IOP Conf. Series: Materials Science and Engineering **732** (2020) 012086 doi:10.1088/1757-899X/732/1/012086

The use of decision table for reducing complex rules in software testing

J Joosten, A E Permanasari* and T B Adji

Department of Electrical and Information Engineering, Faculty of Engineering, Universitas Gajah Mada, Yogyakarta, Indonesia

*adhistya@ugm.ac.id

Abstract. The testing phase is always one of the important stages of developing software. Without this stage, software errors will be difficult to find at the beginning of software development. This study uses a decision table method that aims to shorten the rule of software testing. The advantage of this method is that it can help make fewer rule combinations for testing. This study uses a medical information system as a case study. In the medical information system, 20 test cases were identified early in the software testing process. Of the 20 test cases, seven test cases that have the same characteristics so that they can be combined into only 13 test cases. Therefore, it can be concluded that the decision table method can reduce software testing time.

1. Introduction

There is an increasing demand for various software in the present as the development of technology [1]. Software development that is so much cannot be separated from several stages, one of which is the testing phase. Software testing is one of the stages of software development. With software testing, system failures can be prevented as many error cause of failure can be identified at an early stage of system development. Allocation of time become one of the main factors required in the testing process and affect the quality of a system [1]. The portion of time in doing software testing should be sufficient. However, because it is considered a waste of time, the testing phase is always ignored by the developer. The developers assume that the time allocated to software testing will be more useful for algorithm development.

Testing is one of the important stages in software development but still manual, error-prone, and costly [2]. Then software testing is a testing process that takes long and tedious and accumulated more than 50% of the cost of the software [3]. Software testing is considered unnecessary by developers because of two things, namely the large cost spent and wasting time. Both of these are described in the following two paragraphs.

The use of excessive costs in software testing causes the quality of the software to be poor. Every year, poor software quality losses exceed $ 500 billion [4]. The National Institute of Standards of Technology (NIST) conducted a study in 2012 that explained the United States economy calculates a loss of $ 59.5 billion every year due to bugs in software created [5]. NIST explains about one-third of these losses can be avoided if the developers do software testing better. More than half the cost of fixing bugs given to software users is given to developers and vendors [6].

Besides the use of large costs, another thing that assumes that software testing is not needed is the time spent. In the 1980s, one of the most notorious cases of software development failure was the Ariane

rocket [7]. The rocket exploded due to software failure. As a result of the explosion of the rocket, the researchers over the years focusing their efforts on looking for problems or bugs in the software on the rocket.

Because of these two things, software testing should be considered in detail so that those two things do not happen. The trick is to provide an optimal time when doing testing in order costs to be used is not too big. This paper focuses on simplifying rules to optimize the use of software testing time [8]. The importance of optimizing time is to test software quickly. There are four stages of the testing process: extracting test requirements, creating test cases, test executions, and bug reports. The phase that uses the most time is the stage of making test cases [1]. The stage of making a test case is also a stage that costs a very large [3]. Making the lack of proper test cases could cause the software to fail to operate and needs repair. Repair software can reach 66-90 percent of the cost of the software and half is used for understanding the code [9]. Problems faced are the decision rules of software testing is too much when developed so that at the time of testing, the error will be hard to find because of the complexity of the rule. It also takes time testing. Therefore, the aim of this study was to shorten the decision rules by making rules more simple so that errors in the software are easy to find and do not spend a lot of time.

This paper is divided into several sections. Section 1 discusses the few studies on software testing. Section II discusses the previous research on software testing. Section III describes the method and stages of the Decision Table as well as the research material. From the use of such methods, the results obtained from the application program methods will be discussed in Section IV. Section V discusses the conclusions and future work on this study.

