TEST CASE GENERATION OPTIMIZATION USING COMBINATION PREDICTION MODEL AND PRIORITIZATION IDENTIFICATION FOR COMPLEX SYSTEM RETESTING

NURUL SHAZANI BINTI SAHIDAN

UNIVERSITI TEKNOLOGI MALAYSIA

TEST CASE GENERATION OPTIMIZATION USING COMBINATION PREDICTION MODEL AND PRIORITIZATION IDENTIFICATION FOR COMPLEX SYSTEM RETESTING

NURUL SHAZANI BINTI SAHIDAN

A thesis submitted in fulfilment of the
requirements for the award of the degree of
Masters in Science (Computer Science)

Faculty of Computing
Universiti Teknologi Malaysia

APRIL 2017

All praises to Allah the Almighty for

the strengths and His blessing in completing this thesis.

Specially dedicated to;

my beloved husband Mohd Farhan bin Ahmad

my adorable daughter Nur Jannah Humairah

my wonderful parents Sahidan Zakaria and Noorhayah Che Halim

my understanding parents in law Ahmad and Jamilah

my precious siblings Shahida, Shahrin, Shafizan, Shahanim, Shahiruddin, Shakirin

ACKNOWLEDGEMENT

"In the name of Allah, the most Gracious and the most Merciful"

This thesis would have not been completed without the help and splendid support from many individuals and teams. Firstly, my sincere gratitude goes to the backbone of my research, my supervisors, Dr. Hishammuddin Asmuni and Dr. Muhamad Razib Othman for their excellent supervision, knowledge, belief, patience and interest in the work and for pushing me farther than I thought I could go. To my beloved parents, thank you for always being there and never fail to give me words of encouragement. Your endless support is making me for who I am today. To my dearest colleagues and research-mates, I am thankful for the friendship, supportive comments and ideas in reviewing each other's works and also fun times throughout last three and half year of this study. To my precious close friends and loved ones, thank you for helping me surviving all the stress and not letting me giving up. To everyone who has consistently giving support and advice directly or the other way round, including the team of Software Engineering Research Group (SERG) and GATES IT Solutions Sdn. Bhd., I refer my appreciation. My greatest thanks should also credit to all lecturers at the Faculty of Computing, Universiti Teknologi Malaysia (UTM) for their understanding and support. Last but not least, I appreciate the financial support from the GATES IT Solutions Sdn. Bhd. under GATES Scholar Foundation (GSF) and Malaysian Ministry of Higher Education under MyMaster funds.

ABSTRACT

Nowadays, the retesting process has become crucial in assessing the functionality and correctness of a system in order to ensure high reliability. Although many techniques and approaches have been introduced by researchers, some issues still need addressing to ensure test case adequacy. To determine test case adequacy, it is crucial to first determine the test set size in terms of number of test cases to prevent the system from failing to execute. It is also crucial to identify the requirement specification factor that would solve the problem of insufficiency and scenario redundancy. To overcome this drawback, this study proposed an approach for test case generation in the retesting process by combining two models, which would reveal more severe faults and improve software quality. The first model was enhanced through determining the test case set size by constructing a predictive model based on failure rate using seed fault validation. This model was then extended to requirement prioritisation. Next, it was used to schedule the test cases that focus on Prioritisation Factor Value of requirement specifications. The Test Point Analysis was used to evaluate test effort by measuring level of estimation complexity and by considering the relationship among test cases, fault response time, and fault resolution time. This approach was then evaluated using complex system that called as Plantation Management System as a project case study. Data of Payroll and Labour Management module that applied in 138 estates been collected for this study. As a result, the test case generation approach was able to measure test effort with *High* accuracy based on two combination model and it achieved a complexity level with 90% confidence bounds of Relative Error. This result proves that this approach can forecast test effort rank based on complexity level of requirement, which can be extracted from early on in the testing phase.

