequence for multiple pages into a common block (see Fig. 1.5). The recorded actions and states will not be duplicated. When writing the second and subsequent tests, the system adds only unknown conditions and operations. Although the model interface can be split into separate files, it will not prevent the system from linking blocks common for several pages. Often, automated tests complicate the process of automation as a result of an unsuccessful candidate decomposition code. A single model of the whole test interface can help to avoid duplication and to refactor the source of recorded tests.

The system determines an appropriate relationship between the states if a previously unknown combination of actions was done between the known conditions in the process of test recording.

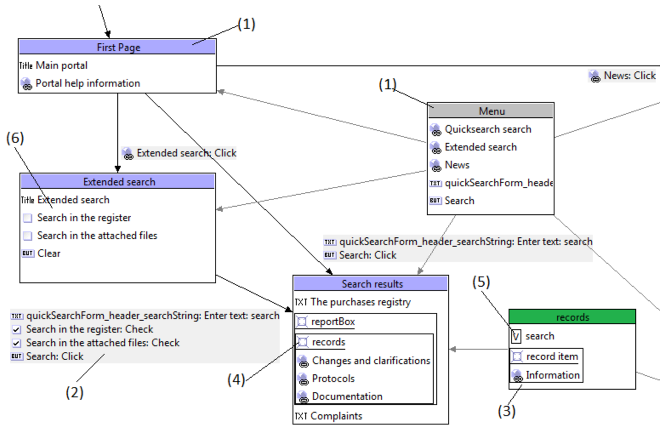


Fig. 2 Program Algorithm

The third level network describes the tests and defects of the tested program. The top layer describes a set of written tests (see Fig. 3.1) and is connected to the nodes pages (see Fig. 3.2). Each test case describes what action and what graphics should be checked (Fig. 3.3). Subsequently, the system will find preliminary steps for testing, using an algorithm to find a way to graph states proposed by S. Russell [5] to perform one or more tests.

The relationship between the test and page nodes can be divided by a bug note to describe the defect (see Fig. 3.4). The defect can be in one of the following states turning a positive test into a negative one (see Fig. 3.5):

- presence of an undocumented and uncorrected defect (the node is absent)
- expectance of an uncorrected and described defect (the defect node created and verify defect reproduce)
- absence of the expected defect (the defect node can't reproduce the defect)
- confirmed lack of the described defect (the defect note verifies the defect absence)

The test system displays test results in a different way for developers and testers. This allows evaluating the correctness of the automated tests and independently assessing the quality of the tested application. The presence of the life cycle of a defect integrates accounting system defects and automated testing.

The priority value is associated with each test node. This characteristic is actually the probability that the test result will be incorrect, for example, the bug will not be reproduced or the expected page will not load properly. The higher the probability of the failure, the more important it is to run the test to fix the problem and increase the stability of testing.

The priority of the test run can be set manually by the tester, or can be statistically calculated on the basis of the associated defect status changes, or the associated source code changes, or on the basis of the results of the same test for the same controls of other pages. Typically, these tests are associated with blocks of common elements (see Fig. 3.6).

The most important testing task is to measure the relatedness of the test results from the internal state to the application, or previous operation. The main problem of such measurements is an extremely large number of conditions with should be measured by the test system. The whole history of the automated testing system is preserved, and each performed activity is associated with a corresponding network node .