2. Literature review

Khalid in research are seeing a trend that software testing in automated applications is vital in the development of software [10]. This became clear at the time of the application development applied to the fields of navigation, military, and biomedical. Errors are not detected in these applications will lead to losing a lot of cost and a lot of people's lives. To anticipate large losses, researchers have developed several techniques for software testing such as Unit Testing, Integration Testing, and System Testing. All the techniques developed will be divided into two categories: Black Box testing and White Box Testing. Advantages of Black Box Testing is to be used at all levels of the abstract (Unit Testing, Integration Testing, and System Testing) since this test does not require deep knowledge of programming languages. But to get a result, Black Box Testing should require the number of test cases more than any input. To complete the analysis and testing, White Box Testing methods are needed to analyze the details of the source code under test. The advantage of White Box Testing is that it can detect programming errors. But White Box Testing can only be conducted in the early stages of software development requires testing of the deep because of each module. Khalid then creates automated software testing techniques to take advantage of Black Box and White Box Testing to cover the weaknesses of both methods. The device proposed in this study was made using C language because it is more user-friendly, has a strong operator, and has many library functions.

Ali Shahbazi and James Miller discussed the methods of Black-Box Testing Namely Random Testing (RT) and Adaptive Random Testing (ART). The two methods produce a test case without accessing the source code of the program being tested. Therefore, both of these methods become independent of the language of the source code [4]. The research focuses on the Black-Box testing method where the test case is a string as input. RT is a direct testing approach. RT is not only easy to implement but also requires quite low computational costs. However, RT is not effective in error detection. The error detection performance of RT can be improved if the test case is distributed differently in the input space. While the ART approach was developed to improve RT performance. ART produces a more effective test case, but still random domain-wide input so it needs to increase the probability of error detection. To improve both methods, an approach called Random Border Centroidal Voronoi Tests (RBCVT) is used, but this method is limited to numerical test cases. In addition to numeric input, string as input also can be used many programs. Then this study developed a method to produce a string of test cases automatically due to focus on the string test cases.

3. Methodology

This research will propose a method that can shorten the rules made in the administration system in animal hospitals so as to shorten the time used in the testing phase. The method used is the Decision Table method. Decision Table is a table used as a tool to solve logic in the program [11]. Decision table testing is a software testing technique used to test system behavior for different input combinations which is also a systematic approach that combines different inputs and outputs accordingly and is summarized in tabular form [12].

Decision Table is also known as cause-effect tables for making techniques causal graphs are used to get the decision table. The reason the decision table is quite important is as follows [13]:

- Very helpful in test design techniques.
- Help testers to look for the effect of the combination of various inputs and the status of other software that must implement business rules properly.
- Provide a regular way to express complex business rules, which is beneficial for both developers and testers.
- Assist in the development process with a developer to do a better job. Testing with all combinations might not be practical.
- This method is basically a technique used in testing and managing requirements.
- This method is a structured exercise to prepare the requirements when dealing with complex business rules.
- It can also be used in complex logic models.

3.1. Decision table structure

There are four main parts contained in the decision table, namely [11]:

- Condition Stub: contains the condition to be tested.
- Condition Entry: contains the possibilities - possibilities of the condition to be tested, namely fulfilled (can be given the symbol "Y", "1", or "T") and not fulfilled (can be given the symbol "N", "0", or "F "). To find out how many Y possibilities occur, then see how many X condition to be tested. The formula determining the number of possible Y is 2X = Y.
- Action Stub: contains statements that will be carried out with the tested conditions are met or not met.
- Action Entry: used to signal which actions will be carried out and which ones will not do.

The shape of the decision table is shown in table 1:

Table 1. Decision table form.

	Rules				
	1	***2***	***3***	***…***	***N***
Condition Stub	*Condition Entry*				
Action Stub	*Action Entry*				

3.2. Decision table method stages

There are five stages of the decision table method as follows:

- Define the conditions to be tested first.
- The condition that has been set inserted into the Stub Condition.
- Create rules that occur from conditions that have been set to be filled in the Condition Entry.
- After making conditions and the number of rules, make a statement of the actions to be carried out and fill in the Action Stub section
- In the Action Entry action marked what should be done of the conditions tested.

