Proc. of the Intl. Conf. on Advances in Computing and Information Technology-- ACIT 2014

ISBN: 978-981-07-8859-9

Measuring Test Case Reusability Based on Simplicity and Independence

Mohammad Rava, Asst. Prof. Dr. Jirapun Daengdej

***Abstract*— Test Case reuse is known as a prominent solution for reducing the cost of testing while increasing the test reliability and productivity. There are many methodologies presented that concentrate on generating reusable test cases, however there are very few that concentrate on measuring how reusable a test case is based on internal and external characteristics. This research investigated several metrics for measuring test case reusability and selected two that were most prominent. The two metrics are then combined and result a measurement model for measuring test case reusability. The prominent characteristic of this model is that it can serve as a template for test case reusability as it will allow for further metric factors to join and expand upon it.**

***Keywords*— Test Case, Reusability, Metric, Independence, Simplicity.**

I. Introduction

Software testing has been estimated to take as much as 70% of the overall cost of producing the application or software [1]. This implies the importance of software testing in software development process.

A common practice to reduce the cost of test development is through reusability [2], particularly test case reusability. This practice helps testers to avoid duplicating their efforts in order to create the same test case, doing so would also improve the quality of software testing and greatly reduce the cost of production which would lead to further enhancing the productivity of software companies [3]. This has led software developers and organizations to consider test case reusability as method to reduce the costs.

In order to better choose test cases to reuse, it is essential to understand how to measure test case reusable quality [4]. Several methods of reusability are investigated in this paper, and two prominent factors are presented. The factors are independence and simplicity, and are combined together to measure test case reusability. This measurement model serves as a basic template that will allow new factors to be added and included in later iterations of the model.

This research has three main objectives. The first objective is to discover several potential criteria for measuring reusability in software systems, and identify potential reusability factors for test cases in software testing. The second objective is to create a basic expandable template for measuring test case reusability based on independence and simplicity for test cases generated from control flow diagrams and use cases. And the final objective is to evaluate the template by using different samples for generating test cases based on control flow diagrams and use cases.

The remainder of this paper is organized as the following: Section two starts with briefly explaining and elaborating the basic background required to understand the model, and later expands on some of the previous work that have been done in measuring reusability in general, test case reusability, and different measurement methods suggested for creating a metrics for reusability in test cases. Section three expands on the current problems faced by the different measurement models, and why some of the factors they provide are not entirely usable in other scenarios. Section four elaborates the proposed solution and the new presented factors. Section five evaluates and analyzes the solution by implementing it on test case scenario. Section 6 concludes the paper by briefly going over the advantages and the drawbacks of the model, and also how can future studies expand upon the current solution.

II. Background and Literature Review

A. *Test Case*

The Institute of Electrical and Electronics Engineers defines a test case as "A set of test inputs, execution conditions, and expected results developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement. [5]". A test case is essentially a mechanism in which a tester will use to evaluate if a software or application is functioning properly. The mechanism is usually a set of conditions or variables under which the tester will use to make those evaluations.

Test cases are considered to be an essential part of testing process, thus reusability is often directed towards test cases with the intent on reducing the cost of testing. The main purpose of reuse in testing is to reduce implementation time and decrease the chance of bugs and errors appearing, since prior testing on those modules has refined them. Test cases often contain many different components and attributes, however according to IEEE Standard 829 [21] there are several basic attributes that can be used in order to generate a test case. For the purpose of this research the components have been modified based on the standard [21] and requirements seen on test case reusability metrics model [4]. Thus we have a total of eight criteria which are:

Mohammad Rava
Assumption University
Thailand

Asst. Prof. Dr. Jirapun Daengdej
Assumption University
Thailand

Proc. of the Intl. Conf. on Advances in Computing and Information Technology-- ACIT 2014

