A SOFTWARE TRACEABILITY APPROACH TO SUPPORT REQUIREMENT BASED TEST COVERAGE ANALYSIS

SITI FAIZAH BINTI OMAR

A thesis submitted in fulfillment of the
requirements for the award of the degree of
Master of Science (Computer Science)

Faculty of Computing
Universiti Teknologi Malaysia

MARCH 2013

ALHAMDULLILAH
I dedicated this thesis to
my parents,
Hj Omar Saman and Hjh Siti Rohani
My mothers Asmah Ismail,
Hjh Rosmah M.Y.
My spouse Rusdi Sembak
My children Marissa and Aaron
My sisters Fauziah, Fadilah,
Fairus, Faridah, Fatinah
My brothers Faisal, Marzuqi, Azri
My relatives

ACKNOWLEDGEMENT

I would like to offer my heartiest gratitude to my supervisor, Assoc. Prof. Dr Suhaimi Bin Ibrahim for his continuous encouragement, advice and technical know-how; throughout three years of study. I am also indebted to the respondents from Media Digital Alliance Sdn Bhd for their participation in this research. I would also like to thank RMC UTM and MOSTI staff, who have contributed their time processing and funding the research paper publication. I am grateful to the Advanced Informatics School (AIS), UTM International Campus and Faculty of Computing staff for guiding me on thesis completion and submission.

ABSTRACT

Requirement based test coverage (RBTC) is an important deliverable of a software testing process. There are problems in the process whereby the current RBTC analysis does not integrate with the black and white testing types nor does it generate a multi-direction RBTC analysis report. This research aims to address the problems by investigating RBTC analysis using software traceability and review its usefulness and efficiency. Initially, literature review on the comparison of the existing test coverage approaches and software was conducted followed by the development of a prototype using Java and MySQL. The prototype took into consideration the problems of RBTC analysis and this proposed concept which is RBTC Analysis using software traceability approach was modeled and constructed into a prototype called GRAYzer. Software artifacts from a bank project called 'Fleet Management System' (FMS) were used and embedded into the prototype. Questionnaires and feedback from FMS expert users of the prototype were collected. Data collected include the usefulness rating and time taken by the FMS experts and GRAYzer to do the RBTC analysis. A descriptive analysis of the data showed that a majority of the FMS experts rated the prototype as "Very Useful" and indicated that GRAYzer provided an efficient RBTC analysis. When compared to the test coverage approaches, the prototype provided a forward and backward test coverage analysis which can be used as analysis for any given artifact type. Besides that, it has also integrated gray box coverage types and multi-directions for the RBTC analysis. The research has shown that a software manager could use the prototype to quantify the effort needed by a team member and as a means to visualize the RBTC. However, this research did not cater for RBTC analysis after an artifact change and the source code was not catered for the class inheritance and polymorphism, and these could be viewed as future related works.

ABSTRAK

Liputan ujian berdasarkan keperluan (RBTC) adalah hasil penting dalam proses pengujian perisian. Antara masalah-masalah ketika dalam proses analisa RBTC terkini ialah tiada integrasi antara jenis-jenis pengujian kotak hitam dan putih serta tidak dapat menyediakan laporan analisa RBTC pelbagai arah. Penyelidikan ini bertujuan untuk menyingkap permasalahan analisa RBTC menggunakan jejakan perisian dan meninjau kebergunaan dan kecekapannya. Pada awal penyelidikan, terbitan kajian-kajian terdahulu difahami bagi membandingkan cara-cara liputan ujian dan perisian serta diikuti dengan pembangunan prototaip menggunakan Java dan MySQL. Prototaip diambil kira dalam permasalahan analisa RBTC dan konsep yang dicadangkan iaitu penganalisaan RBTC menggunakan jejakan perisian telah dimodelkan dan dibangunkan menjadi prototaip yang dinamakan sebagai GRAYzer. Artifak-artifak perisian dari projek bank yang dinamakan sebagai 'Fleet Management System' (FMS) digunakan dan dibenamkan ke dalam prototaip. Soal selidik dan maklum balas dari pakar FMS terhadap prototaip dikumpulkan. Data yang telah dikumpulkan termasuk taraf kebergunaan dan masa yang diambil oleh pakar-pakar FMS dan GRAYzer dalam menganalisa RBTC. Analisa deskriptif menunjukkan bahawa majoriti pakar-pakar FMS menarafkan prototaip ini sebagai "Sangat Berguna" dan menunjukkan bahawa GRAYzer dapat menyediakan analisa RBTC yang cekap. Apabila dibandingkan dengan cara-cara liputan ujian, prototaip ini dapat menyediakan analisa liputan ujian secara ke depan dan undur yang boleh digunakan untuk menganalisa pelabagai jenis artifak. Disamping itu, ia dapat mengintegrasi jenis-jenis liputan ujian kotak kelabu serta analisa RBTC pelbagai arah. Kajian menujukkan, dengan menggunakan GRAYzer, seseorang pengurus perisian dapat mengira usaha untuk ahli-ahli kumpulan dan sebagai cara untuk melihat RBTC secara visual. Namun, kajian ini tidak meliputi analisa bagi perubahan artifak dan tidak mengambil cara perwarisan dan polimorfisme, di mana kedua-duanya boleh dilihat sebagai kajian-kajian pada masa hadapan.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ACRONYMS AND SYMBOLS