3.3. Research materials

The research material used in this study is the medical record application program on animal hospital administration system. This study emphasizes on the part of patient enrollment. The display form of the animal registration section is shown in figure. 1:

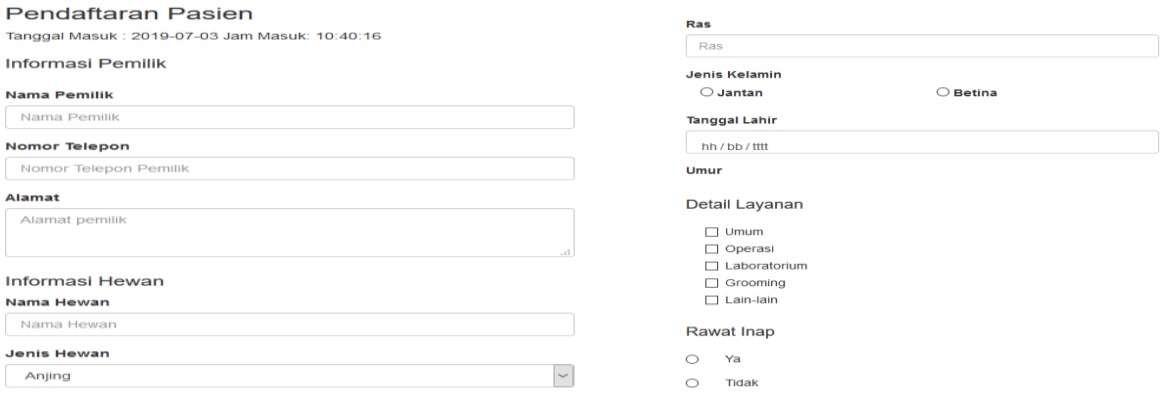


Figure 1. Display of patient registration.

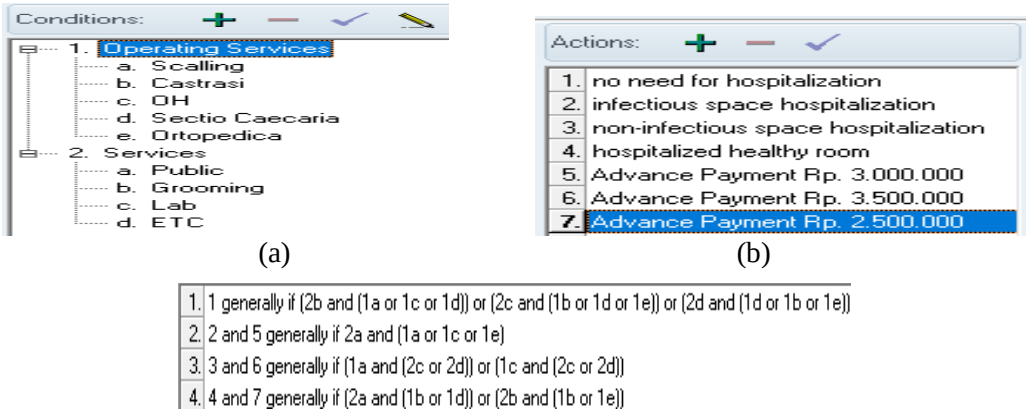
4. Result and analysis

This paper applies the decision table method to the administration program in an animal hospital. In using the decision table method, the part that will be shortened is the patient registration section. In the surgery services and inpatient options in the patient registration section, there are several services shown in figure 2:

(a) (b)

Figure 2. Type (a) surgery services and (b) inpatient rooms.

In the surgery services section synchronized with the inpatient section by combining the two parts so that the animal owner does not need to be asked again whether to be hospitalized still or not. For the down payment has also been determined based on the services taken and inpatients using software that functions to make a decision table, namely Prologa version 5.6. The conditions, actions and rules section are filled in as shown in figure. 3:



(a) (b)

(c)

Figure 3. Conditions to be included (a), Actions taken (b), Rules of conditions and actions (c).

