System Testing using Black Box Testing Equivalence Partitioning (Case Study at Garbage Bank Management Information System on Karya Sentosa)

Yudie Irawan[1], Syafiul Muzid[2], Nanik Susanti[3], R. Rhoedy Setiawan[4]
{yudie.irawan@umk.ac.id[1], syafiul.muzid@umk.ac.id[2], nanik.susanti@umk.ac.id[3]}

Information System Department, Engineering Faculty, Universitas Muria Kudus, Indonesia.[1234]

Abstract. The system testing is one of the stages in system development. The system testing becomes very important because it ensures that users will not find errors in the system used. Black Box testing is one of the testing methods that focus on the functional requirements of software. The Black Box method is able to uncover error classes in the White Box test. Black Box Testing Method consists of several techniques, such as Equivalence Partitioning, Boundary Value Analysis, Comparison Testing, Sample Testing, Robustness Testing. In this study, the Equivalence Partitioning testing technique was chosen to test the Management Information System at Karya Sentosa Garbage Bank by observing the use case system as the basis for making the test case. The results of this study will show that there are several system validations that have not been fulfilled, even though they have been tested from structural testing.

Keywords: testing, black box, test case, equivalence partitioning.

1 Introduction

To determine the quality of software it depends upon how the software system is tested. Organizations and testers suggest giving 40-50% of their resources (time and budget) on testing. To achieve high level of reliability, maintainability, availability, security, survivability, portability, capability, efficiency and integrity the system must need to be properly tested. Software testing provides absolute assurance that the system is functioning properly. Critical and modern software system must be correct and provide functionality as specified. [1]

In general, there are two methods known in software testing, first is to test how the application works, this test is directed to show the level of correctness of the method used, how it works according to internal procedures and specifications. This first method is called White Box testing. The second way is to test the application functionality. This test is directed to show that the product can fulfill the function properly. This method is called Black Box testing. [2]

Along with the development of testing methods, the third method appears between the white box and black box testing, it called Grey Box Testing. This is a testing technique where there is little knowledge about the application's internal work. This technique is an independent language and platform. The grey box combines the benefits of white boxes and black box testing. Grey box designs test cases using algorithms and internal data structures less than white box testing but more than black box testing. [1][3]

ICCSET 2018, October 25-26, Kudus, Indonesia
Copyright © 2018 EAI
DOI 10.4108/eai.24-10-2018.2280526

Black Box testing has the advantage of not requiring source code, so that it does not require instrumentation and source code availability. Conversely, someone might hypothesize that accessing source code using white box testing can increase source code coverage and increase initial error statements. The white-box technique has also been claimed can be expensive and that the use of coverage information from previous versions might degrade prioritization effectiveness over multiple releases. White-box test case prioritization techniques may be inapplicable where source code is unavailable, or instrumentation is impossible, so a tester may have no choice but to use black-box strategies. It is therefore useful to know how the different black-box techniques perform against each other, motivating our second.[4]

Therefore Black Box Testing is not an alternative test, but is an important part of testing a system that is not covered by White Box Testing. In this study testing will be carried out on 8 (eight) modules that have functional including member data collection, member types, types of garbage, savings amount, types of goods, savings transactions, goods transactions and reports.

2 Research Method

2.1 Testing Software

Software testing is an essential part of the software development process. The importance of software testing can simply not be overestimated in the business practice, if the quality of the system production is to be ensured. The importance of software testing must not be too excessive in business practices, if the quality of the production system must be ensured. Effectiveness and economics are the two most important aspects of choosing a testing methodology. While effectiveness means that the designed test must be able to reveal the maximum number of errors in the software, economics implies that the test itself should consume as little time and resource as possible [5]. There are a number of rules that serve as testing targets as follows:

1. Testing is the process of executing a program with the intention to finding errors
2. A good test case is a test case that has a high probability of finding errors that have never been found before.
3. Successful testing is testing that reveals all mistakes that have never been found before. [6]

In software testing also known the principles of testing that must be understood by each engineer before testing the system are explained as follows:

1. All tests must be able to test all requirements set by the customer.
2. The test must be planned long before it begins, planning testing is made simultaneously at the stage of system development planning.
3. The Pareto principle applies to all software testing.
4. Testing starts from the small one and develops into a big test.
5. In-depth testing is not possible because of the large number of program permutation pathways.
6. Testing is carried out by a third party to get a high value of effectiveness.