ACM	- Association for Computing Machinery
ANT	- Another Neat Tool (Apache)
ANTLR	- Another Tool for Language Recognition
AOSD	- Aspect Oriented Software Development
API	- Application Program Interface
ASCII	- American Standard Code for Information Interchange
ATM	- Automated Teller Machine
BCEL	- Byte Code Engineering Library
CFG	- Control Flow Graph
CLS	- Class
COSDG	- Call-based Object Oriented System Dependence Graph
COV	- Coverage
CRS	- Company Reservation System
CTM	- Concern Traceability Metamodel
CxM	- Class to Method
CxR	- Class to Requirement coverage
CxT	- Class to Test cases coverage
CxP	- Class to Package coverage
FAA	- Federal Aviation Administration
FMS	- Fleet Management System
GNU	- GNU Not Unix (eg. Linux)
GUI	- Graphical User Interface
HTML	- Hypertext Markup Language
IDE	- Integrated Development Environment
IEEE	- Institute of Electrical and Electronics Engineering
IT	- Information Technology

ISTQB	- International Software Testing Qualifications Board
JDBC	- Java Database Connectivity
JRE	- Java Runtime Environment
JVM	- Java Virtual Machine
JVMPI	- Java Virtual Machine Profiler Interface
J2EE	- Java 2 Enterprise Edition
LCSAJ	- Line Code Sequence and Jump
LOC	- Line of Code
LSI	- Latent Semantic Indexing
MB	- Megabytes
MCDC	- Modified Condition / Decision Coverage
MSTB	- Malaysian Software Testing Board
MTD	- Method
MxC	- Method to Class
MxP	- Method to Package
MxR	- Method to Requirement
N.A.	- Not Available
OCL	- Object Constraint Language
OS	- Operating Systems
PHP	- Pre-Hypertext Processing
PKG	- Package
PxR	- Package to Requirement
PxC	- Package to Class
RBTC	- Requirement Based Test Coverage
RCT	- Requirement Centric Traceability
RDBMS	- Relational Database Management Systems
REQ	- Requirement
RMS	- Room Management Systems
RTM	- Requirement Traceability Matrix
RTS	- Regression test selection (RTS)
RxT	- Requirement to Test case
RxM	- Requirement to Method
RxC	- Requirement to Class

RxP	- Requirement to Package
SCI	- Source Code Instrumentation
SDD	- Software Design Document
SQL	- Standard Query Language
SRS	- Software Requirement Specification
STD	- Software Test Document
TC	- Test Case
TCM	- Test Coverage Monitoring
TMG	- Term-Document Matrix Generation
TxC	-Test case to Class coverage
TxP	-Test case to Package
TxR	-Test case to Requirement
UAT	- User Acceptance Test
URL	- Uniform Resource Locator
XML	- Extensible Markup Language

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1 Introduction

This chapter presents the research work on requirement based test coverage analysis using a software traceability approach. The discussion in this chapter includes research background, problem statements, objectives and the importance of the study. This is followed by a brief description of the scope of work and structure of the thesis.