ABSTRAK

Pada masa ini, proses pengujian semula adalah dianggap penting dalam menilai fungsi dan ketepatan sesuatu sistem bagi memastikan tahap kebolehpercayaan yang tinggi. Walaupun pelbagai teknik dan pendekatan telah diperkenalkan oleh para penyelidik sebelum ini, terdapat beberapa isu yang masih perlu ditangani untuk memastikan kecukupan kes ujian. Bagi menentukan kecukupan kes ujian, perkara pertama yang perlu dipastikan adalah menentukan kecukupan saiz kes ujian, iaitu bilangan kes-kes ujian yang diperlukan untuk mencegah kegagalan sesuatu sistem. Selain itu, perkara kedua yang perlu diberi perhatian adalah mengenal pasti faktor keperluan spesifikasi yang boleh membantu menyelesaikan masalah kekurangan dan mengatasi masalah lebihan senario. Oleh itu, bagi mengatasi kelemahan-kelemahan tersebut, kajian ini mencadangkan satu pendekatan baharu bagi menjana kes ujian dalam proses pengujian semula dengan menggabungkan dua buah model. Model pertama dipertingkatkan melalui penentuan saiz kes ujian dengan membina model ramalan berdasarkan kadar kegagalan dengan menggunakan benih kesalahan pengesahan. Model ini kemudiannya diperluas kepada keutamaan keperluan, kemudian digunakan untuk menjadualkan kes-kes ujian dengan menggunakan nilai faktor keutamaan bagi spesifikasi keperluan. Ujian Analisis Titik juga telah digunakan bagi mengukur tahap anggaran kerumitan dan mempertimbangkan hubungan antara kes-kes ujian, masa tindak balas kerosakan, dan masa penyelesaian kerosakan. Pendekatan ini kemudian dinilai menggunakan sistem yang rumit, dikenali sebagai sistem pengurusan ladang bagi kajian kes projek. Data berkaitan modul pengurusan gaji dan tenaga kerja yang digunakan di 138 ladang telah dikumpul untuk digunakan dalam kajian ini. Hasil daripada kajian ini, pendekatan penjanaan kes ujian dapat mengukur kesungguhan ujian dengan ketepatan yang tinggi berdasarkan gabungan dua model ini dan mencapai tahap kerumitan dengan 90% batas keyakinan Ralat Nisbi. Keputusan ini membuktikan bahawa pendekatan tersebut boleh digunakan untuk meramal taraf ujian kesungguhan berdasarkan tahap kerumitan keperluan, yang boleh diambil kira dari awal semasa dalam fasa pengujian.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS

H	-	High
L	-	Low
M	-	Medium
VH	-	Very High
VL	-	Very Low

LIST OF ABBREVIATIONS

4FTC	-	Four Factor Test Case
AE	-	Accuracy Estimation
CP	-	Customer-assigned priority
CPU	-	Central Processing Unit
CR	-	Change Request
DR	-	Defect Request
DRL	-	Defect Request Logical
DRS	-	Defect Request Syntax
ERP	-	Enterprise Resource Planning
FPA	-	Function Point Analysis
IC	-	Developer-Perceived Implementation Complexity
IT	-	Incomplete Task
LSE	-	Least Square Estimation
MLE	-	Maximum Likelihood Estimation
MRE	-	Mean Relative Error
MSF	-	Mean Square Faults
NC	-	No Category
PC	-	Process Configuration
PFV	-	Prioritization Factor Value
PMS	-	Plantation Management System
QC	-	Quality Control
RE	-	Relation Error
RT	-	Requirement Traceability

RV	-	Requirement Volatility
SLA	-	Service Level Agreement
TPA	-	Test Point Analysis
TSFD	-	Total Severity Fault Detection
UAT	-	User Acceptance Test

CHAPTER 1

INTRODUCTION

1.1 Background

Software testing is a form of investigation that is conducted to provide stakeholders with information about the quality of the product or service under test. Boehm *et al.* (2003) states that software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include the process of executing a programme or application with the intent of finding software bugs (errors or other defects), and to verify that the software product is fit to use. In this part, the software tester and business analyst play important roles to make sure all features of the application given by the end user works correctly. Usually, the software tester will execute manual testing to detect any defects in the system. To ensure completeness of testing, the tester often follows a written test plan that leads them through a set of important test cases. With this informal approach, the tester does not follow any rigorous testing procedure, but rather explores the user interface of the application using as many of its features as possible, and using information gained in prior tests to intuitively derive additional tests. The success of exploratory manual testing relies heavily on the domain expertise of the tester, because a lack of knowledge will lead to incompleteness in testing.

