

Case Study on Testing of Web-Based Application: Del's Students Information System

Arnaldo Marulitua Sinaga
Del Institute of Technology, North Sumatera, Indonesia.
aldo@del.ac.id

Abstract—Software Testing is an important process to assure the quality of software including web-based application. The aim of testing is to detect all failures to ensure that the software is built in accordance with its specifications. There are two methods of testing, such as the white box and the black box testing. White box testing is done with the approach to the program code, while black box testing is done without referring to the source code, but to the output of the resulting program. User interface testing is the type of testing that is usually conducted on a web-based application. In this research, functional and user interface testing were conducted into Del's Student Information System. Testing techniques used in this research was a Simple Functional Acceptance Test (FAST). This technique was done by testing each function and checking each component of the user interface. Selenium IDE was used to execute test cases resulting in the implementation. From all steps carried out in this research, it can be concluded that the functional testing can be carried out in line with the testing of the user interface. A testing scenario for web-based application has been proposed and experimentally applied into an application under testing.

Index Terms—Web Testing; Functional Testing; UI Testing; Functional Acceptance Simple Test (FAST).

I. INTRODUCTION

Nowadays, the growth of Web-based applications development is increasing remarkably [1]. Web-based applications have many advantages compared to desktop-based applications, such as the accessibility of web-based applications is broader through the availability of network connection. This phenomenon has also increased the complexity of the requirements of web-based applications [2]. Therefore, a web-based application developer needs to develop a web-based application using a systematic approach in order to obtain a good quality web-based application [1]. A typical operation of a web component generates HTML pages by writing HTML content in the tags [3]. These pages are returned to the user and displayed through a browser [4]. Along with the increase of the use and development of web-based applications, the need for quality assurance of web-based applications is also increasing. Therefore, software testing becomes an important aspect in the development of an application.

Software testing is an activity that is conducted to ensure that the software is built in accordance with the specifications, in which all potential failures have been detected and the corresponding faults have been fixed [4, 5]. It is intended to detect all failures in the software [6]. Failure is indicated when the actual output is different from the expected one. In software testing, test scenario is needed to identify the flow of test case execution process on the

application. Scenarios are stories that work through complex problems in testing that is defined hypothetically [7]. One important activity in a testing process is to design test cases [8]. Test case consists of the values of the input (test data), the expected output (produced by the test oracle), and the purpose of testing [8]. Software testing requires test cases that can be generated either manually or automatically.

This research was conducted using a case study on web-based applications testing. The types of testing applied were functional and user interface testing. The performed testing method was applied into an application that is still in the development stage. Applications used as the object under test are Del's Dormitory and Student Information System. This application was used because this application was on the latest stage of implementation and has never been tested yet.

The purpose of the implementation of this research is to propose a test scenario that can be used on a web-based application testing and the proposed method is experimentally examined by applying it to the object under testing. In addition, this research will also prove whether functional testing and user interface testing can be performed simultaneously on a web-based application testing.

II. WEB TESTING

Web testing is a type of software testing that focuses on web application software specification [2]. Web-based application is different from other traditional software in the process of operation because the web application is heterogeneous and autonomous [1]. Thus, the web server must have the ability to interact with users and handle transactions. It can be concluded that the web testing is more difficult than traditional software testing for web application with a more sophisticated functionality, in terms of publications, search and indexing [1]. By executing the test scenario, the conduct the web testing is to test that the web application has been run in accordance with the predetermined specifications.

In the book *Testing Applications on the Web* [2], it is explained that there are several types of testing frequently implemented into the web application [2], as follows:

- User interface testing: testing the web application interface to ensure that the web interface of the application is in accordance with predetermined specifications.
- Functional testing: testing the function that is executed when the web application operates to ensure that all functional requirements can be run based on specifications.

- Testing Database: testing the validity of data and data integrity to ensure that data are not corrupted and implemented as data design.
- Installation Testing: testing installation and Uninstallation procedures of the program in the server and client side to ensure that the installation of the application can be run properly.
- Configuration and compatibility testing: testing on the configuration of the operating system, web browser, system hardware and supporting software to detect functional errors in the application.
- Performance, load and stress testing: testing on the performance of web applications by measuring how long the data processing system respond to request and meet the requirement.

In this research, functional and user interface testing were carried out on Del's Dormitory and Student Information System.

Functional testing technique with simple acceptance test (FAST) is a technique that runs the smallest function for every command from a program. This testing technique is done to ensure that every input and navigation controls web application can run as had been expected. The following are the features often tested in testing techniques FAST:

- Links, such as the contents of the selected links, thumbnail links, link bitmap and image links
- Basic controls, such as forward and backward navigation, zoom in and zoom out, other UI controls, and commands such as add, modify, and delete objects.
- Other important features such as log in, log out, email notifications, search, and handling forgotten passwords.