1.2 Background of the Research Problem

Many people consider software development and testing as time consuming and expensive. However, due to project resource constraint, people are trying to find efficient and effective ways of testing simply to cut time and cost. Computer software and systems testing services worldwide market is reaching fifty six billion dollar industry by year 2013 (AFP, 2009). The source added that the high demand has resulted in a skills shortage in India and led an increasing numbers of testing services jobs going to China, Malaysia and North Africa. In Malaysia, the Malaysian Software Testing Board (MSTB) has spent $11 million to setup a software testing lab (The Star, 2010). Software testing labs built are meant to support research and development activities and enhance software testing methodologies. To date, there

are twenty eight techniques for regression test selection (RTS) being studied, introduced and published, with the objective to cut down tests of a test suite in order to save time and money. Software testing includes test coverage percentage report. More time is needed to iterate tests and to obtain test coverage percentage.

Requirement based test coverage percentage is normally an indicator of software quality. However, in this research, it can be proven that it can be an indicator for other things. It can also provide judgment for critical resources during testing. Software engineers in an online community were asking the testing experts if requirement traceability matrix (RTM) can be used to find test coverage (Johnson, 2007). There has been similar concept of test coverage using software traceability to the design level (Lormans et. al, 2005) using latent semantic indexing. However, the concept of requirement based test coverage should be extended to the source code level. There is a need to find out if it is feasible or not, using other software traceability technique such as code instrumentation. If it is feasible, then will it be efficient?

1.3 Statement of the Problem

The problem to be resolved by this study is whether software traceability approach will be able to efficiently support requirement based test coverage analysis. The research question is, "How to produce an efficient requirement based test coverage analysis model by using a software traceability approach?"

The sub questions of the main research question are as follow:

i. Why are current test coverage analysis models, approaches and tools still not able to support requirement based test coverage analysis?

ii. What is the best way to capture the requirement based test coverage of software components in the system?

iii. How to measure the requirement based test coverage by the proposed approach?

iv. How to validate the efficiency of test coverage analysis using the software traceability approach?

Sub question (i) will be answered via literature review in Chapter 2. This chapter will provide a special attention to explore the test coverage analysis, its models and traceability issues. From the test coverage analysis perspective, this chapter will present a study on the detailed test coverage analysis processes, the techniques and existing tools used. The strengths and drawbacks are drawn based on a comparison framework in order to propose a new model and approach to support test coverage analysis.

The above study provides some leverage to answer the sub question (ii). Chapter 3 describes a design methodology and evaluation plan before the research is carried out. Sub question (iii) will be counter balanced by a solution to measure the potential effects. The sub questions (ii) and (iii) will be further explained in the traceability modeling and implementation as describe in Chapter 4 and 5. Lastly, sub question (iv) leads to the evaluation of the model and approach quantitatively and qualitatively as described in Chapter 6.

1.4 Objective of Study

The problem statement serves as a premise to establish a set of specific objectives that will constitute major milestones of this research. The objectives of this research are listed as follow:

To study and identify issues in software testing particularly related to requirement based test coverage.

To build a new test coverage analysis model by using software traceability approach that includes requirements, test cases and code.

To verify the concept of the proposed model by developing and using the supporting tools.

To demonstrate and evaluate the efficiency of the requirement based test coverage analysis model using software traceability approach.

1.5 Importance of Study

In the year 2010 to 2012, the world is at the war of talent as it is difficult to get software professionals. In early 2012, CIO.com has reported that Google and Facebook are competing each other for the best brain (The Independent, 2012). When skilled software engineers are difficult to hire, managing projects are likely to be affected as well. On top of that, when software professionals are hired, other companies will ask them if they want to quit the job so they can get higher pay.

This study is conducted on the motivation of saving project resources. By identifying which part of source code map to which part of requirement, it is possible for a manager to identify which project resource is critical. Management of a software house will ask how much can they save on the allowance of the software engineer for outstation tasks? 'Which software engineer shall I bring to handle source code for requirement X, Y and Z?'

Software managers may want to know time or effort needed to deal which each requirement. It is not cheap to assign a software engineers to be at a client site. A manager has to provide outstation allowances which cover travelling cost, lodging and parking. By knowing the percentage of program sections covered for each

requirements, the software manager can quantitatively select which software engineer, should come to trouble shoot for another round of pre-UAT.

The requirement based test coverage provides a way to estimate on how many code portion will a requirement covers. For instance, a requirement such as 'Login', an application uses six methods from a single class. Perhaps, one can say, it takes a ten percent of the overall code sections. If there are five requirements on 'Authentication Module', how many code sections will the application run into? Will it be ten, twenty, fifty or even sixty percent of the code portion?