Large-scale engineering projects that rely on manual software testing follow a more rigorous methodology in order to maximise the number of defects that can be found. A systematic approach focuses on predetermined test cases and generally involves steps such as prediction of test case size and prioritisation of test case running number. A major issue when dealing with incomplete testing is the shortage of taxonomies to achieve a satisfactory level of information about defects in the requirements phase. Thus, it can be concluded that no matter how good the subsequent phases are, the quality of the requirements phase will still be the main determinant that affects the overall quality of the subsequent phases, including the testing phase of the software development process. The Software Requirement Specification (SRS) document defines the capabilities of the provided software (Alshazly *et al.*, 2014). Therefore, if an analyst or a developer does not share the same understanding regarding the requirements, the outcome of the development process will not satisfy customer needs (Gutierrez *et al.*, 2004).

Besides that, the software tester usually generates the test suites based on a coverage criterion that is given without considering the issue of ensuring that parts of the model are exhaustively exercised. Implementation of impracticable testing will allow errors to occur when generating many or way too many test cases in a set of test suites. This situation leads to a decrease in the likelihood of selected test suites uncovering errors in the implemented system. For efficiency of the testing process, the goal is to choose test cases from the test suite in order to establish the correctness of the modification (Gutierrez *et al.*, 2004). Such test suite reuse, in the form of retesting, takes up as much as one-half of the cost of software maintenance (Boehm *et al.*, 2003). For this reason, researchers have considered various techniques to reduce the cost of the retesting process. Retesting is used to verify alternations and to ensure that the changes have not corrupted other functionalities of the software (Peng *et al*, 2014). Logically, the increment in test cases size will lead to increased project size and time. Due to this reason, project development will face difficulty in predicting and managing time (Xie *et al*, 2003). It will also become the main problem to address when the testing deals with a vast and complex system. As is generally known, the test cycle is an important factor in testing; late changes or additions to function at the final moments of the testing phase can incur high costs in the execution process. Therefore, intelligent planning and decision-making must be thoroughly done throughout the generation of the test case in order to achieve optimisation.

In addition, the test cases must also be prioritised to the best new positions based on software requirement specifications, so as to reduce the need for additional test efforts. These techniques let testers order their test cases so that the test cases with the highest priority are executed earlier in the retesting process (Peng *et al.*, 2014). Besides that, test maturity has also become a significant factor that effects project size. Test maturity is the process that is done to ensure a system achieves stability. In the progress of reaching stability, changes in the function will keep occurring in the project development life cycle and this can increase the test execution phase and time pressure (Elbaum *et al*, 2001).

1.2 Challenges in Modelling Test Case Generation

Even with the current technology and sophisticated tools for generating a test case for testing areas nowadays, there are still issues and challenges that the researcher needs to address in order to reduce test efforts and save time and cost. Techniques like root cause analysis and orthogonal defect classification are some of the commonly used practices.

Firstly, there is a significant challenge in identifying the adequate test case size. The number of test case size must be taken into consideration when dealing with system testing. The application of a few test cases or overdrawn test cases can result to inaccuracy rate of fault detection. As a consequence, the assumption for the test case size is important for improving the ability of detecting the fault, thus reducing the cost and time for the retesting process. There are some defects that affect the option selection of test case size, which is due to lack of knowledge on the part of the tester in defining multiple scenarios when dealing with the requirement phase or indequate testing executed by the internal user.

The second challenge involves planning for the test cases to be executed to achieve the performance goal. The important of increasing the performance goal can be done by producing high rates of fault detection in the system. With this, when the test case is executed based on complexity, the fault can be detected at the earliest time in the testing phase (Krishnamoorthi & Sahaaya 2009). The best way to assist the testing process is by prioritising its functionality based on the requirement criteria. Prior planning of prioritisation is one of the test strategies that contribute to improving the rate of fault detection with the aim of increasing the performance and bettering the quality in the testing phase (Boehm *et al.,* 2003). The idea of this technique is to release the test case with higher priority so it is executed first, which is then followed by the lowest priority test case. The level of priority is based on the complexity of the requirement. Since test case prioritisation techniques do not discard test cases, the drawbacks of test case minimisation techniques can be avoided.