Software testing is an element of a broad topic that is often interpreted as Verification and Validation (V&V).

1. Verification: refers to a collection of activities that ensure that the software has implemented a specific function.
2. Validation: refers to a different collection from activities that ensure that the software that has been built can be traced to customer needs. [7]

2.2 Black Box Testing

Black Box Testing, also called behavioural testing, focuses on the functional requirements of the software. That is, black box testing enables the software engineer to derive sets of input condition that will fully exercise all functional requirements for a program. Black-box testing attempts to find errors in the following categories:

1. Incorrect or missing function.
2. Interface errors.
3. Errors in data structures or external database access.
4. Behaviour or performance errors.
5. Initialization and termination errors. [6]

Most known black box software testing techniques are the following: [1]

1. Boundary value analysis
   There is a possibility that the system may be fail on boundary. Because there are chances of error that the programmer done at the boundaries of equivalence classes. Therefore this technique focuses on edges or values that are chosen at extreme boundaries.
2. Equivalence partitioning
   This technique helps us to reduce the number of test cases. It basically works on dividing the program input domain based on input values into equivalence classes. Test cases are generated from these equivalence classes which are derived from the input domain.
3. Orthogonal Array Testing
   Orthogonal array testing is a statistical way of test. It is used where the input domain is very small and helps to reduce the number of test combinations. Variable is represented by columns and test case is represented by rows.
4. Fuzzing
   Fuzzing is a black box testing technique that is developed by Barton Miller in 1989 at University of Wisconsin. This technique is based on feeding random input to application. Fuzz testing technique can be used to find implementation bugs in automated or semi-automated session fuzz test use malformed/semi-malformed data injection.
5. Graph based testing
   It is a black box testing technique which starts by creating a graph. The graph is created from input modules. An identifier is given to input modules. Through graph a connection is established between effect and its causes.
6. All Pair Testing
   A sort of black box technique in which the purpose of test cases is to execute all discrete combinations which are possible for input parameter of each pair to cover all the pairs we need to use a number of test cases.

7. State Transition Diagrams (or) State Graphs
   A brilliant tool that is used to capture several types of system requirements and to documented internal system design. This tool is also used to test state machine and to navigate GUI (graphical user interface).

2.3 Equivalence partitioning

The analysis of Equivalence-Classes Partitioning examines the input conditions contained explicitly or implicitly in the Software Requirements Specification (SRS). These input conditions are partitioned into a number of valid and invalid equivalence classes. Test cases are then generated for each class. The assumption here is that the test case designed for one class is representative for all other input values within the same class. The charm and the reason of the wide use of this methodology lie in the fact that by deploying a small number of test cases, a large range of input conditions can be effectively covered. [5][8]. To determine valid and invalid equivalence classes there are guidelines that must be considered as follows:

1. If an input condition specifies a range, one valid and two invalid equivalence classes are defined
2. If an input condition requires a specific value, one valid and two invalid equivalence classes are defined
3. If an input condition specifies a member of a set, one valid and one invalid equivalence class are defined
4. If an input condition is Boolean, one valid and one invalid class are defined.