Which software engineer from a team of four can help to trouble shoot the requirement 1,4,6,9 and 15? Why a software manager should select software engineer X to troubleshoot these five problematic requirements and not software engineer Z?

A requirement based test coverage test coverage analysis can be very useful. It allows tracing the relationship between requirements to source code. Testing experts around the world have agreed on using software traceability (or commonly known as trace matrix) to find test coverage as in (Johnson, 2007),(Copeland, 2009) and (Crispin, 2009). However, an experiment needs to be carried out as a proof of concept. In addition, is the test coverage extendable from requirement to code? Executing all test cases to get test coverage can be time consuming. The optimum solution would suggest that can we apply any approach that can support the requirement and code coverage in a more meaningful and inexpensive manner?

A quick survey conducted using questionnaire indicated that all six experienced Malaysian IT personnel including a project manager and software engineers agreed that other project team members should be able to view coverage percentage. In addition, all agreed to know which part of codes (method, functions) are linked to the requirements. To achieve these, a requirement based test coverage analyzer using study should be deployed.

A laboratory experiment by (Omar and Ibrahim, 2010) which was conducted in software testing lab has shown that a prototype with support of software traceability can aid user in test coverage analysis. However, it is important to know whether this approach make sense among the IT personnel from the industry.

1.6 Scope of Work

First, the scopes of work include verifying the concept of the RBTC analysis model using software traceability in a single software project case study. It is suggested that the best system to consider is to adopt and use the most recent software development system. This research does not reach large, complex or legacy systems. Second, the system documents encompass only to the system requirement document, software testing document and source code. Relevant information includes a set of functional requirements, test cases and source code. It does not include other software artifacts such as the use case and design. Third, the research bounds to analyzing RBTC before implementing any change requests. Prior to change, the test coverage percentage is taken in order to verify the concept of software traceability can support the RBTC analysis. Last, the work does not include the procedural portion but only the object oriented codes.

1.7 Thesis Outline

This thesis covers some discussions on the specific issue associated to software traceability for impact analysis and understanding how this new research is carried out. The thesis is organized in the following outline.

Chapter 2: Discusses the literature review of software testing, software traceability and software coverage. A few areas of interest are identified from which all the

related issues, works and approaches are highlighted. This chapter also discusses some techniques of test coverage method. The discussion on some existing tools that exist in the industry is also included in this chapter. This leads to the improvement opportunities that form a basis to develop newly proposed software of test coverage analysis approach by software traceability method.

Chapter 3: Provides a research methodology that describes the research operation flow from start to end. This chapter includes a research design diagram, which leads to an overview of the data collection and analysis approaches. It is followed by research assumptions.

Chapter 4: Explains the detailed model of the proposed test coverage analysis approach using software traceability approach. A set of software traceability matrix concepts, test coverage finding and calculation algorithm are described. It is followed by some approaches and mechanisms to achieve the model specifications.

Chapter 5: Presents the design and implementation of the prototype as a proof of concept. The design includes the prototype's system architecture, use case, class diagram and sequence diagram. The implementation process includes the development aspect and the user interface. Other supporting programs are discussed at the end of this chapter.

Chapter 6: Specifies the evaluation process of the prototype for its efficiency. The evaluation criteria and methods are described and implemented on the model that includes modeling validation, a case study and experiment. This research performs evaluation based on quantitative and qualitative results. Quantitative results are checked against the time taken by the experts to perform a set of test coverage analysis questions, with and without the support of the prototype. Qualitative results are obtained based on user perception on the prototype's usefulness and comparative study made on the existing models and approaches.

Chapter 7: Summarizes the research achievements, contributions and conclusion of the thesis. This is followed by the research limitations and suggestions for future work.

REFERENCES

AFP (2009). *Software testing market resilient despite crisis: report*. Singapore. Mar 10, 2009. http://www.google.com/hostednews/ afp/article/ ALeqM5hAfn1MgC5beeA3aSQRbYHHCRh0OQ

Answers.com (2009). http://www.answers.com/topic/code-coverage

Asuncion, H., Francois, F., et al. (2007). An End-To-End Industrial Software Traceability Tool. *Proceeding of the 6th Joint Meeting of the ESEC/FSE*. Dubrovnik, Crotia, Sep, 2007.

Atlassian. (2009) http://www.atlassian.com/software/clover/screenshots/

Beizer, B. (1990). *Software Testing Techniques*. 2nd edition, New York: Van Nostrand Reinhold.