ISBN: 978-981-07-8859-9

Test Case ID, a test case specification identifier [21], it is used to create a unique identifier for each test. Test Item is the item that is being tested from the system/application. Test Case Objective is the purpose of the test case, sometimes referred to as Test Case Description, used to describe the test case in a few simple words. Test Case Keywords are used for the purpose of searching test cases; this particular attribute is based on Test Case Reusability Metrics Model [4] which is used as a metrics component. Test Inter-Case Dependency – a components based on the standard [21] – is required in the measurement of test case reusability according to TCRMM [4]. This component mainly keeps track of the number of precursor test cases. Test Steps also known as steps to execute, are procedures that need to be executed in order to successfully perform a test case. Test Data which is based on the standard [21], which mainly tracks the values involved in the verification of the test case. And finally Expected Result which is used to verify the result of the test case, mainly used as comparison with actual result. A test case only passes when the actual results and expected result are matching.

There are several methods to generate test cases. The main method used in this research is based on Control Flows and Use Cases. In order to generate a test case, the user needs access to the use case or the control flow of a system which will grant him different sequences and pathways to be tested in a specific sub-system or function [22]. This method grants a total coverage of all the functions in the test case.

B. *Reusability Factors*

Many studies and researches from the computer science and software industry have analyzed the benefits of software reuse and reusability and believe that it plays a key strategic factor in improving software quality, productivity and reliability as well as reduce development cost [6] – [13].

Depending on the aspect of reuse, researchers have provided different insight into the factors that affect reusability. The study [12] follows the common belief that large components are harder to reuse. According to widely used cost estimation model knows as The Constructive Cost Model II (COCOMO II) [14], it is considered that software reuse and reusing components cost is higher if the reusable component is larger. Using regression analysis the relationship between the complexity of a component and the ease of reuse was analyzed. An inverse correlation was found between the complexity and ease of reuse, which means the higher the complexity, the lower the ease of reuse. This means complexity is an essential factor for measuring reusability on any level of software development.

The study "Test Case Reusability Metrics Model" (TCRMM) [4] presents four factors for measuring test case reusability based on the collective experience of the authors in the field of software testing and reference to software engineering product quality ISOI/EC 9126 [15] came up with four factors they used for measuring test case reusability. The factors are Understandability, Changeability, Independence and Universal. It is assumed that more understandable a test case is the more reusable will it be, thus understandability is measured based on the number of characters in a test case summery, and the number of test keywords and test items. In the second factors it is theorized that the more changeable a test case is the more reusable it will be. Changeability is measured based on the number of variables and constants in a test case. The third factor is that the more independent a test case is from other test cases, then the more reusable that test case will be. The independence factor is measured based on the number of precursor test cases that relate to the measuring test case. Universal factor is that the more universal a test case is, the more reusable it will be, and it is measured based on the number of involved functions and the number of hardware and software scenarios.

C. *Problem Statement*

The factors elaborated in Test Case Reusability Metrics Model are rather situational and work based on presumed criteria. To measure understandability the researchers assume a degree of meaningfulness and relevance of the sentences and information provided. However in many cases when the number of characters is among the main metric factors, then there could be cases in which the sentence could be completely unrelated to the test case and still yield a full score for understandability. Although understandability is considered to be a strong factor for reusability [12] there are alternative models available that rely on more scientific approach [16] on measuring understandability rather than a purely linguistics approach.

The changeability factor is also based on presumed values that are not freely defined in many test case generation methods. The changeability formula provided by the authors of "Test Case Reusability Metrics Model" [4] does not follow the theoretical principle behind it. The formula is seen in Equation (1), where C_C stands for changeability of constant c in test case t:

$$C_C(t) = \begin{cases} \frac{1}{c}, & c \neq 0 \\ 0, & c = 0 \end{cases} \quad [4] \quad (1)$$

The theory is that the fewer constants a test case has, the more changeable the test case will be. However as seen in Equation (1), if there are no constants, then the changeability value is set to zero and not one. This means that the changeability value for no constants is mathematically equal to the changeability of an infinite number of constants.

