Implementation of The Equivalent Partitioning Method in Testing for Automatic Test Case Generation on The Digi-OTA System

Silfi Nur Amalia*, Sri Widowati, Donni Richasdy

School of Computing, Informatics Study Program, Telkom University, Bandung, Indonesia
Email: *silfinformatika@student.telkomuniversity.ac.id, ^sriwidowati@telkomuniversity.ac.id, ^donnr@telkomuniversity.ac.id

Abstract—Software testing is one of the important phases in determining software quality. In the software development cycle, the testing phase takes more than 50% of the development time. The process of creating test cases in software testing is the most difficult process and determines the success of the testing phase. Test cases for software testing can be created based on the existing analytical modeling in the software specifications. This kind of testing technique is known as model-based testing, which is one of the black box testing approaches. In this study, the analytical model used is the UML Activity diagram. The reason for choosing UML Activity diagrams is because this diagram can model activities in software based on behaviors and conditions that are by the sequence. The output of this research is a prototype of a test case generator using an activity diagram. The analysis of the suitability of the test cases generated for the Digi-OTA application using the equivalence partitioning method is 100% when tested with valid data test specifications, while when tested with invalid test data specifications it produces 100% for approving actors, 95.98% for employee actors and 95.45% for detail officer actor.

Keywords: Software Testing; Test Case Generation; Activity Diagram; Depth First Search; Equivalence Partitioning

1. INTRODUCTION

Software testing is an important process in software development that aims to ensure the suitability of the software to its needs that lead to its development [1]. In the process, software testing requires more than half the time and cost of software development [2]. Before carrying out the testing phase, testing planning is carried out in advance, which includes making test cases in software testing. Test case creation is one of the most difficult processes in testing, therefore test case creation takes time [2]. Test cases can be generated manually by writing test cases or automatically [3] using several approaches in determining the test sequence, namely test cases generated based on program source code and test cases generated from requirements and design specifications.

The approach using requirements specification and design has been widely used by researchers in recent years which has the advantage of allowing test cases to be available early in software development, thus making planning more efficient and effective [4]. The approach to user requirements specifications is better known as black box testing. Black box testing is a testing method where the product is tested according to software specifications or requirements [5]. One of the implementations of black box testing is model-based testing[6]. Model-based testing is a software testing technique with a System Under Test (SUT)[7] which is checked for conformity with the predictions made by the model. Model-based testing has several advantages including improving the quality of testing, reducing costs and testing time [8]. There are several models used, one of which is the Unified Modeling Language (UML)[9] which includes a set of graphical notation techniques to create visual models that can describe the behavior of very complex systems and can assist in making proper documentation of software and maintaining consistency between specifications and designs, document.

Unified Modeling Language such as activity diagrams, use case diagrams, sequence diagrams, and others. In the final project, the model that will be used is the activity diagram model because it has dynamic properties owned by the model. The dynamic aspects of objects can be constructed, visualized, specified, and documented by activity diagrams. Activity diagrams can be modeled by focusing on the sequence of behaviors and conditions to organize a sequence of events [2]. There is one method that can produce a series of tests that are more efficient in increasing complexity according to priorities [10], namely the method proposed by Tiwari et al that can generate a sequence of test cases using activity diagrams that have been converted from use case diagrams based on actor/user[11]. Therefore, to improve the quality of testing, reduce costs and time in testing, at this research will build a prototype that is able to generate test cases based on activity diagrams by using the Depth First Search (DFS) algorithm [12] by testing a case study, namely the digi OTA website. The results of the test cases obtained will be retested to ensure the suitability between the test cases obtained generated by the test results using the equivalence partitioning method.

2. RESEARCH METHODOLOGY

2.1 Research Stages

The research carried out is to build a test case generator system using activity diagrams based on the actor's role by testing a case study of the Digi-OTA website. The resulting test cases will be analyzed level conformity with the results of testing the Digi-OTA website. The following are the stages that will be carried out in this research:

*Correspondence Author Email: silfinformatika@student.telkomuniversity.ac.id
© JURNAL MEDIA INFORMATIKA BUDIDARMA 2022
The following is an explanation of the method used in this study in producing the sequence and implementation of the test:

1. Preprocessing
   At this preprocessing stage, the initial stage is changing the use case diagram into an activity diagram based on the actor/user. The following is an explanation at the preprocessing stage:
   a. Use case diagrams
      Use case diagrams to describe the function of a system or the interaction between the system and the actors. In the digi-OTA use case diagram, there are several actors including employees, detailing officers, and Approvers (Finance). Here is a digi-OTA use case diagram:

   b. Flow Of Events
      The purpose of making digi-OTA use case diagram scenarios is to describe each function of the system. The following is a use case scenario carried out by a digi-OTA Approving (Finance) actor:

   c. Activity diagram
      The activity diagram stage is the stage of making activity diagrams based on actors from use cases and scenario diagrams, aiming to see the activity or flow of each actor with the system. The following is the result of the activity diagram of the approval (Finance):