Some errors may be found in the testing process are as follows:

- The link damage
- The image missing
- The link fault on the link
- The picture error
- Link is true, but the contents of the link has been expired
- An error in sorting and calculation of the transaction
- Content and writing automated email improper retaliation
- No response from the server
- Inability to validate a user's email address

The development of Internet technology expands the use of the web user interface. A Web User Interface (WUI) is a GUI in a hierarchical structure that contains frames and pages, with geometric and temporal constraints between pages [9]. WUI provides a graphical front-end for a software consisting of multiple programs, the application of different languages, executed on multiple platforms simultaneously, and all are connected via the Internet [9].

Just like GUI, input in WUI occurs in the form of events and graphic. WUI has all the characteristics of the GUI, which include event driven inputs that change the status of WUI, graphical output, hierarchical structure and graphic objects with their property. There are three approaches in WUI testing: The automated, semi-automated and manual approach [9].

Automated approach makes a clone of a web browser to generate requests. The response of each request is analyzed

and it determines the truth in the context of a single request. Disadvantage of this approach is the lack of perspective of the tester of user interaction on a web-based application and collection of effects that occur in any sequence of events. Semi-Automated approach tests user interaction that designer noted in the WUI, edits captured script to generate a visible difference in test cases and executes automatically the WUI. But the capture/replay tools have limitations to examine the output. Overall coverage of test cases depends on the interaction of the test designer with the WUI. Manual approach generates more realistic test cases. Human tester interacts directly with the WUI. In testing the WUI, human tester tries to find errors.

There are several types of user interface elements on web-based applications that are often tested in testing the user interface, as follows [2]:

- Instructional and technical information - the accuracy of information and instruction.
- Fonts - consistency font style, text legibility, the resulting visual clutter of font faces in a document as well as the availability of font faces at the targeted platform.
- Color - The color of the background, foreground and font, the improper color can cause negative effects and confusion.
- Border - a three-dimensional effect in the command button can be effective visual requirement for the user.
- Image - suitability background, label legibility, button legibility, the suitability of the size of the images.
- Frame - some earlier browsers cannot display frames and settings.
- Table - the nested tables (tables within tables) can affect the appearance of the browser that depends on the display settings and browser type.

There are some conventional software testing stages for the GUI, such as [9]:

1. Determine the object to be tested. In this phase, the tester uses the coverage criteria to determine the object to be tested. Coverage criteria is a set of rules used to determine the object to be tested.
2. Generate test input. In this phase, the tester generates test inputs for testing. Test input can consist of events such as mouse clicks, menu selection and input from the keyboard.
3. Generate expected output. After determining the test input, test oracle generates expected output. Test oracle is used to check whether the software runs well at the time of testing. Test oracle is the mechanism to determine that the actual output of the software together with the expected output. In the GUI, expected outputs include a screen snapshot, the position and the title of the windows.
4. Execute test cases and verify output. At this stage, test case execution on the software is being tested. Then, the actual output is compared with the expected output that is obtained from a test oracle.
5. Ensure that the GUI is adequately tested. After all the test cases are executed on the applied software, the software is analyzed to examine the parts tested. In the GUI, the analysis is needed to determine the status of all events that related to that GUI.

After testing, if there are any failures in the software, it will be fixed or modified. Modifications to the software being tested, lead to regression testing. Regression testing is the re-execution of all or some subsets of test suite [9].

III. METHODOLOGY

A. Case Study

In this research, Del's Dormitory and Student Information System were used as the application under test. This system consisted of 34 web pages for 13 main functions [10]. Each function had criteria that were used as the basis to generate test cases. The total amount of criteria for all functions was 71. In order to avoid oracle problem, mutants were manually generated. Mutants were generated by changing the code in a file in the application, so that it would change the behavior of a function in the application. The changed files (hereinafter referred to mutant files) were stored in the directory "Mutants", in accordance with the functions and criteria so that the mutant was only valid for one criteria in 1 function.

B. Tools

In this experiment, Selenium IDE was the tool to apply testing. Selenium IDE is a tool that can be used to test the functionality and user interface on web-based applications [11]. In addition to Selenium IDE, this experiment also used a simple comparison program called FileComparing.java created in the experiment with the aim to facilitate the tester in testing a web-based application by comparing the actual and expected output.