Berg, v.d. K., Conejero, J.M., Hernández, J. (2006). Analysis of crosscutting across software development phases based on traceability. *Proceedings of the 2006 international workshop on Early aspects* at ICSE, Shanghai, EA '06. 43 - 50.

Black, R. (2004). *Critical Test Processes: Plan, Prepare, Perform, Perfect*. Pearson Education. 301-304.

BS7925-2 (1997). Standard for Software Component Testing. *British Computer Society SIGIST*.

Buy, U., Orso, A., Pezze, M. (2000). Automated Testing of Classes. *ISSTA'00.* Portland, Oregon. ACM.

Chilenski, J.J and Miller, S.P. (1994). Applicability of Modified Condition/Decision Coverage to Software Testing, *Software Engineering Journal.* September 1994, Vol. 9, No. 5, 193-200.

Cleland-Huang, J. (2005). *Toward Improved Traceability of Non-Functional Requirements*. TEFSE, November 8, 2005, Long Beach, California, USA. ACM 1-50503-243. 14-19.Cobertura. (2009). http://cobertura.sourceforge.net/sample/

Copeland, L. (personal communication, August 1, 2009)

Costello, R.J. (1995). Metrics for requirements engineering. *Journal of Systems and Software*. Volume 29, Issue 1, April 1995, Pages 39–63

Crispin, L. (2009) www.coderanch.com February 3, 2009

Cysneiros, G., Zisman, A. (2008). *Traceability and Completeness Checking for Agent-Oriented System. SAC'08*.Brazil.71-77.

Eclemma. (2009). http://www.eclemma.org/userdoc/coverageview.html

Egyed, A. (2003). A Scenario-Driven Approach to Trace Dependency Analysis, *IEEE Transactions on Software Engineering*, vol 29(2).

Emma. (2009) http://emma.sourceforge.net/coverage_sample_a/index.html

Gieszl, L.R. (1992). Traceability For Integration. *Proceedings of the Second International Conference on Systems Integration*. Volume , Issue 15-18 Jun 1992, New Jersey, pp. 220 – 228.

Glass, R.L. (2009) . A Classification System for Testing, Part 2. *IEEE Software*. Lane, Piscataway, NJ. 103-104

Grinwald, R., Harel, E., Orgad, M., Ur S., Ziv, A. (1998). IBM Research Lab, Haifa. Dac 1998. San Francisco, CA USA. 158-163.

Gupta, R., Harrold, M.J., and Soffa, M.L. (1992). An approach to regression testing using slicing. *Proceedings in Conference on Software Maintenance 1992* (Cat.No.92CH3206-0). IEEE Comput. Soc. Press, 299-308.

Harrold, M.J., McGregor, J.D., Fitzpatrick, K.J. (1992). Incremental testing of object-oriented class structures. *Proceedings of the 14th international conference on Software engineering, ICSE '92*, Melbourne, Australia, 68 - 80.

Hayhurst, K.J., Veerhusen, D.S., Chilenski, J.J., Rierson, L.K. (2001). *A Practical Tutorial on Modified Condition/Decision Coverage*. Langley Research Center, National Aeronautics and Space Administration (NASA).

Hoffman, H.F., Lehner, F. (2001). Requirements engineering as a success factor in software projects. *Software, IEEE*. Jul/Aug 2001. Volume 15. 58-66.

Hoffman, M., Kuhn, N., Weber, M., Bittner, M. (2004). Requirements for requirements management tools. *Proceedings of the Requirements Engineering Conference*, 2004.. 12th IEEE International, Kyoto, Japan, pp. 301-308.

Horgans, J. R., London, S., Lyu M.R. (1994) Achieving Software Quality with Testing Coverage Measures. *IEEE Xplore*.

Huang, J.C. (1975). *An Approach to Program Testing. Computing Surveys*. Vol. 7, No. 3 September.

Ibrahim, N., Wan Kadir, W.M.N., Deris, S. (2008). Comparative Evaluation of Change Propagation Approaches towards Resilient Software Evolution. *Proceeding of The Third International Conference on Software Engineering Advances, 2008. ICSEA '08*, Sliema, Malta, 198-204.

Ibrahim, S. (2006). *A Document-Based Software Traceability to Support Change Impact Analysis of Object-Oriented Software.* Ph.D Tesis. Universiti Teknologi Malaysia, Skudai.

