SOFTWARE TESTING TECHNIQUES AND STRATEGIES USE IN NOVICE SOFTWARE TEAMS

Egia Rosi Subhiyakto[1)], Danang Wahyu Utomo[2)]
[1,2] Department of Informatics Engineering, Computer Science, Universitas Dian Nuswantoro
Jalan Nakula I, No. 5-11, Semarang 50131, Indonesia
Telp : (024) 3517261, Fax : (024) 3569684
E-mail : egia@dsn.dinus.ac.id[1)]

Abstract

Software testing is one of the stages in the life cycle of software development. Novice software developers typically perform software testing improperly. The aims of this study are to know the strategies and to get proper testing techniques that are used in black and white box, along with the enabler and inhibitor. There are 76 novice software developers involved in this study as software testers. Finally the results show that the top down strategy (93%) is more popular than bottom up (7%) in terms of testing strategy. Meanwhile in testing technique graph based and basis path are the most popular techniques in black box and white box respectively.

Abstrak

Pengujian perangkat lunak merupakan salah satu tahapan dalam siklus hidup pembangunan perangkat lunak. Pengembang perangkat lunak pemula biasanya melakukan proses pengujian perangkat lunak tidak secara tepat. Penelitian ini memiliki tujuan mengetahui strategi dan teknik pengujian yang tepat baik black box maupun white box, beserta faktor-faktor pendukung dan penghambat. Terdapat 76 pengembang perangkat lunak pemula yang terlibat dalam studi ini sebagai penguji perangkat lunak. Berdasarkan hasil studi menunjukan strategi top down (93%) lebih populer daripada strategi bottom up (7%) dalam hal strategi pengujian. Sementara dalam teknik pengujian, graph based dan basis path merupakan teknik yang paling populer masing-masing dalam black box dan white box.

Kata kunci: *Perangkat Lunak, Pengujian, Strategi, Teknik, Pemula*

1. INTRODUCTION

There are many phases in software engineering such as requirement engineering, analysis, design, implementation, testing, maintenance and retirement. In the [1] presented that students need to acquire the knowledge and skills of software engineering that cover all of the software development phases to be a competent software developers. Software testing is neither complex nor difficult to implement, yet it is a discipline that is seldom applied to anything approaching the necessary rigor to provide confidence in delivering software [2]. Testing software is not able to ensure the quality of the software, but can provide confidence and assurance of software to a certain extent. There are several reasons, including the increasing complexity, increasing market pressures and customer demands for higher quality requires a combination of carefully selected, validated, and verification to provide a software product on time, within budget, and according to quality which are desired [3]. Pham et al in the [4] stated that the main problem in software development is the lack of experience in testing students.

2. RESEARCH BACKGROUND

Testing stages are one step in the development of a software. Stages of testing become very important because it is the final stage in determining the quality of software that is built or developed. Related to quality assurance would require detailed and specific things in the process. Especially for a team of software developers, beginners become very urgent because at this stage they learn to identify factors that support or impede the process of software testing. Determine the testing strategy has also become very important because if one in determining the proper technique, the testing will be time-consuming and a big expense.

3. LITERATURE REVIEW

In this section discuss about literature review that related to software testing, software testing

for novice and software testing method and tools.

3.1 Software Testing

Software testing is a testing method that used to determine quality of software which is, it should be satisfied with the requirement. Testing method such as automatic testing, unit testing, and regression testing have different type to test the quality of software and process development. Software testing is an important part of the software development lifecycle [5]. Other researches state that software testing is a practice and study to assess and improve quality of software [6]. Based on previous research, software testing is an effective way to determine software quality. Practitioners, researchers, or developers use it to revealing faults of the program. Lemos in the [7] state that software testing is program execution using test case to find fault of program. Testing technique revealing faults based on a test case, for example test case on white box and black box. In white box testing derives test case for implementation. The way to revealing fault is check whether the output is appropriate with designing output. In black box testing derives test case from the description of the program structure. It checks complexity of the program through source code, usually use Cyclomatic complexity (quantitative measurement through source code).

In software engineering education, software testing practice can be used to improve student's ability when applies their knowledge in software development. Pham in the [8] state that main problem in software development is low experience of testing skills. Student have trouble to apply basic testing technique or method. In software project development, student misunderstanding with others when applying basic testing concept and avoid their project. Utomo in the [9] state that misunderstanding can be caused by a lack of communication between student in a group project. Low experience and misunderstanding is a factor of software engineering failure.