In figure 3 (a) determine what conditions you want to work on to be included in the decision table. From the existing conditions, there are conditions for surgery services and ordinary services. From the conditions performed, the actions to be taken in figure 3 (b) are obtained. In figure 3 (b) the actions that will be performed are services such as those that do not need to be hospitalized and those that need to be hospitalized with the type of room and advance that must be paid in advance. After the conditions and

actions have been determined, then create a rule like in figure 3 (c). The first rule is for grooming services with Scalling, OH, or Sectio Caecaria surgery services or laboratory and ETC services with Castrasi, Sectio Caecaria, or Orthopedic surgery services without needing hospitalization.

The second rule explains that public services with Scalling, OH, or Orthopedic surgery services need to be hospitalized in the infectious room and must pay a down payment of Rp. 3,000,000. The third rule explains that for Scalling or OH surgery services with Laboratory or ETC services, it is necessary to be hospitalized in a non-infectious room and must pay an advance of Rp. 3,500,000. The fourth rule explains that for Scalling or Sectio Caecaria surgery services with Public services or for Scalling or Orthopedic surgery services with Grooming services need to be hospitalized in a healthy room and must pay a down payment of Rp. 2,500,000. From the 4 rules in figure 7 you will get the results of the decision tree and the decision table shown in figure 4 and figure 5:

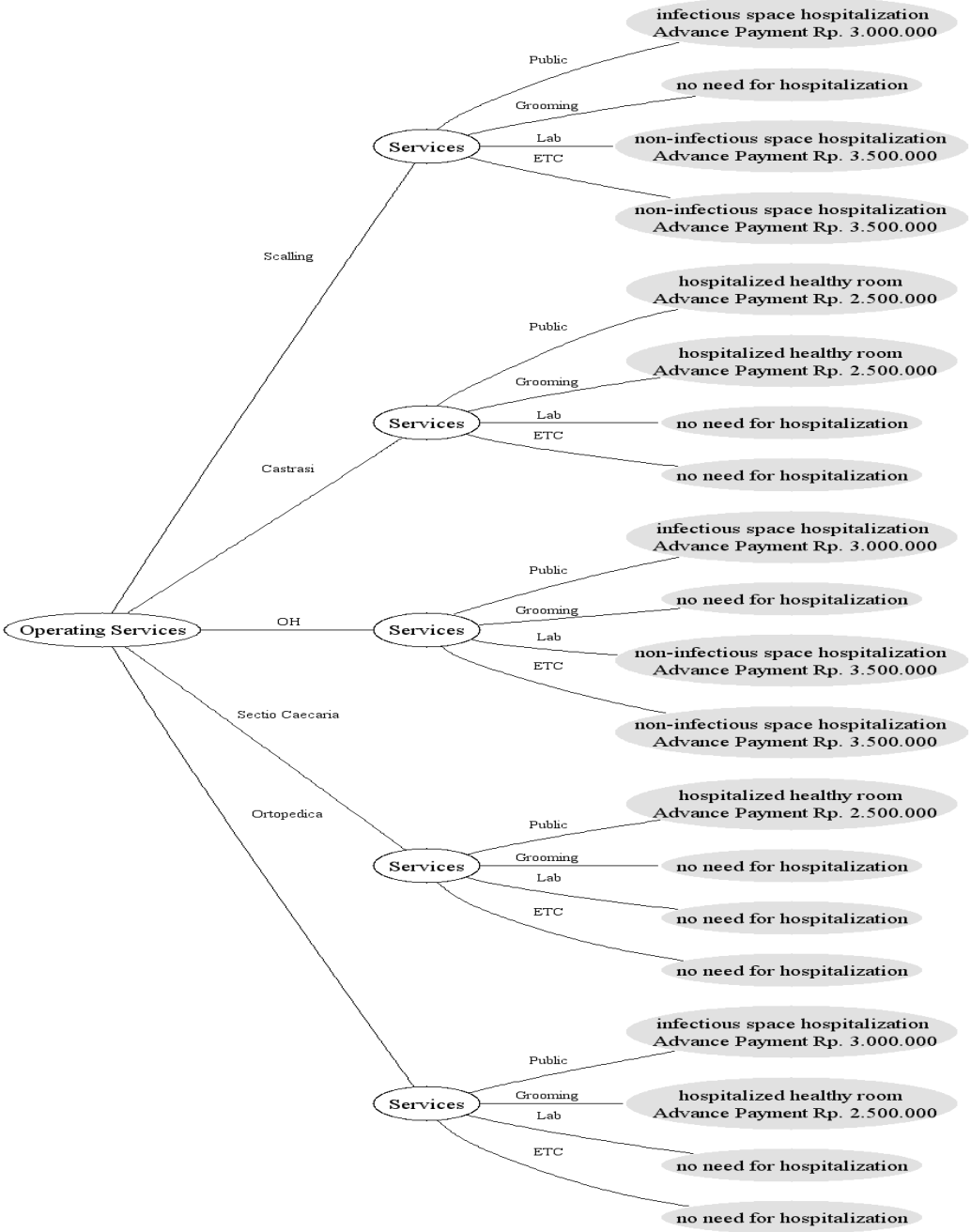


Figure 4. Result of decision tree.

1. Operating Services	Scalling				Castrasi				OH				Sectio Caecaria				Ortopedica			
2. Services	Public	Grooming	Lab	ETC	Public	Grooming	Lab	ETC	Public	Grooming	Lab	ETC	Public	Grooming	Lab	ETC	Public	Grooming	Lab	ETC
1. no need for hospitalization	.	x	.	.	.	.	x	x	.	x	.	.	.	x	x	x	.	.	x	x
2. infectious space hospitalization	x	.	.	.	.	.	.	.	x	.	.	.	.	.	.	.	x	.	.	.
3. non-infectious space hospitalization	.	.	x	x	.	.	.	.	.	.	x	x	.	.	.	.	.	.	.	.
4. hospitalized healthy room	.	.	.	.	x	x	.	.	.	.	.	.	x	.	.	.	.	x	.	.
5. Advance Payment Rp. 3.000.000	x	.	.	.	.	.	.	.	x	.	.	.	.	.	.	.	x	.	.	.
6. Advance Payment Rp. 3.500.000	.	.	x	x	.	.	.	.	.	.	x	x	.	.	.	.	.	.	.	.
7. Advance Payment Rp. 2.500.000	.	.	.	.	x	x	.	.	.	.	.	.	x	.	.	.	.	x	.	.
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20

Figure 5. Result of decision table.

In the explanation of figure 4 and figure 5, there are test cases that have the same results. As in the Scalling surgery service, test case 3 is the same as test case 4. In Castrasi surgery service, test case 5 is the same as test case 6 and test case 7 is the same as test case 8. In OH surgery service, test case 11 is the same as test case 12. In the Sectio Caecaria surgery service, test case 14 is the same as test case 15 and test case 16. In Orthopedic surgery services, test case 19 is the same as test case 20. The test cases have the same action so that they can be combined and the results of the merging of the test case shown in figure 6 and figure 7:

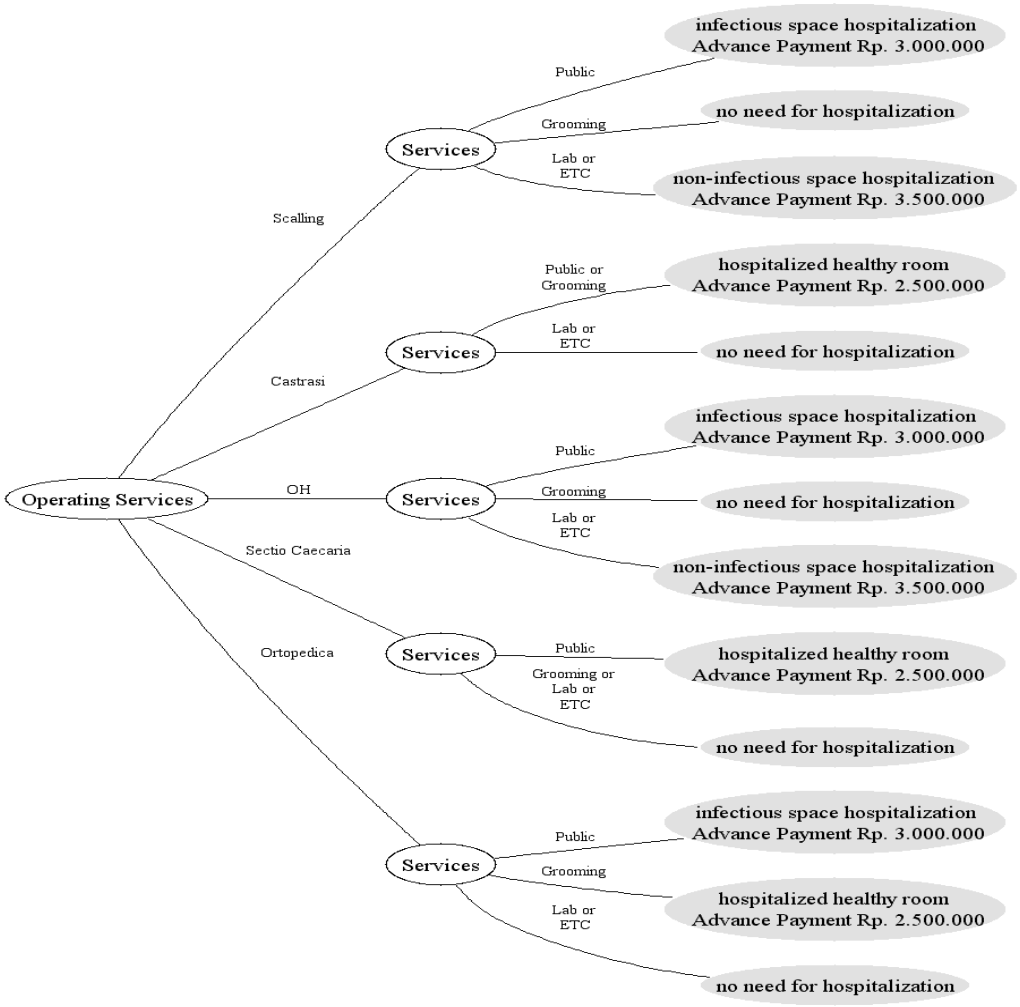


Figure 6. The results of the decision tree that have been merged.

1. Operating Services	Scalling			Castrasi		OH			Sectio Caecaria		Ortopedica		
2. Services	Public	Grooming	Lab or ETC	Public or Grooming	Lab or ETC	Public	Grooming	Lab or ETC	Public	Grooming or Lab or ETC	Public	Grooming	Lab or ETC
1. no need for hospitalization	.	x	.	.	x	.	x	.	.	x	.	.	x
2. infectious space hospitalization	x	.	.	.	.	x	.	.	.	.	x	.	.
3. non-infectious space hospitalization	.	.	x	.	.	.	.	x	.	.	.	.	.
4. hospitalized healthy room	.	.	.	x	.	.	.	.	x	.	.	x	.
5. Advance Payment Rp. 3.000.000	x	.	.	.	.	x	.	.	.	.	x	.	.
6. Advance Payment Rp. 3.500.000	.	.	x	.	.	.	.	x	.	.	.	.	.
7. Advance Payment Rp. 2.500.000	.	.	.	x	.	.	.	.	x	.	.	x	.
	1	2	3	4	5	6	7	8	9	10	11	12	13

Figure 7. Decision table results that have been merged.