IEEE. (1998a). *IEEE Standard For Software Maintenance*. New York, IEEE Std. 1219-1998.

IEEE (1998) *IEEE Standard for Software Verification and Validation*. 1012-1998. IEEE Press, Piscataway, N.J., 1998.

IEEE (1990), *Standard Glossary of Software Engineering Terminology*. 610.12 . IEEE Press, Piscataway, N.J., 1990.

ISO/IEC 14764. (2009). http://www.iso.org/iso/ catalogue_detail.htm? csnumber=39064

ISTQB (2007). *Standard glossary of terms used in Software Testing*. www.istqb.org: International Software Testing Qualifications Board. Version 2.0, December, 2nd 2007

Jfeature. (2009). https://jfeature.dev.java.net/.

Johnson, Karen N. (2007) searchsoftwarequality.techtarget.com, February 13, 2007

Kaner, C., Brian, L. (2000). Grey box testing. *Los Altos Workshop on Software Testing* #9. Sunnyvale, CA, March, 2000.

Kapfhammer, G.M., Soffa, M.L. (2008). Database-Aware Test Coverage Monitoring. *Proceedings of the 1st conference on India software engineering conference ISEC'08*. Hyderabad, India, ACM.77-86.

Kessis, M., Ledru, Y. Vandome, G. (2005). Experiences in Coverage Testing of a Java Middleware. *SEM 2005*, Lisbon, Portugal. ACM. 39-45.

Kichenham, B.A., Travassos, G.H., Mayrhauser, A.V. and Schneidewind, N. (1999). Towards an Ontology of Software Maintenance. *Journal of Software Maintenance: Research and Practice.* 11:365-389.

Lazaro, M. and Marcos, E. (2005). Research in Software Engineering: Paradigms and Methods. *Proceedings of the 17th International Conference on Advanced Information System (CAiSE'05),* Porto, Portugal, June 2005.

Lingampally, R., Gupta, A., Jalote, P. (2007). A Multipurpose Code Coverage Tool for Java. *Proceedings of the 40th Annual Hawaii International Conference on System Sciences, IEEE Computer Society*. 261b, 2007.

Li, Y., Li, J., Yang, Y., and Li, M. (2008). *Requirement-Centric Traceability for Change Impact Analysis: A Case Study. In Change,* Q. Wang, D. Pfahl, and D. M. Raffo, eds. (Springer-Verlag), pp. 100-111.

Liu, S., Chen, Y. (2008). Model-Based Software Testing. *Journal of Systems and Software*. Volume 81, Issue 2, February 2008, Pages 234–248.

Lormans, M., Deursen, A.v. (2005). Reconstructing Requirement Coverage Views from Design and Test using Traceability Recovery via LSI. *TEFSE 2005.* Long Beach, California, USA. ACM 2005.

Lyu, M.R., Bellcore, Horgan, J.R., London, S. (1994). A coverage analysis tool for the effectiveness of software testing. Reliability, *IEEE Transactions*. Dec 1994. Volume: 43, Issue: 4. 527-535.

Maletic, J.I., Collard, M.L., Simoes. (2005). An XML Based Approach to Support the Evolution of Model-to-Model Traceability Links. *Proceedings of 4th International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE'05,* Long Beach, California, November 8, 2005.

Marcus, A., Xie, X., Poshyvanyk, D. (2005). When and how to visualize traceability links? *Proceedings of the 3rd international workshop on Traceability in*

emerging forms of software engineering, TEFSE '05, Long Beach, California, November 8, 2005, 56 - 61.

Marick, B. (1985). *The Craft of Software Testing, Subsystem testing Including Object-Based and Object-Oriented Testing*. Prentice-Hall.

Munson, E.V., Nguyen, T.N. (2005). Concordance, conformance, versions, and traceability. *Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering, TEFSE '05*. Long Beach, California, November 8, 2005, 62 - 66.

Myers, G.J. (1979). *Art of Software Testing*. John Wiley & Sons, Inc., New York, NY, 1979

Najumudheen, E. S. F., Mall, R. and Samanta, D. (2011). Test coverage analysis based on an object-oriented program model, *Journal of Software Maintenance and Evolution: Research and Practise*, 2011, 23:465–493.

Neumuller, C., Grunbacher, P., Automating software traceability in very small companies: A case study and lessons learned, in *ASE'06. IEEE CS*, 2006, 145–156.