By applying these guidelines for the derivation of equivalence classes, test cases for each input domain data object can be developed and executed. Test cases are selected so that the largest numbers of attributes of an equivalence class are exercised at one. [9]

3 Results and Analysis

The initial stage of Equivalence partitioning is done by determining the input domain in each application. The researcher analyzed the existing input domain by referring to the existing equivalence partitioning. Examples on the garbage savings data input form have a saving code field, member code, savings transaction date, savings transaction type, savings transaction weight and savings transaction nominal. The test case does not cover fields that are automatically filled in, or have been presented with definite choices. In this case, for example the Savings Code, Member Code, Type of Savings Transaction and Savings Transaction Nominal. The system will automatically generate the code in the Savings Code field, the Member Code field and the Saving Transaction Type containing the definite choices in the form of a dropdown, while the Savings Transaction Nominal is the result of the calculation of the specified weight and standard price. So that the test case will be created for the Savings Transaction, Transaction Date and Savings Transaction Date field. Partitions for valid and invalid values are specified as follows:

1. Savings Transaction Date

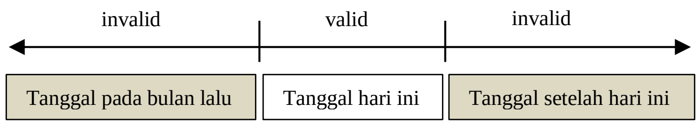

Fig. 1. Equivalence Partition for Savings Transaction Date.

In Figure 1, the date of the savings transaction has a valid provision for 30 days before the date of filling, charging less than that provision is considered invalid. And also for charging on the date after the charging date, it is also considered invalid.

2. Savings Transaction Weight

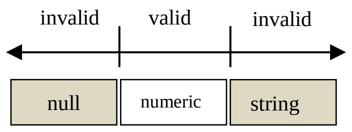

Fig. 2. Equivalence Partition for Savings Transaction Weight.

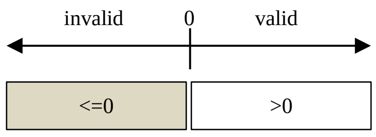

Fig. 3. Equivalence Partition for Savings Transaction Weight.

Savings transaction weight has two equivalence categories as presented in Figure 2 and Figure 3. Equivalence of the first partition is valid if it is charging using a numeric data type, while charging is invalid if it is not filled (null) or filled with a string data type. In the equivalence of the second partition, the field will be valid if it is filled with a numeric that is more than 0 (zero), and invalid if it is filled with a value below 0 (zero). After determining the equivalence partition class, the next step the researcher makes a test case.
For designing the test cases, it must be considered which input data leads to which result of the decisions or partial conditions and which parts of the program will be executed after the decision. The expected output and expected behaviour of the test object should also be defined in advance in order to detect whether the program behaves correctly. [10]

Table 1. Test case dan Test Result for Input Savings Data.

Test Case ID	Scenario/condition	Savings transaction date	Savings transactio n weight	Expec ted Result	Actual Result	Conclusi on
TC 1	Add Record Success	20 Juli 2018	1	T	T	Success
TC 2	All Record is Empty (Null)	Empty	Empty	F	F	Success
TC 3	Transaction Date is less than 30 days	20 Juni 2018	1	F	T	Failed
TC 4	Transaction Date exceeds the date when input	23 Juli 2018	1	F	F	Success
TC 5	Savings Transaction Weight is filled with strings	20 Juli 2018	ABCDE	F	F	Success
TC 6	Savings Transaction Weight is less than 0	20 Juli 2018	-5	F	F	Success

By determining the equivalence partition, the testers have the opportunity to do testing by saving more time, because they do not have to test all invalid conditions, but just take one value from each invalid class. Another advantage is that in the case of charging others, some fields have the same character as the equivalence partition class that has been determined from the previous field. So this saves time because it doesn't need to create a similar class equivalence partition again. An easy example is for each field that has a numeric type, it will be invalid if it is filled with a string, or vice versa, the string should be invalid if it is filled with numeric. So that the equivalence class of the case can be reused in the same case.