Measuring universal does not hold any weight system for the number of applications and the number of hardware/software scenarios. Equation (2) demonstrates how universal is measured for case of application universality. Consider UN_F as the percentage of universal properties to applications of the test case "t". The number of application in which test case "t" can be used is shown with "f":

$$UN_F(t) = \begin{cases} 0, & f = 0 \\ 1 - \frac{1}{f}, & f \neq 0 \end{cases} \quad [4] \quad (2)$$

As seen, there is no indication for the weight of the applications, whether they are complex or simple. In cost

Proc. of the Intl. Conf. on Advances in Computing and Information Technology-- ACIT 2014

ISBN: 978-981-07-8859-9

calculation methods such as COCOMO II, there is a weight system of the application programming language [14]. The level of programming language used to write the code directly affects the complexity of the application and in turn increase the effort required to create them. However such consideration is not seen in universal factor, and thus would make for an unreliable metric in measuring how universal and general a test case is.

III. Proposed Solution

A. *Independence*

Independence is based on the same factor elaborated in Test Case Reusability Metrics Model [4] by which is measured based on the number of precursor test cases. To calculate independence, the variable "I" is considered a percentage of independence property of test case "t", and the number of precursor test cases which test case "t" has is represented with "p". The formula is seen in Equation (3). The value of I(t) is always between 0 and 1, with the possibility of being 1 when p is equal to 0, meaning that there are no precursor test cases.

$$I(t) = 2^{-p} \quad [4] \qquad (3)$$

B. *Simplicity*

Simplicity is based on theories that measure complexity of the test case and components. Among the most prominent and reliable methods of measuring complexity in software systems is cyclomatic complexity, which is a software measurement technique developed in 1976 by Thomas J. McCabe [17]. This method is based on software control flows and it mainly measures the complexity of the system by how big and how diverse it is. The more conditions and states exist in a system, the more complex the system will be. Cyclomatic Complexity grants a basic idea of the need to measure complexity from a white box perspective.

However according to researches performed in the field of measuring complexity [17], [18], [19] and [20], it has been suggested that a hybrid solution of both black box and white box methods is required in order to accurately measure the complexity of a component, and thus in order to measure the complexity of the test case two aspects are used that comprise of a grey box method. However for the purpose of measuring reusability, simplicity is measured. The reason simplicity is used as a measurement factor instead of complexity, is mainly because in sense of measuring reusability percentage the closer the value is to 1 (one) the more reusable it is considered, and thus simplicity is introduced as a form of measuring complexity in test cases.

To measure simplicity two factors are considered. Simplicity based on test items which measured simplicity based on the number of test items [21]. The more items a test case tests, the more complex it would be, and thus the less reusable. This results in Equation (4), in which N_{TI} is the number test items in a test case, and S_{TI} is the value of simplicity based on test items. This form of simplicity is based on the black box aspect of the test.

$$S_{TI}(t) = \frac{1}{N_{TI}} \qquad (4)$$

To fully measure simplicity a combination of both white box and black box is needed. In order to measure white box simplicity the concept of Cyclomatic Complexity [17] is used in which complexity is measured based on the number of nodes and edges in a control graph. Since test items are considered a black box element then a white box element is needed in order to create the hybrid simplicity measurement equation. Thus the number of edges is used to measure the white box simplicity. The simplicity based on edges is shown in Equation (5), in which N_E is the total number of edges in a control flow traverse scenario for test case t. S_E stands for the value of simplicity based on control flow edges.

$$S_E(t) = \frac{1}{N_E} \qquad (5)$$

In order to reach the total value for overall simplicity of the test case an average of both values are used as they each represent a different aspect of simplicity. The value of S (t) in Equation (6) is the final simplicity value for test case t, which is a combination of both simplicity values.

$$S(t) = \frac{S_E(t) + S_{TI}(t)}{2} \qquad (6)$$

C. *Test Case Reusability Final Template*

In order to measure the reusability metrics for the test case "t" the two factors (Independence and Simplicity) are combined and averaged. In Equation (7) R(t) stands for the reusability value of test case t.

$$R(t) = \frac{I(t) + S(t)}{2} \qquad (7)$$

There is currently no weight system between the factors, both independence and simplicity have equal weight and effect in measuring reusability of a test case.