FileComparing.java is the source code that was created with the Java programming language and served to help the tester find a failure during testing. This program would examine 2 pieces of output files in HTML format with the same functions and criteria. Furthermore, the program would produce a report if there was a failure on the test or not. If failure was detected, then there was existing fault that need to be fixed. Determining the detected failure in more detail to the level GUI properties, can be seen in Selenium Log. This program would only stop if the matching process has been completed and no failure found or when any failure was detected in the first time. This was in accordance with the principle of the test itself, which directly fix a fault that produce detected failure before detecting another failure. It was possible that a failure was the result of another failure. Comparisons were made into two types of program output, namely the expected and the actual output file. For the test with the mutant, there should be at least one failure detected, since there was a fault in the mutant. The testing was correct if the execution of mutant resulting one or more failure/s, otherwise the testing was not valid. This was also to prove the quality of generated test cases to "kill" the mutants. The program was scanned based on the directory of "Output Test Case", "Oracle Scripts", and "Output Mutants" which have the same sub-directory hierarchy (Functions and Criteria). Comparisons would be made starting from the first function of the directory and the first criterion to all criteria (m) and function/s (n), where m and n were greater or equal to 1. The value of m and n were obtained from the argument when the program was being executed.

C. Experiment Process

The steps of the experiment are as follows:

1. Make the testing criteria for each function tested on object under test.
2. Create an oracle GUI and scripts using Selenium tool.
3. Create a test case for each criterion by using services provided by Selenium IDE.
4. Create mutant manually by injecting a fault into the program.
5. Execute the test cases and mutants that have been created using Selenium tool.
6. Save the output resulting from the execution of the test case.
7. Compare the expected and actual result by using a program FileComparing.java.
8. If a failure is detected, the program checks the Selenium Log, then go back to step c.
9. If the failure is not found, the test will proceed to another test case.

The implementation phase of the experiment was divided into four parts, namely: the creation of criteria for each function, test case using Selenium, execution of test cases using Selenium and checking whether the test success or fail. Examination of testing was performed by the program FileComparing.java and Selenium tool automatically.

IV. RESULTS AND DISCUSSION

A. Experiment results

The experiment on testing object under test was done by executing the test cases for functional testing and user interface simultaneously. In the implementation of this experiment, there were 11 functions being tested along with each position type of user interface elements on each of output pages. There were seven user interface components involved in this experiment: image, text, link, input text, input password, input checkbox, and input submit. All oracles for each of test cases have been generated by using Selenium IDE. These oracles were used as the expected output.

Figure 1 shows all the executed commands. Line/s with green background indicate/s that the output was identical with the expected output, hence the execution of the corresponding command has not detected any failures. Meanwhile, failure detection was indicated by the command (line) with red background. As shown in Figure 1, all executed commands produce the same output with the expected one, hence the executed test case did not detect any failures.

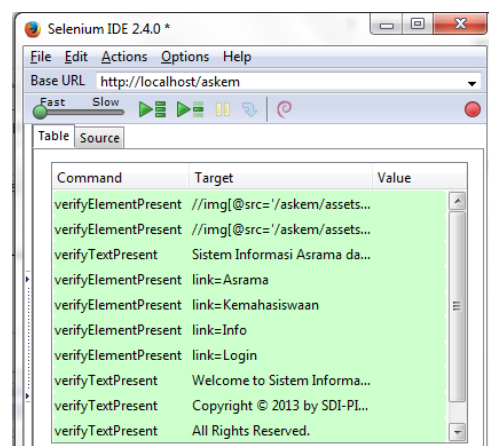


Figure 1 Execution result without any failure

Testing log of the execution of test case in Figure 1 is shown in Figure 2. It contains detailed information about the test results.

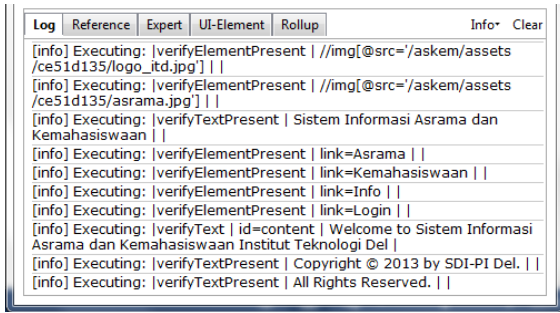


Figure 2: Testing log without failure

The result of the execution of the same test case with the one in Figure 1 into a mutant is presented in Figure 3.

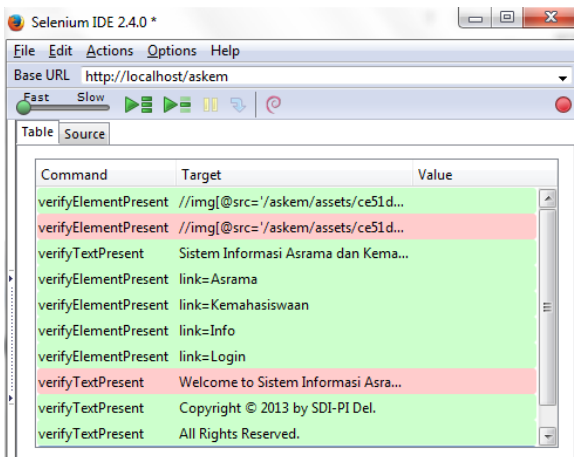


Figure 3: Execution result with two failures

The use of mutant contains two faults. The first fault was injected by changing the source image as indicated in the second line of Figure 1. The second fault was injected by deleting the title text as in line eight of Figure 1. These two faults have been discovered properly. They are indicated by the red background of line 2 and 8 in Figure 3. Therefore, test case has successfully detected the failure.