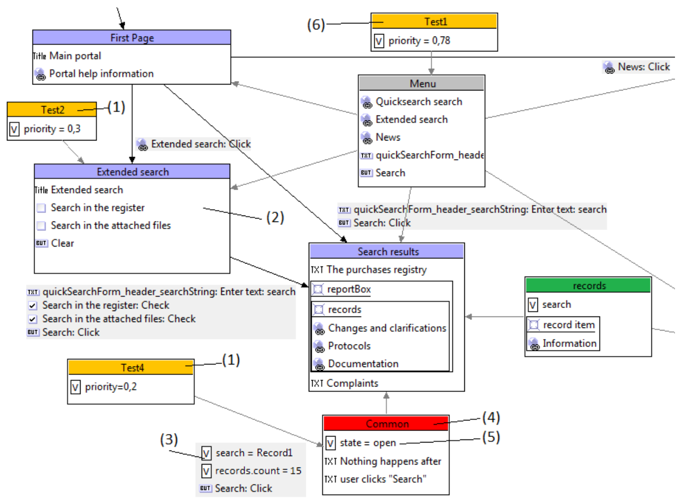


Fig. 3 Description of Tests and Defects

The fourth level network describes the knowledge about of testing purposes (see Fig. 4). The network consists of the nodes which represent the testing goal (see Fig. 4.1) and is associated with one or more tests (see Fig. 4.2). The example of the target can either be one or a group of pages and of the tested interface program (see Fig. 4.3).

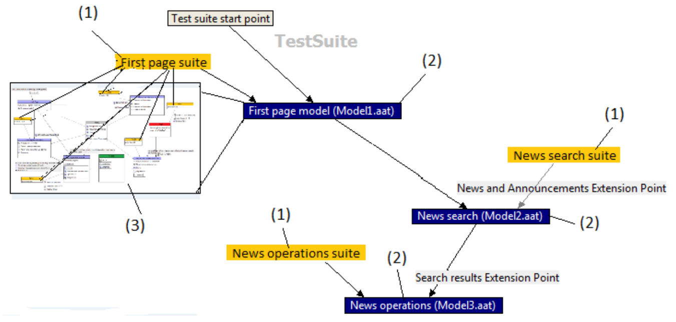


Fig. 4 Description of Test Purposes

Two algorithms are used for the network work; they are the calculation network algorithm and the path finding algorithm. The calculation algorithm determines the status of the tested application using patterns of GUI elements, and calculates the priority of tests running, analyzing what associated source files have been changed and what defects have been fixed. The path finding algorithm finds the sequence of preparatory steps to perform the test in order to select a sequence of tests that will allow to reduce the total test time.

III. NETWORKS CALCULATION ALHORITHM

The test system uses a modification of the Bayesian networks calculation algorithm proposed by R. Schechter [6]. The modified algorithm can calculate the network even in the presence of the following features:

- Probabilistic network links can be directed or undirected.
- Probabilistic network links can contradict each other.

The first level network must be recalculated, despite the controversy because the program interface can be wrong: the graphic elements may not work properly, requirements may be outdated or the tester can make mistakes. The goal of the test system is to detect these mistakes.

Probabilistic networks nodes can take multiple values which are characterized by probabilities. The probability evaluate whether the node actually takes this particular probability value. The condition corresponding to the node, its condition is called a characteristic. The sum of all characteristics of the multivalue node equals 1.

$$P(A1)+P(A2)+\ldots+P(An)=1 \qquad (1)$$

The network connection may be contradictory. Contradictions arise when there is a problem in the test program. The algorithm has to consider the mutual influence of links and to make approximation solutions. On the other hand, the system can independently adjust its work in case of the loss of control of the tested application.

To describe the algorithm we shall present an example of calculating the characteristics of the two states of simple networks. For simplicity, we use only the connections between two nodes while the binary characteristics and the conditional probabilities equal 1 or 0. We shall use Bayes' formula to calculate the characteristic of the required node:

$$P(A)=P(A|B)*P(B) \qquad (2)$$

Let's consider an example where the communication is in conflict. Let's suppose that we know that:

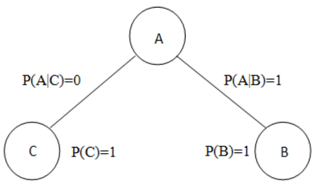


Figure 5. Contradictory Conditions

When looking at Figure 5 we can consider connections C-A and B-A independent, and the probability node A is calculated as the probability of two independent events:

$$P(A)=(P(A|B)*P(B)+P(A|C)*P(C))/2 \qquad (3)$$

Another difficulty is the presence of cycles in the network. Let's add to the previously described structure of the network Figure 5 connection C-B, and calculate the values of the characteristics B and C on the basis of the given vertex A

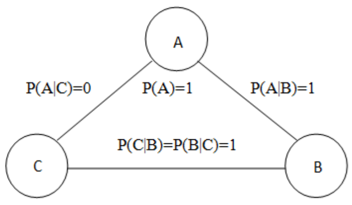


Figure 6. Contradictory Dependencies

When looking at the network (Figure 6) we can see an apparent contradiction: the links from node A assign different states to nodes B and C but the link C - B requires the identity of node values.