From the example of one case test in this case it can be concluded that for one field to be tested it requires a minimum of 1 (one) equivalence partition with 3 (three) valid and invalid class partitions. Although from the preparation guide equivalence partition allows at least 2 (two) class partitions only, that is in the Boolean data types, in practice Boolean data is usually inputted with choice forms such as radio buttons and dropdowns/lists/menus, so it is less likely to fill out those options. Because this can be ignored, the prediction of the total number of tests to be carried out is 3 test data for one field. For example, if in one form there are 2 (two) fields tested, then the minimum number of test data needed is 6 (six) test data. Of the 35 (thirty-five) fields tested we found a total of 7 (seven) cases consisting of errors 2 (two) filling in dates and 5 (five) string filling errors with numeric.

In a previous study of the method of Boundary Value Analysis, it was explained that the Limit Value Analysis method requires more test cases to be tested. [11] Because in the Boundary Value Analysis method, if an input condition specifies a range of values between m and n, test cases should be designed with values m and n as well as values just above and just below m and n.. The number of test cases generated is higher as compared to other functional methods. [8]

4 Conclusions

The results of testing using Black Box Testing Equivalence Partition, it can be concluded as follows:

1. Testing with the Black Box Equivalence method Partitioning begins with determining the equivalence partition class based on 4 equivalence preparation guidelines which are then used to compile a test case.
2. In the Black Box Equivalence Partitioning test, it is able to estimate the number of tests carried out as a whole based on the number of fields, the minimum number of equivalence partitions and partition classes available.
3. Equivalence Partitioning can reduce the number of total test cases that must be developed because of the error classes that represent each case of possible errors.
4. The test results in this study indicate that there are still errors in the system that must be corrected immediately, especially in the validation process, so that the system can be operated properly according to its function.

References

[1] S. Roohullah Jan, S. Tauhid Ullah Shah, Z. Ullah Johar, Y. Shah, and F. Khan, "An Innovative Approach to Investigate Various Software Testing Techniques and Strategies," Int. J. Sci. Res. Sci. Eng. Technol., vol. 2, no. 2, pp. 682–689, 2016.
[2] S. Nidhra, "Black Box and White Box Testing Techniques - A Literature Review," Int. J. Embed. Syst. Appl., vol. 2, no. 2, pp. 29–50, 2012.
[3] M. E. Khan and F. Khan, "A Comparative Study of White Box, Black Box and Grey Box Testing Techniques," Int. J. Adv. Comput. Sci. Appl., vol. 3, no. 6, pp. 12–15, 2012.
[4] M. Shi, "Software Functional Testing from the Perspective of Business Practice," vol. 3, no. 4, pp. 49–52, 2010.
[5] C. Henard, M. Papadakis, M. Harman, and Y. Le Traon, "Comparing White-box and Black-box Test Prioritization," 2016.
[6] R. S. Pressman and B. Maxim, Software Engineering, a Practitioner's Approach, 8th ed. New York: McGraw-Hill, 2014.
[7] M. S. Mustaqbal, R. F. Firdaus, and H. Rahmadi, "(Studi Kasus : Aplikasi Prediksi Kelulusan SNMPTN)," J. Ilm. Teknol. Inf. Terap., vol. I, no. 3, pp. 31–36, 2015.
[8] V. Akshatha and V. Illango, "A Comparative Analysis On Equivalence class partitioning And Boundary value analysis," Int. J. Recent Trends Eng. Res., vol. 4, no. 3, pp. 542–554, 2018.
[9] V. Arnicane, "Complexity of Equivalence Class and Boundary Value Testing Methods," Sci. Pap. Univ. Latv., vol. 751, no. May, pp. 80–101, 2009.
[10] A. Spillner, T. Linz, and H. Schaefer, Software Testing Foundation, 4th ed. Santa Barbara: Rockynook, 2014.
[11] D. Andriansyah, "Pengujian Kotak Hitam Boundary Value Analysis Pada Sistem Informasi Manajemen Konseling Tugas Akhir," vol. 7, no. 1, pp. 20–25, 2018.