Testing log of the execution of test case in Figure 3 is shown in Figure 4. It contains detailed information about the test results. The information regarding line 2 and 8 shows that the faults on that two lines have been discovered.

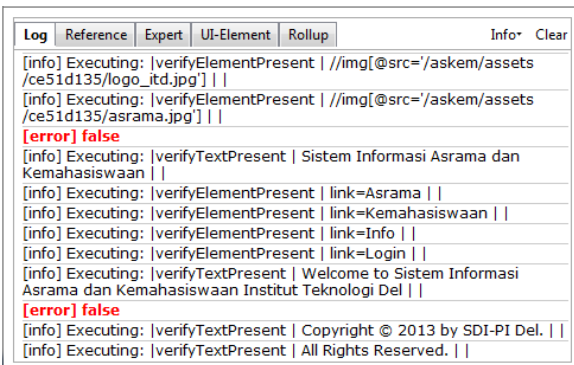


Figure 4: Testing log with two failures

B. Discussion

Based on the results of the experiment, it has been found that the proposed method of web-based application testing has run as expected. The implementation of WUI/GUI approach was able to ensure the validity of the property values of GUI elements on each page. This was indicated by the execution of test cases, in which the obtained actual output was in accordance with the expected output that has been defined in the oracle before the testing execution. The results that were shown on the Selenium IDE tables indicate that the testing has been running as expected. The green highlighted lines on the table indicate the “pass” execution (no failure detection), whereas red highlighted lines on the table indicate the “fail” execution (failure detection).

From all the test cases execution into original and mutants of object under test, it was found that the testing has been running as expected. All injected faults can be detected properly. The example of the execution presented in previous section has indicated this finding.

Through this experiment, it was found that the functional testing with validation approach along with the user interface testing run properly. The functional testing is conducted by using the properties’ value of each element on the web page. However, in this experiment, the test on externally generated files cannot be processed yet.

Each of the possibilities criteria established by using naming specification for each criterion. For example, the functions of authentication are the criteria FieldUsernameNPasswordBlank. It indicates criteria for blank value of field username and password. All criteria have been implemented in the experiments and they are all working as expected.

V. CONCLUSION

The defined test scenario in this research has been implemented into a web-based application, Del’s Dormitory and Student Information Systems. Functional testing on web-based applications has been conducted simultaneously with the testing of the user interface of the web-based application. The experiment has proved that the proposed testing has been running properly. All injected faults have been discovered. The used tools provide information regarding this result.

However, this research is needed to be extended by improving test cases generation. Involving criteria and coverage information may improve the capability of test suite. This research that applied manually generated mutant, hence become a threat to the validity of the experiment. Therefore, it is important to apply real faults. The more reliable test instrumentation should improve the validity of the experiment. Applying more objects under test is also needed in future research.

ACKNOWLEDGEMENT

We are grateful for Del Institute of Technology support to this research.

REFERENCES

- [1] Jiang Jixiang, Xu Baowen, and Xu Lei. 2005. Testing Web Applications Focusing on Their Specialties
- [2] Nguyen, H. Q. 2001. *Testing Applications on the Web*. Robert Ipsen. John Wiley & Sons.

- [3] Halfond G.J., William. 2008. Web Application Modeling for Testing and Analysis.
- [4] Pressman, Roger S. 2001. Software Engineering a Practitioner's Approach.
- [5] Gelprin D. and Hetzel B. 1988. The Growth of Software Testing. *Communication ACM*. 31(6): 687-695.
- [6] Myers G. 1979. *The Art of Software Testing 2nd Edition*. John Wiley & Sons
- [7] Cem Kaner. An introduction to scenario testing, 2003. <http://www.kaner.com/pdfs/ScenarioIntroVer4.pdf>.
- [8] Lee Copeland. 2003. A Practitioner's Guide to Software Test Design. Artech House, Inc., Norwood, MA, USA.
- [9] Memon, Atif M. 2001. A Comprehensive Framework For Testing Graphical User Interfaces.
- [10] ITD-SDI. 2013. Business Requirement Specification of Del's Dormitory and Student Information System
- [11] Selenium Documentation, <http://docs.seleniumhq.org/docs/>, accessed on 17 December 2013