IV. Implementation and Evaluation

The proposed model was applied to three different systems with their test case groups to demonstrate the applicability of the solution in different scenarios. Among the selected cases the "ATM Withdraw" activity diagram and initial test case selections are used to illustrate how the model applies. The activity diagram is taken from Ian Summerville book on Software Engineering [24] with minor modification by another research [25] that also extracted the initial set of test cases for the "ATM Withdraw" case. Figure 1 shows the activity diagram for the ATM Withdraw.

Proc. of the Intl. Conf. on Advances in Computing and Information Technology-- ACIT 2014

ISBN: 978-981-07-8859-9

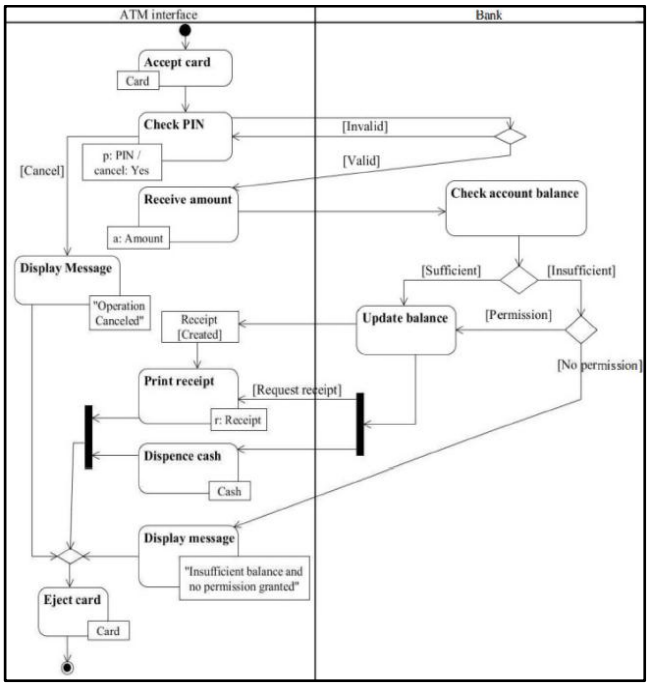

Figure 1. The ATM Withdraw Activity Diagram [24 – 25]

The system has two main interfaces; one is the ATM interface by which the user interacts with, and the other the Bank interface which the ATM interacts with.

The system starts by accepting the ATM card from the user, once accepted the user proceeds to insert his pin number which is promptly checked by bank, if invalid the user may have to enter the pin again or cancel which then the system will respond by ejecting the card. If the pin is accepted the user then proceeds to enter the required amount, the amount is then checked with the bank and the account holder's balance, if insufficient the bank then checks if the user has permission to over withdraw from balance, if there is no permission the message "Insufficient balance and no permission granted" will appear on the screen and the card is ejected. If there is permission the balance is then updated and a receipt created. The receipt is then printed and the cash is dispensed at the same time and the system proceeds to eject card and end.

In order to extract test cases from activity diagrams, they must be converted to Activity Diagram Graphs (ADG) which in this case essentially serves as control flows for the system. The converted diagram is created by the researchers of Test Case Generation Based on Activity Diagram [25] and is shown in Figure 2.

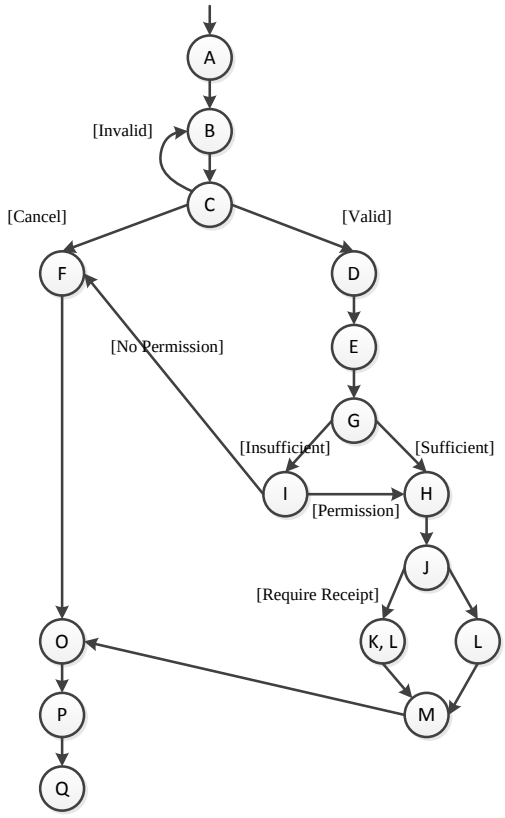

Figure 2. The ATM Withdraw Activity Diagram Graph [25]