We could solve the contradiction by reducing the trust in relations of the network but we can't do that until we know the correct values. The temporary solution should be the construction of the set of the skeletons of trees of a network for any given performance with equal confidence in relations and the known value of the node A. There are three skeletons for the network (see Figure 6). It's easy to calculate the probability value of the nodes for each such skeleton. Finally we find the average value for each characteristic for each skeleton tree. The solution can be presented in the following way:

$$P(C=1)=P(B=0)=0{,}333,\ P(C=0)=P(B=1)=0{,}667 \qquad (4)$$

The advantage of the algorithm is that the connection can combine more than two characteristics and the logic of the relationships conversions can be defined by the programmer manually. The link may be represented as a function of several variables that return the value to the node to which it is directed and that can be defined in any programming language. The presence of a double direction link between the two characteristics can be described by two oppositely oriented links.

IV. AUTOMATION PROCESS

The probabilistic network for the application testing can be created on the basis of the "record and play" tool. This method is useful when the testing system has a poor knowledge of the tested application. When recording the test system stores the sequence of the application states and interface actions. After the recording of the test the test automation system invites the tester to answer some questions. The recorded net diagram of transitions between the states should become the result of the recording.

The tester creates a test node and describes the data need for the test to define the test case. He can create a set of tolerance values for each GUI element of the page (see Fig. 2.3). In this case, it will reach the coverage criterion according of the black box strategy "covering the tolerance range", based on the testing criteria of the class of input and output data.

The network for the application testing can be created using the answers to the questions about the interface. This interface is effective when the model contains enough knowledge about the tested program. The system will be testing the application in the background, and if there is a problem, it will ask the tester without stopping the execution of other tests.

The system operation and the work of the tester start with some initial page and state of the tested application. This condition is evaluated and if the condition does not correspond to GUI templates, the system will suggest that we add a new state to the model. To facilitate the dialogue with the user all the questions are simply reduced to the confirmation of the changes, or, in case of an error, the choice of the right solution. For example, if the test system reliably determines all the basic controls, it prompts you just to confirm a page layout. Next, the system selects the highest priority operation for testing, then performs it, and analyzes the next state. In case of conflict such as some unexpected behavior or the appearance of the tested application the system will propose to create a characteristic describing the defect.

CONCLUSION

The technology of the test automation using probabilistic networks uses generic templates of interface graphics to conduct the analysis of the interface test program which allows to carry out the testing of the applications based on the "black box" criterion by covering the tolerance range on the basis of the testing criteria of the classes of input and output data.

The developed measures allowed to vary the order of the execution of tests for related modules, analyzing the test results for the current or previous versions of the application and can serve as a new measure to evaluate the relation between the test results and various modules of the program for its overall functionality.

The mechanism of defects detection, designed and tested by the author, can be used to evaluate the correctness of the automated testing work and independently assess the quality of the tested application.

This technology has been tested in the project WebCP by automation Ajax interface testing and has shown its effectiveness and convenience in comparison with the development of GUI Unit Tests writing.

The author thanks his scientific adviser I. Piletski for his help in preparing this paper.

REFERENCES

[1] G. J. Myers, The Art of Software Testing, John Wiley & Sons, Inc., New Jersey, 2004.

[2] I. Vinnichenko Automation of testing processes, Peter Press, C-Piterburg, 2005.

[3] The V-Model as Software Development Standard; IABG Information Technology

[4] Martin Steinegger and Hannu-Daniel Goiss Introducting a Model-based Automated Test Script Generator - Testing Expirience Magazine. pp.70-75

[5] S. Russell, P. Norvig, Artificial intelligence: a modern approach (AIMA), Williams, Moscow, 2007.

[6] R. Shachter, Evaluating influence diagrams. Operations Research, 34 (1986), 871–882.