3.2 Software Testing for Novices

The student should understand how to adopt testing techniques, reduce barriers, and improve communication when testing process is done. Lee in the [10] state that software testing practice have problems in applying the method and tool. For instructors, need to pay attention ability of student, behavior, limitation, and requirement of the method and tool. The instructor also focuses on the testing process and activity to evaluate progress of student work. Understandability of study aims to determine the appropriate method and tool that used in software testing with development model has been proposed. Result of student's progress can be used to improve knowledge of student when applying basic testing concept in software development. The instructor can analyze progress reports to know whether student improve their ability, reduce barriers during the testing process, and improve communication between groups.

Inexperience of software testing can be problem in software development. Basically, software testing is an important part of a development model. Pham in the [8] state that obstacles and constraints of software development is lack of experience in software testing practice. Studies as developer have trouble applying testing method and tool. The main problem is how to apply a method and tool in many testing techniques. In this case, the student needs guidance to learn methods and tool and reduce misunderstanding of basic concepts with others. Teacher as instructor need to provide additional class give guidance about method and tool based on software testing technique. In addition to, the teacher can monitor their student during practice is done.

Itkonen and Lassenius in the [11] use the knowledge and experience to understand the failure of the testing process. It can be used to evaluate the failure of student testing practice. The evaluation, based on personal knowledge of the student and their progress report on testing practice. The experience not only used to evaluate the ability of the student, but also to know the behavior of the testing method. It means that students with experience can easily determine what is the method and tool use in software testing practice. For example, experience using test case on regression testing and automatic testing. The tester should be able to determine domain in the method. Test cases must be in accordance with the proposed testing method.

3.3 Software Testing Method and Tool

Method and tool is aimed to support improved quality of software. Lee in the [10] state that method and tool can improve software testing practices in term of the use of the method and tool, the barrier in software testing practice. Software testing methods are concepts, techniques consist of rules and steps of software testing tasks, for example black box and white box is technique for test design. In test design, it used to test the application that related to design such as Cyclomatic complexity, interface, input

and output of the system. Software testing tools are tools or software products included testing method to support software testing tasks. In software testing practices, software testing tools assist students to apply their knowledge and experience. A student can practice what they learned in theoretic class. Theoretically, it can help the student to improve their ability coverage knowledge and experience, but in practice, student avoids testing tools because they should learn first the tools before applying the method. The instructor should guide student how to use the tool, how to apply the testing method in the tool. In addition to, the institution should provide a cost to buy the tool. Lee in the [10] state that the usage of software testing tool is lower than the usage of software testing method because the high complexity and difficult to use. Melo state that there is no methodology to analyze and select a testing tool based on the requirement has been designed. Furthermore, there is no literature that focuses on testing tools. It is one of factor software testing tool rarely used in software testing practices [12].

Other factors that influence software testing practice are the usage of questionnaire and interview. The questionnaire used to find the weakness of software, characteristics of personal, experience of the user, and testing domain included method and tool. For novice team, the questionnaire is a low-cost tool to practice and apply testing method. Student only prepares some question related the topic of testing, give it to the user that related with the system and the last, analyze the result. Interview used to find the influence of software testing based on perception and opinion of users. The interview is important for novice teams to improve communication skill with stakeholder. Different with a questionnaire, in the interview conducted like a conversation, tester asks directly to the stakeholder. The opportunity of the interview is tester know what the user really want the system, what the weakness of the system.

4. RESEARCH METHOD

The research method used in this study is the survey method. This method performs the survey process by distributing questionnaires to a novice software teams. There is 19 software that used for testing, consist of 16 web application, 2 mobile application, and 1 desktop application. All of the teams conduct the testing along two weeks, and they are conducting the testing based on the software that they build. After that, they fill the questionnaire to evaluate testing stage.

The questionnaires were measured using Likert scale [5 points], where "1" for strongly disagree, "2" for disagree, 3 for undecided, "4" for agree, and "5" for strongly agree". In the table 1 present the survey questionnaire that consists of three part namely system metrics, process metrics, and usability.