Ntafos, S. (1998). A Comparison of Some Structural Testing Strategies. *IEEE Transaction, Software Engineering*, Vol.14, No.6, June 1988, 868-874.

Oliveto, R., Antoniol G., et. al. (2007). Software Artefact Traceability: The Never-Ending Challenge. *Proceeding of the the 23rd International Conference on Software Maintenance.* Paris, France. IEEE Computer Society, Paris, Oct 2-5, 2007.

Omar, S.F, Ibrahim, S. (2009). A Preliminary Study on Finding a Tool That Uses Software Traceability Approach to Support Test Coverage. *Proceeding of the 5th Postgraduate Annual Research Seminar. PARS'09.* UTM, Skudai, Malaysia.

Omar, F. and Ibrahim, S. (2010). Designing Test Coverage for Grey Box Analysis. *Proceeding of the IEEE 2010 10th International Conference on Quality Software (QSIC2010).* Zhangjiajie, China, 14-15 July 2010. 53-356.

Omar, F and Ibrahim, S. (2011). A Requirement Based Test Coverage Analysis using Software Traceability Approach. *International Journal of Information Technology & Computer Science (IJITCS).* ISSN (Online) : 2091 – 1610.

Pierce, R.A. (1978). A Requirements Tracing Tool. *ACM Software Engineering Notes.* Nov 1997, 67-71.

Prather, R.E. (1984a). An Axiomatic Theory of Software Complexity Measure. *The Computer Journal,* 27(4):340-347.

Prather, R. E. (1984). Theory of Program Testing – An Overview. *Bell System Technical Journal,* Vol. 62, No. 10, 1984, pp. 3073–3105.

Rapps, S. , Weyuker, E.J. (1985). *Selecting software test data using data flow information.* IEEE Transactions on Software Engineering, 11:367 – 375.

Robson, C. (2002). *Real world research: A resource for social scientists and practitioner-researchers*. Wiley, Oxford: Blackwell, UK.

Rochimah, S., Wan Kadir, W. M.N., Abdullah A.H. (2007). An Evaluation of Traceability Approaches to Support Software Evolution. *Proceeding of the International Conference on Software Engineering Advances (ICSEA 2007).* IEEE.

Roper, M. (1994). *Software Testing*. London, McGraw-Hill Book Company.

Runeson, P., Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empir Software Eng*. 14: 131-164.

Seaman, C. B., (1999). Qualitative Methods in Empirical Studies of Software Engineering. *IEEE Transactions on Software*. 25(4):557–572.

Seo, K.I, Choi, E.M (2006). Comparison of Five Black-box Testing Methods for Object-Oriented Software. *Proceedings f the Fourth International Conference on Software Engineering Research, Management and Application (SERA'06)*.

Tekinerdogan, B., Hoffman, C., Aksit, M., (2007). Modeling traceability of concerns in architectural views. *Proceedings of the 10th international workshop on Aspect-oriented modeling, AOM '07*. 49 - 56.

Tonella, P. (2004). Evolutionary testing of classes. *Proceedings of the 2004 ACM SIGSOFT international symposium on Software testing and analysis ISSTA '04*.

The Independent. (2012). Facebook v Google: The tech tug of war. Saturday, 17 March, 2012. http://www.independent.co.uk/life-style/gadgets-and-tech/news/facebook-v-google-the-tech-tug-of-war-7575934.html

The Star (2010). MSTB sets up RM11mil software testing lab. Steven Patrick. Wednesday April 7, 2010. http://thestar.com.my/news/story.asp?file=/2010/4/7/technology/20100407161544&sec=technology

Whalen, M.W, Rajan, A., Heimdahl, M.P.E, Miller, S.P. (2003). *Coverage Metrics for Requirements-Based Testing*. ACM. ISSTA'06, July 17–20, 2006, Portland, Maine, USA.

Wikipedia. (2009). http://en.wikipedia.org/wiki/Code_coverage

Woodward, M.R., Hedley, D.(1980) and Hennell, M.A., Experience with Path Analysis and Testing of Programs. *IEEE Transactions on Software Engineering*, May 1980, Vol. SE-6, No. 3, 278-286.

Yin, R. K. (2003). *Case study research: Design and methods* (3rd ed.). Thousand Oaks,CA: Sage.

Zhou, x., Huo, Z., Huang, Y., Xu, J. (2008). *Facilitating Software Traceability Understanding with ENVISION*. IBM China Research Lab