2. Test Case Generation
   At the stage of generating test cases, this is the stage in determining the test sequence obtained from the activity diagram that has been made at the preprocessing stage. The following is an explanation of each stage in the test case generation stage:
   a. Activity diagrams to XMI
      The activity diagram model that has been created is then converted in XMI form using IBM Rational Software Architect to get a metamodel. Metamodels are needed as input material for prototypes for making test sequences[13].
b. XMI parse
   After getting the XMI form from the UML activity diagram model, the next step is parsing or extracting
data on each of its constituent attributes.

c. XMI to Graph
   The next step after parsing the XMI form is to convert or convert it into a directed graph[14]. The use of
directed graphs is based on the UML activity diagram model that describes how the system runs or connects
activities with other related activities.

d. Test Path
   After obtaining the results of a directed graph, the next step is to search for the test path. This test path
search process is the last step to determine test cases. In the search for the test path used is the DFS (Depth
First Search) algorithm, which is one of the algorithms used for path searching.

e. Test Case
   The last stage is the results of the test case data after doing the test path stage, the results are stored in the
form of a CSV file.

3. TESTING
   At this stage it will be explained about the method used in the testing stage, namely by using the equivalence
partitioning method. In the figure 3 described the steps taken in conducting the test using the equivalence
partitioning method[15]. There are several stages, the first is the test case, the test case generated in the previous
stage is used as the basis for testing the application that has been made, namely the automatic test case generator
[16]. Next, define test data based on the partition of each specified specification, namely input valid test data
and invalid test data input. After getting test cases based on partitions, testing is carried out. The last stage is
to analyze and evaluate the results of the tests that have been carried out to determine the suitability between
the test cases generated from the automated test cases and the implementation of the tests. at this stage it will
produce feedback whether the digi-OTA application still has deficiencies or does not have defects or
deficiencies. The following is a block diagram at the testing stage using the equivalence partitioning method:

![Figure 3. Testing using equivalence partitioning](image)

Based on Figure 3, the first stage in the testing stage is the test case obtained from the previous stage,
namely the test case generation stage. The results of the test case generation are in the form of a test case in the
form of a table, the table contains test data that will be used as material for determining test cases based on a list
of data. The test is valid test data input and invalid test data input. The next stage is a direct test on the digi-OTA
application using a test case table that has been generated from the previous stage which was executed using the
selenium IDE[17]. The examiner can fill in the test results in the table with the results of "pass" or "fail". After
testing, the last stage is to analyze and evaluate the test results, at this stage you can find out the advantages and
disadvantages of the digi-OTA application and can add recommendations for improvement so that the digi-OTA
application becomes much better.

3. RESULTS AND DISCUSSION
   The test carried out in this final project is to test the prototype test case generator that has been built has been able
to generate test cases based on activity diagrams. In the testing process, a case study is used, namely the digi-OTA
website application.

3.1 Test Case Results Table
   The following is a test case for approval (Finance) generated from a prototype built with the specifications for
valid test data input and invalid test data input as follows:

<table>
<thead>
<tr>
<th>Scenario (SC)</th>
<th>ID</th>
<th>Node</th>
<th>Test Data (Valid)</th>
<th>Test Data (Invalid)</th>
<th>Expected Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC1</td>
<td>start</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC2</td>
<td>Add node</td>
<td>Add note</td>
<td>Blank</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

When the record input is the system can verify the submission and when it is
3.2 Test Results With Selenium

The next step is to conduct tests based on the previously obtained test cases using the Selenium IDE aimed at checking the suitability of the test cases generated by the prototype with the test results. The following are the results of testing using selenium from the activity diagram of the approval (finance) with the specification of valid test data and invalid test data:

**Table 3. Test Results with Selenium**

<table>
<thead>
<tr>
<th>Scenario (SC)</th>
<th>ID</th>
<th>Node</th>
<th>Test Data (Valid)</th>
<th>Test Data (Invalid)</th>
<th>Expected Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>TC1</td>
<td></td>
<td>Add node</td>
<td>Add note</td>
<td>Blank</td>
<td>When the record input is the system can verify the submission</td>
</tr>
<tr>
<td>TC2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC3</td>
<td></td>
<td>Approved?</td>
<td>Choose agree</td>
<td>Choose disagree</td>
<td>When you choose to agree, it will be returned to the detailing officer to print the document and when you don't agree, the application is cancelled</td>
</tr>
<tr>
<td>TC4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC1</td>
<td>Start</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC2</td>
<td></td>
<td>Input catatan</td>
<td>Add note</td>
<td>Blank</td>
<td>When the record input is the system can verify the submission</td>
</tr>
<tr>
<td>TC3</td>
<td></td>
<td>Approved?</td>
<td>Choose agree</td>
<td>Choose disagree</td>
<td>When you choose to agree, it will be returned to the detailing officer to print the document and when you don't agree, the application is cancelled</td>
</tr>
<tr>
<td>TC4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC1</td>
<td>Start</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC2</td>
<td></td>
<td>Input catatan</td>
<td>Add note</td>
<td>Blank</td>
<td>When the record input is the system can verify the submission</td>
</tr>
<tr>
<td>TC3</td>
<td></td>
<td>Approved?</td>
<td>Choose agree</td>
<td>Choose disagree</td>
<td>When you choose to agree, it will be returned to the detailing officer to print the document and when you don't agree, the application is cancelled</td>
</tr>
<tr>
<td>TC4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TC5</td>
<td>Approval (Finance) agrees</td>
<td>Financial officer approves</td>
<td>Financial officer disapproves</td>
<td>When the finance officer approves, the detailing officer can print the document</td>
<td></td>
</tr>
</tbody>
</table>
3.3 Analysis of Test Results

From the results of test cases that have been executed using selenium, it can be seen that the specifications with valid test data if tested using selenium produce the expected result, namely "Pass". As for specifications with invalid test data, there are several test cases that produce unexpected results, namely "fail" or the system can still be run even with incorrect input. Here are the results tested with selenium:

Table 4. Test Results with Equivalence Partitioning Technique

<table>
<thead>
<tr>
<th>Specification</th>
<th>Actor</th>
<th>Total</th>
<th>Test result</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TC</td>
<td>SC</td>
<td>Pass</td>
</tr>
<tr>
<td>Valid</td>
<td>Employee</td>
<td>299</td>
<td>14</td>
</tr>
<tr>
<td></td>
<td>Detailing Officer</td>
<td>132</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Approvers (Finance)</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Employee</td>
<td>299</td>
<td>12</td>
</tr>
<tr>
<td></td>
<td>Detailing Officer</td>
<td>132</td>
<td>9</td>
</tr>
<tr>
<td></td>
<td>Approvers (Finance)</td>
<td>5</td>
<td>2</td>
</tr>
</tbody>
</table>

It can be seen from the test results with the equivalent partitioning technique that several test cases resulted in "fail" with invalid test data input specifications, namely, the system still runs when inputted with incorrect data. This proves that there is a discrepancy. This discrepancy indicates that the software built, namely "digi-OTA" does not follow the software requirements specification. The following is a test case calculation with specifications for valid test data and invalid test data:

Table 5. Test case calculation

<table>
<thead>
<tr>
<th>Specification</th>
<th>Actor</th>
<th>Test results</th>
<th>Total testing</th>
<th>Suitability (%)</th>
<th>Incompatibility (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TC</td>
<td>SC</td>
<td>Pass</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Valid</td>
<td>Employee</td>
<td>299</td>
<td>0</td>
<td>299</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>Detailing Officer</td>
<td>132</td>
<td>0</td>
<td>132</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>Approvers (Finance)</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>100%</td>
</tr>
<tr>
<td></td>
<td>Employee</td>
<td>287</td>
<td>12</td>
<td>299</td>
<td>95.98%</td>
</tr>
<tr>
<td></td>
<td>Detailing Officer</td>
<td>126</td>
<td>6</td>
<td>132</td>
<td>95.45%</td>
</tr>
<tr>
<td></td>
<td>Approvers (Finance)</td>
<td>5</td>
<td>0</td>
<td>5</td>
<td>100%</td>
</tr>
</tbody>
</table>

Based on the calculation results, it can be analyzed that all actors or users who have been tested with valid data test specifications produce 100%, which means that the test cases produced are by the expected results at the testing stage. While the test cases were tested with invalid data test specifications, several test cases were not by the test results, namely 4.02% of users did not match and 4.55% did not match the detailing officer. Therefore, improvements are needed for test cases that are not in accordance with the expected results at the testing stage because to improve the quality of the software.

4. CONCLUSION

Based on the results of this study, it was found that this final project has built a prototype that can generate test cases using activity diagrams with the Depth First Search (DFS) algorithm. The test case generated by the prototype is then carried out in the testing phase using the equivalence partitioning method with selenium IDE as the testing tool. At the testing stage, it was found that all actors or users resulted in 100% compliance when tested with valid data test specifications while when tested with invalid test data resulted in 4.02% not suitable for employee actors and 4.55% not suitable for detail officer actors. The discrepancy is because there are several test cases which when tested are not by the expected results. Therefore, to improve the quality of the software, it is necessary to improve the software so that it can follow the specifications of software requirements.
REFERENCES