Table 1. Survey Questionnaires

No	Statement
1	System Responsive
2	Size of system is big
3	System has a good performance
4	System efficient
5	Has done the maximum effort in testing
6	Use the time provided with maximum
7	Total defect in the software is very much
8	Interesting interface
9	Easy to use
10	Easy to understand
11	User Interface
12	Functionality

Table 2. Questionnaires testing strategies and techniques

Testing Strategies
1. Top Down
2. Bottom Up
Testing Techniques
1. Black Box
a. Graph Based Testing
b. Equivalence Partitioning
c. Boundary Value Analysis
2. White Box
a. Basis Path Testing
b. Graph Matrix
c. Control Structure Testing
d. Data Flow Testing
e. Lines of Code

In the other hand, survey questionnaire also provides testing strategies, testing techniques, enabler and inhibitor in the testing of software. In the table 2 presented about that. For testing strategies and technique every student can choose 1 to complete the questionnaire based on the testing software. Top down strategy is a software testing strategy that tests the software from the largest to the smallest parts in the software (module) while the bottom-up strategy is a software testing strategy that tests the software from the smallest to the largest part. Besides, for enabler and inhibitor students can fill minimum 3 factors that enable or inhibit in the testing software. Results of the questionnaire will be analyzed in next section to ascertain the validity of the study.

5. TESTING PRACTICES AND PREFERENCES

A general overview of respondent demographics for the complete survey is presented in table 3.

Table 3. Respondent demographic

Gender	
Male	74
Female	2
Semester	
5	7
7	60
9-above	9
Team Size	
I am not in the team	2
1-2 people	9
3-4 people	5
5-6 people	5
Cost	
Rp. 0-500.000	14
Rp. 500.000-1.000.000	4
Rp. >1.000.000	1
Schedule	
< 1 month	9
2-3 months	6
> 3 months	4

In this study conducted testing software involving 76 participants. Based on table 3 shows respondent consist of 7 students a third year, 60 students fourth year, and 9 students the fifth year and above. Then, they divided into team size with details "I am not in the team", group's 1-2 people, groups' 3-4 people, and groups 5-6 people. In table 3 also shows cost and schedule development of the software before perform the testing.

6. DATA ANALYSIS AND DISCUSSION

In this section, we carried out an evaluation results from questionnaires that conducted. There are 76 participants in this study consist of 74 male and 2 female. Participants in this study consist of 7 third year software engineering students, fourth-year software engineering students and 9 fifth year and above software engineering students.

Based on research method and practices and preferences we can analyze data. This analysis is performed on eight different aspects such as system metric, process metric, usability, user satisfaction, testing strategy, testing technique, enabler, and inhibitor. Based on the questions in the previous section can obtain percentage for each questions using the formula: Y=P/Q * 100%, in which P is a number of respondents answer each question, Q is a number of respondents and Y is a percentage value.

Table 4. Evaluation result for system metric, process metric, usability, user satisfaction

	Number of Respondents				
System Metrics	SA	A	U	D	SD
System Responsive	16	35	10	13	2
Size of system is large	7	7	19	34	9
Good performance	7	22	16	25	6
System efficient	6	23	6	37	4
Process Metrics	SA	A	U	D	SD
Maximum effort	23	48	2	3	0
Use the time maximum	17	47	5	7	0
Total defect	17	27	13	18	1
Usability	SA	A	U	D	SD
Interesting interface	9	26	17	22	2
Easy to use	11	36	8	19	2
Easy to understand	12	24	12	25	3
User Satisfaction	SA	A	U	D	SD
User Interface	8	26	16	23	3
Functionality	7	14	20	32	3

Based on figure 1 below can be concluded in system metrics for each parts with system responsive (67% strongly agree and agree), the size of the system is not too large (over 56%), system haven't good performance (over 40%), and system not efficient (over 53,9%). From this result can be concluded in the system metrics software have been tested in this study that overall system responsive, the size of the system is not too large, overall system haven't good performance, and overall system not efficient.