In figure 7, there are a smaller number of test cases compared to the number of test cases in figure 5. The number of test cases in figure 7 shows several decisions that can be combined so that it is more efficient to be designed and developed by the developer. From figure 7, we get 13 test case results shown in figure 8:

Test case

Operating Services	Scalling			Castrasi		OH			Sectio Caecaria		Ortopedica		
Services	Public	Grooming	Lab or ETC	Public or Grooming	Lab or ETC	Public	Grooming	Lab or ETC	Public	Grooming or Lab or ETC	Public	Grooming	Lab or ETC
no need for hospitalization	.	x	.	.	x	.	x	.	.	x	.	.	x
infectious space hospitalization	x	.	.	.	.	x	.	.	.	.	x	.	.
non-infectious space hospitalization	.	.	x	.	.	.	.	x	.	.	.	.	.
hospitalized healthy room	.	.	.	x	.	.	.	.	x	.	.	x	.
Advance Payment Rp. 3.000.000	x	.	.	.	.	x	.	.	.	.	x	.	.
Advance Payment Rp. 3.500.000	.	.	x	.	.	.	.	x	.	.	.	.	.
Advance Payment Rp. 2.500.000	.	.	.	x	.	.	.	.	x	.	.	x	.
	1	2	3	4	5	6	7	8	9	10	11	12	13

Figure 8. Result of test case.

5. Conclusion

The results of the decision table method show that several test cases that have the same actions and characteristics. From the initial 20 test cases, several test cases that have similar actions and characteristics that are taken so that the number of test cases on the decision table becomes 13 test cases. The results of these 13 test cases are expected to accelerate automated testing of the coding results that have been shortened to the results of the decision table above. For the future, a combination of decision table methods and other methods will be planned that can speed up testing time to be even more efficient in the future.

Acknowledgment

The authors would like to acknowledge funding from DIKTI through the PDUPT program in 2019, Universitas Gadjah Mada.

References

[1] Nagowah L and Kora-Ramiah K 2017 Automated Complete Test Case Coverage for Web-Based Applications *2017 International Conference on Infocom Technologies and Unmanned Systems (ICTUS)* **17** 383-390

[2] Hierons R M 2014 Generating complete controllable test suites for distributed testing *IEEE Transactions on Software Engineering* **41**(3) 279-293

[3] Sakti A, Pesant G and Guéhéneuc Y G 2014 Instance generator and problem representation to improve object oriented code coverage *IEEE Transactions on Software Engineering* **41**(3) 294-313

[4] Shahbazi A and Milner J 2016 Black-Box String Test Case Generation through a Multi-Objective Optimization *IEEE Transactions On Software Engineering* **42(**4) 361-378
[5] NIST Rep 2002 *The Economic Impacts of Inadequate Infrastructure for Software Testing* [Online], retrieved from: http://www.abeacha.com/NIST_press_release_bugs_cost.htm
[6] Wong E W, Gao R, Li Y, Abreu R, and Wotawa F 2016 A Survey on Software Fault Localization *IEEE Transactions on Software Engineering* **42(**8) 707-740
[7] Lynch J 2017 *The Worst Computer Bugs in History: The Ariane 5 Disaster* [online], retrieved from: https://www.bugsnag.com/blog/bug-day-ariane-5-disaster
[8] Bohme M and Paul S 2016 A Probabilistic Analysis of the Efficiency of Automated Software Testing *IEEE Transactions On Software Engineering* **42(**4) 345-360
[9] Borstler J and Paech B, 2016, The Role of Method Chains and Comments in Software Readability and Comprehension – An Experiment *IEEE Transactions On Software Engineering* **42(**9) 886-898
[10] Khalid R 2017 Towards an Automated Tool for Software Testing an Analysis *IEEE Conferences 2017 14th International Bhurban Conference on Applied Sciences & Technology (IBCAST)* **17** 461-465
[11] ReQtest 2012 *A guide to Using Decision Tables* [online] etrieved from: https://reqtest.com/requirements-blog/a-guide-to-using-decision-tables/
[12] Guru99 2019 *Decision Table Testing: Learn with Example* [Online], retrieved from: https://www.guru99.com/decision-table-testing.html
[13] GeeksforGeeks 2018 *Software Engineering | Decision Tabl*e [Online] retrieved from: https://www.geeksforgeeks.org/software-engineering-decision-table/