Using the Activity Diagram Graph in Figure 2 the researches generated test paths for testing the diagram. However the test paths used for the measurement model do not have any test case performance techniques applied and are of average quality (including the redundancies present in the system) and expect to generate an average result of reusability [25]. The test paths are shown in Table 1.

TABLE 1. The ATM Withdraw Test Paths

Test Path	Test Path Sequence
$Test\ Path\ 1$	$\{A \rightarrow B \rightarrow C \rightarrow B\}$
$Test\ Path\ 2$	$\{A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow G \rightarrow H \rightarrow J \rightarrow KL \rightarrow M \rightarrow O \rightarrow P \rightarrow Q\}$
$Test\ Path\ 3$	$\{A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow G \rightarrow H \rightarrow J \rightarrow L \rightarrow M \rightarrow O \rightarrow P \rightarrow Q\}$
$Test\ Path\ 4$	$\{A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow G \rightarrow I \rightarrow F \rightarrow O \rightarrow P \rightarrow Q\}$
$Test\ Path\ 5$	$\{A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow G \rightarrow I \rightarrow H \rightarrow J \rightarrow KL \rightarrow M \rightarrow O \rightarrow P \rightarrow Q\}$
$Test\ Path\ 6$	$\{A \rightarrow B \rightarrow C \rightarrow D \rightarrow E \rightarrow G \rightarrow I \rightarrow H \rightarrow J \rightarrow L \rightarrow M \rightarrow O \rightarrow P \rightarrow Q\}$
$Test\ Path\ 7$	$\{A \rightarrow B \rightarrow C \rightarrow F \rightarrow O \rightarrow P \rightarrow Q\}$

Each test path represents a single test case, which is represented using the IEEE Standard 829 [21] mentioned earlier in the introduction. Applying the model to each test case separately has resulted in Table 2 which includes all the values that result from the generated test cases.

Proc. of the Intl. Conf. on Advances in Computing and Information Technology-- ACIT 2014

ISBN: 978-981-07-8859-9

TABLE 2. The ATM System Reusability Measurement Results

Test Case ID	Independence	Simplicity	Reusability Percentage
ATM_01	100%	62.50%	**81.3%**
ATM_02	100%	16.35%	**58.2%**
ATM_03	100%	16.35%	**58.2%**
ATM_04	100%	14.54%	**57.3%**
ATM_05	100%	13.57%	**56.8%**
ATM_06	100%	13.57%	**56.8%**
ATM_07	100%	32.14%	**66.1%**

The researchers of "A Study of Reusability, Complexity and Reuse Design Principles" [12] used a 5 point likert scale very similar to a another presented research [26] to interpret the value of reusability: (1 – Not Used, 2 – Difficult to Reuse, 3 – Neither Difficult not easy to reuse, 4 – Easy to Reuse, 5 – Very Reusable.) they measured reusability of a component as perceived by subjects reusing the components. Modifying the likert scale for the reusability equation yielded the results seen in Table 3 which includes how each test case should be interpreted. Note that the value of overall reusability can never reach 0 since the independence value can never be 0.