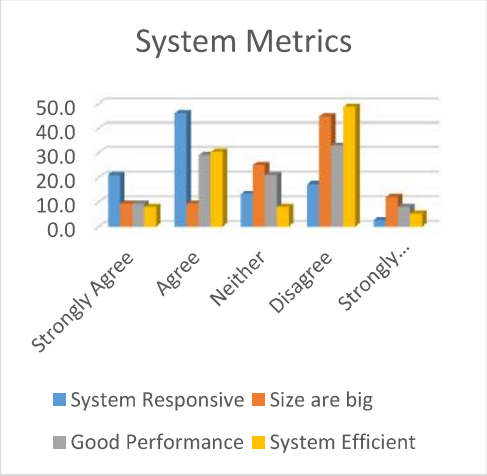

Figure 1 System Metrics

In process metrics based on the graph below figure 2 can be concluded that the students overall using maximum effort to testing (over 93%), using the maximum time to testing (over 84%), and total defect in overall system reached (over 57%).

Figure 2 Process Metrics

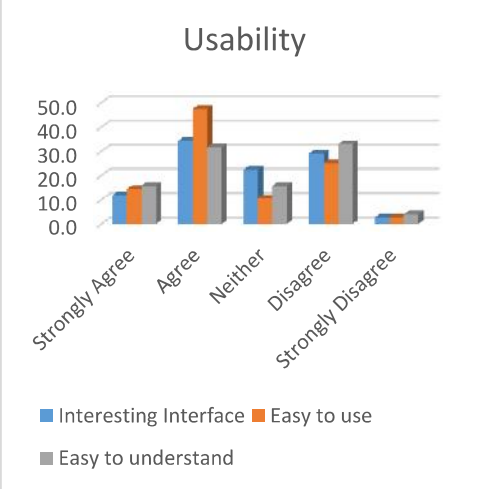

Figure 3 Usability

Usability from the overall software has been tested based on figure 3 shows the students interested with the software (over 46% with strongly agree and agree with interesting interface of the system), then for easy to use the software students easy to operate the software in which it shows overall software easy to use (over 61% strongly agree and agree with easy to use). And last for usability is easy to understand, in figure 3 shows have positive results for easy to understand of the software (over 47% with strongly agree and agree).

Evaluation for user satisfaction presented in figure 4. The user interface from overall software shows positive results (over 44% with strongly agree and agree), but students feel less satisfied with the functionality of the software (over 46%).

Figure 4 User Satisfaction

Table 5. Evaluation result for testing strategy and testing technique

Testing Strategy	Quantity
Top Down	71
Bottom Up	5
Testing Technique	
Black Box:	
Graph Based Testing	63
Equivalence Partitioning	11
Boundary Value Analysis	2
White Box:	
Basis Path Testing	55
Graph Matrix	7
Control Structure Testing	0
Data Flow Testing	0
Lines of Code	14

Evaluation result for testing strategy and testing techniques presented in table 5. Based on questionnaire results, 71 (over 93%) participants choose strategy top down, and only 5 (around 6%) choose bottom up. It can be concluded that participant overall chooses strategy top down when they do testing software. Whereas for testing techniques divide into 2 part including black box and white box testing. In the black box participants mostly prefer using graph based testing techniques (63 participants or over 82%), then 11 participants (over 14%) using equivalence partitioning, and 2 participants (over 2%) using boundary value analysis. In the other hand, for white box testing 55 participants (over 72%) mostly prefer to use basis path testing, 7 participants (over 9%) using graph matrix and 14 participants (over 18%) using lines of code. From this results can be concluded that strategy top down mostly prefer to use in this testing study, then graph based testing technique (black box) and basis path (white box) constitute mostly prefer technique testing in this study.

Last evaluation for enabler and inhibitor are presented in table 6 below. There are 14 factors which became an enabler and 15 factors which became inhibitor in doing testing. For each factor obtained from students either before, during or after doing testing.

Table 6. Result for enabler and inhibitor

Enabler factors	Qty
Good team coordination	41
Software Compatible with Hardware	7
Simple software	15
No need for an internet connection	3
System responsive	8
Application is not using the framework	1
The system is easy to understand	12
Granted full access to the software	7
Scheduled testing time	17
Open Source Applications	1
Installing the program is easy to understand	4
Web-based applications that are familiar	4
Application completed with a module	3
The software is easy to use	7
Inhibitor	**Qty**
The file is too large	4
A change of software	12
Complex program flows	10
Poor communication	20
Delays in software distribution	16
The software is not compatible with OS	10
Not given a note to log in	7
Use unfamiliar framework	5
The developer less cooperative	6
Internet connection is unstable	5
Inefficient User Interface	8
Too many bugs	16
Software easy to hack	4
Elusive software	10
No documentation of analysis	7

Table 6 shows the results for enabler and inhibitor factors in this testing study. Based on the table above shows that good team coordination (53%) is a factor that most support in testing, whereas poor communication is the most hindering factor in this testing study.