Percentage Range	Likert Scale Point
(0% ─ 20%]	**Not Used**
(20% ─ 40%]	**Difficult to Reuse**
(40% ─ 60%]	**Neither Difficult nor easy to reuse**
(60% ─ 80%]	**Easy to Reuse**
(80% ─ 100%]	**Very Reusable**

Test case ATM_01 is considered very reusable mainly due to the value of 81.3% reusability falling in the range of 80% - 100% in the Likert scale. Test cases ATM_02 and ATM_03 have a reusability percentage of 58.2% and thus are considered neither difficult nor easy to reuse. Test cases ATM_04, ATM_05 and ATM_06 also fall into the same category since their values 57.3%, 56.8% and 56.8% respectively are in the scale of neither difficult nor easy to reuse. The last test case (ATM_07) is considered easy to reuse since the value 66.1% is over 60% and below 80%.

Considering all the values generated, it enhanced the main idea that these test cases were not optimized and expected to similar results in case of reusability. Depending on the type of optimization the values may change the overall reusability.

v. Conclusion

In the early stages of the research many factors were discovered that had an effect on measuring test case reusability. A Study of Reusability, Complexity and Reuse Design Principles [12] introduced overall Understandability and Complexity as main factors of measuring reusability. However the method used for measuring Understandability was via surveys and human perception, meaning that the degree of understandability was measured via how the testers presumed it was and not based on solid measurement model. The Complexity factor was mentioned in several other researches [12], [17], [19], [20] as a prominent factor that affects reusability. This factor was later explored and expanded upon in the solutions section and following previous research and calculation methods such as Cyclomatic Complexity [17] and Hybrid Complexity measurement models [19], [20] influenced the direction of which this research used to measure Simplicity (which is defined as reverse complexity). Other factors such as Changeability, Universal and Independence were elaborated and measured in Test Case Reusability Metrics Model [4], but some such as Changeability and Universal were dismissed due to the fact that they were limited to the scope of their research and test case generation method, and they were not compatible with the objectives and scope of this research.

When the potential factors of Reusability were identified, two of the prominent factors were chosen and expanded upon for use in measuring test case reusability. A template metrics system was generated that could measure test case reusability of test cases that are generated only in a certain way. The template uses two factors of independence and simplicity as the main criteria for measuring test case reusability. These two factors are considered as a base for the template, and in later editions new factors could be introduced to the system in order to make it more reliable and precise.

In order to evaluate the results of the test case reusability measurement model three main test case samples were used. One of the three samples (ATM Withdraw) [24] was demonstrated as an example in this article.

There are several drawbacks with the proposed solution. The first drawback is a lack of weight system for the factors. This means that there is no weight system in place for the metrics model. In the current model for measuring reusability all the metrics are considered equal in value and weight. This can be rather problematic when a certain organization or developer wants to have a higher emphasis on a certain criterion or factor. Such would be a higher emphasis on simplicity instead of independence. The second drawback is lack of precision due to low number of influential factors. Currently the formula consists of two main criteria and this would make it inaccurate in most cases. This is particularly the case where the simplicity is very low (closer to 10%) and independence is 100%. This would average to 55% and thus consider it an average reusable test case. Although it may very well be, if there were additional variables, the number could have been more precise. The third drawback is limitations due to scope of work. Currently the metrics model required the tester to have background knowledge from the test case generation process. This is mainly because of the white box natures that exist within the simplicity measurement factor. In order to design the test cases the tester needs to know about the Control Flows and Use Case Scenarios which indicate the innate complexity of the test case. Other than the white box requirement, there is also a need for Black Box attributes that would aid in a more precise measurement of simplicity according to the hybrid complexity theories mentioned in several researches.

Proc. of the Intl. Conf. on Advances in Computing and Information Technology-- ACIT 2014

ISBN: 978-981-07-8859-9

VI. Further Study

This research could serve as a template for expanded work on a field that is currently rarely worked upon. There are very few that concentrate on the aspect of singular test case reusability and fewer tend to look for methods of quantifying it. In further researches the drawbacks could be explored more.

New criteria such as changeability and understandability have a higher potential to be considered for the reusability metrics. Both criteria are included in Test Case Reusability Metrics Model, but they are measured based on factors that are not accurate and produce results that are not reliable..

References

[1] S. Schach, Software Engineering, Aksen Associates,Boston, MA, 1990.

[2] Douglas D. Lonngren. Reducing the Cost of Test Through Reuse. AUTOTESTCON98. IEEE Systems Readiness Technology Conference, 1998 IEEE.

[3] Jin-Cherng Lin, Kuo-Chiang Wu. A Model for Measuring Software Understandability. 2006.

[4] Zhang Juan, Cai Lizhi, Tong Weiqing, Yuan Song, Li Ying, Test Case Reusability Metrics Model, 2010 2nd International Conference on Computer Technology and Development (ICCTD), November, 2010.

[5] Institute of Electrical & Electronics Engineers, Standard 610 (1990), reprinted in IEEE Standards Collection: Software Engineering 1994 Edition.

[6] R. van Ommering, Software reuse in product populations, IEEE Transactions on Software Engineering, vol. 31, pp. 537-550, 2005.

[7] P. Mohagheghi and R. Conradi, Quality, productivity and economic benefits of software reuse: a review of industrial studies, Empirical Software Engineering, vol. 12, pp. 471-516, 2007.

[8] P. Mohagheghi and R. Conradi, An empirical investigation of software reuse benefits in a large telecom product, ACM Transactions on Software Engineering Methodology, vol. 17, pp. 1-31, 2008.

[9] W. B. Frakes and G. Succi, An industrial study of reuse, quality, and productivity, Journal of Systems and Software, vol. 57, pp. 99-106, 2001.

[10] M. Morisio, et al., Success and Failure Factors in Software Reuse, IEEE Transactions on Software Engineering, vol. 28, pp. 340-357, 2002.

[11] W. C. Lim, Effects of Reuse on Quality, Productivity, and Economics, IEEE Softw., vol. 11, pp. 23-30, 1994.

[12] R. Anguswamy, W. B. Frakes,. A Study of Reusability, Complexity,and Reuse Design Principles. ESEM'12, September 19–20, 2012.

[13] Frakes W. Systematic., Software Reuse: A Paradigm Shift. In Proceedings of Third International Conference on Software Reuse: Advances in Software Reuse. Los Alamitos, California: IEEE Computer Society Press, 1994

[14] B. Boehm, et al., Cost estimation with COCOMO II, ed: Upper Saddle River, NJ: Prentice-Hall, 2000.

[15] ISOI/EC 9126 Software engineering Product quality, wwwjtclsc7.org.

[16] Jin-Cherng Lin, Kuo-Chiang Wu., A Model for Measuring Software Understandability. 2006.

[17] Thomas J. McCabe. "A Complexity Measure". IEEE Transactions on Software Engineering, Vol. Se-2, No. 4, December 1976.

[18] Anthony Barrett, Daniel Dvorak, A Combinatorial Test Suite Generator for Gray-Box Testing, Third IEEE International Conference on Space Mission Challenges for Information Technology. 2009.

[19] Lingzhong Meng, Minyan Lu, Baiqiao Huang, Xiaojie Xu, Using relative complexity measurement which from complex network method to allocate resources in complex software system's gray-box testing, International Symposium on Computer Science and Society, 2011.

[20] Harrison Warren,Cook Curtis. A micro/macro measure of software complexity[J]. Journal System Software. 1987 : 213-219

[21] IEEE 829-1998 - Software Quality Engineering - Test Case Specification Template - Version 7.0 - 2001

[22] Jim Heumann, Generating Test Cases from Use Cases, The Rational Edge, 2001.

[23] S. Roongruangsuwan, J. Daengdej. Test Case reduction Methods by Using CBR. Autonomous System Research Laboratory. 2010.

[24] Ian Sommerville, Software Engineering, 7th ed. Harlow: England, 2004. Chapter 14.

[25] Pakinam N. Boghdady, Nagwa L. Badr, Mohamed A. Hashim, Mohamed F. Tolba. An Enhanced Test Case Generation Technique Based on Activity Diagrams. 2011 International Conference on Computer Engineering & Systems (ICCES). Nov. 29 2011.

[26] S. R. Nidumolu and G. W. Knotts, The effects of customizability and reusability on perceived process and competitive performance of software firms, MIS Q., vol. 22, pp. 105-137, 1998.