7. CONCLUSION AND FUTURE WORK

There are several things that can be inferred from this study. To measure the evaluation of test results and feedback from the development team used questionnaires about systems, processes, techniques and strategies as well as the testing of the enablers and inhibitors in this testing study. Based on the results of the questionnaire, 71 respondents (93%) chose a strategy top down and only 5 (7%) who chose a bottom-up testing strategy. It can be concluded that the majority of respondents prefer to do a top-down strategy in software testing. As for the testing, the technique is divided into two parts, namely the technique black box testing and white box. In the black box the majority of respondents prefer a graph-based technique (63 respondents or 82%), and for the white box techniques majority of respondents chose basis path technique (55 respondents or 72%). From these results, it can be concluded that for black box testing techniques, graph-based in the black box and basis path in the white box is more widely used in this testing study. Meanwhile for enabler and inhibitor, good team coordination (53%) is a factor that most support in testing, whereas poor communication is the most hindering factor in this testing study. To obtain a better result, future work should use software with the same platform and longer testing time.

ACKNOWLEDGEMENTS

This research was supported by LPPM Universitas Dian Nuswantoro under grant no 021/A.35-02/UDN.09/X/2015 for funding this research.

REFERENCES

[1] E. Subhiyakto and M. Kamalrudin, "Customization of Requirements Modeling Tool For Software Engineering Education," *Int. Symp. Res. Innov. Sustain.*, vol. 2014, no. October 2014, pp. 1581–1584, 2014.

[2] P. Morgan, B. Hambling, A. Samaroo, G. Thompson, and P. Williams, *Software Testing: An Istqb-Iseb Foundation Guide*. 2010.

[3] A. Aurum and C. Wohlin, *Engineering and Managing Software Requirements*. 2005.

[4] R. Pham, S. Kiesling, O. Liskin, L. Singer, and K. Schneider, "Enablers , Inhibitors , and Perceptions of Testing in Novice Software Teams," *FSE'14*, no. ACM, pp. 30–40, 2014.

[5] V. Garousi and J. Zhi, "A survey of software testing practices in Canada," *J. Syst. Softw.*, vol. 86, no. 5, pp. 1354–1376, 2013.

[6] A. Orso and G. Rothermel, "Software testing: a research travelogue (2000–2014)," *Proc. Futur. Softw. Eng. - FOSE 2014*, pp. 117–132, 2014.

[7] O. A. L. Lemos, F. C. Ferrari, M. M. Eler, J. C. Maldonado, and P. C. Masiero, "Evaluation studies of software testing research in Brazil and in the world: A survey of two premier software engineering conferences," *J. Syst. Softw.*, vol. 86, no. 4, pp. 951–969, 2013.

[8] R. Pham, S. Kiesling, O. Liskin, L. Singer, K. Schneider, R. Pham, S. Kiesling, O. Liskin, and K. S. De, "Enablers , Inhibitors , and Perceptions of Testing in Novice Software Teams Categories and Subject Descriptors,"

Fse'14, no. ACM, pp. 30–40, 2014.

[9] D. W. Utomo, E. R. Subhiyakto, S. Ahmad, P. Studi, T. Informatika, F. I. Komputer, and U. D. Nuswantoro, "Tool Enhancement For Collaborative Software Engineering Education," vol. 2015, no. Sentika, pp. 9–16, 2015.

[10] J. Lee, S. Kang, and D. Lee, "Survey on software testing practices," *IET Softw.*, vol. 6, no. 3, p. 275, 2012.

[11] J. Itkonen and C. Lassenius, "The Role of the Tester ' s Knowledge in Exploratory Software Testing," vol. 39, no. 5, pp. 707–724, 2013.

[12] S. M. Melo, S. R. S. Souza, R. A. Silva, and C. Sp, "Concurrent Software Testing in Practice : A Catalog of Tools," pp. 31–40, 2015.