Last but not least, approach of generating the test cases can also become a major challenge, particularly in an effort to increase software reliability. Software reliability is the failure probability of the software operation. The lower the percentage of failure of the system, the higher the reliability of the product (Lo & Huang 2006). The important aspects that need to be consider an identification of measures, formulation of theories, capturing of historical data and assessment of how effective those effort estimation models in order to achieve realistic effort estimates for the successful management of software development

1.3 Current Method in Modelling Test Case Generation

In general, the current methods for test case generation can be categorised into three: requirement traceability, probabilistic estimation, and multiple performance metrics.

a) Requirement traceability is the process of mapping between requirement and test cases that will be generated. Practically, test cases are used to demonstrate the flow of the requirement provided by the client. It will be difficult for testers to determine whether or not the requirement is adequate for testing if the test cases have no connection to the individual requirement (Vaysburg, 2001). Traceability, as introduced by Gotel *et al.* (1994), has two different criteria, which are (pre-RS) traceability and (post-RS) traceability. Both represent the encompassing solution and provide the basic framework to illustrate the nature of the issue. The authors successfully proved that poor requirement traceability as a widely reported problem. Since traceability is an important characteristic (Krishnamoorthi and Sahaaya Arul Mary, 2009) and end-to-end traceability is derived from software requirements, test cases and their associated defects in detecting the most severe faults must be discovered at the earliest possible moment in the testing life cycle.

b) Probabilistic estimation is the result of using a retesting model and is used to 'predict'/estimate missing or out-of-sample y-values, where *y* is defined as the dependent variable. In practice, it is very difficult to estimate failure rates with such accuracy. Therefore, Debroy and Wong (2011) introduced the idea of predicting the defect based on historical data. Observation of the failure rates was originally based on the entire set of available test cases against a faulty version of a programme. Historical data can be applied in the requirement specifications by identifying the failure in cumulative number when the test cases are run.

c) The validation metrics measures an organisation's activities and performance. In project management, validation metrics is used to assess the health of the project and consists of the measuring of seven criteria: safety, time, cost, resources, scope, quality, and actions (Neville, 2008). In testing, quality becomes a great concern before a product is released to the customer. One criticism of performance metrics is that the value of information is computed using mathematical methods based on historical data that have been collected from an industrial study.

1.4 Problem Statement

The problem in representing modelling of the test case generation, specifically in the retesting process, is described as follows:

"Given a large and complex system that has issues regarding budgetary cost and time in testing effort, the challenge is to predict test case size in terms of the number of test cases that might suffer from the redundancy process. Also, to prioritise test planning that can detect the most severe faults at the earliest moment in the testing life cycle. Lastly, the method must be able to measure the test effort based on complexity level with a high prediction capability and yielding fairly accurate results."

The first challenge is related to insufficient user story for the testing execution when generating the test cases. Thus, this study aims to predict the test case size of the test cases. Test case size can greatly impact the budgeting of cost and time in testing if testers underestimate the prediction of test case size. Hence, the relationship between test case size and fault detection are both taken into consideration in this study. Poor fault detection will occur if too few test cases are generated and the use of too many test cases might incur expensive cost and result in time constraints.

The second challenge involves no interdependencies between user requirements and their prioritisation based on their relative ranking or grouping on a specified criterion or criteria that may suffice. The current requirements for engineering techniques for prioritisation of software requirements implicitly assume that each user requirement will have an independent and symmetric impact on user satisfaction. For example, it is assumed that implementing a high-priority user requirement will positively impact his/her satisfaction and not implementing a high-priority user requirement will negatively impact his/her satisfaction. For this reason, this study aims to generate the test case based on requirement priority collected from the stakeholders themselves. The factor with high value is recognised as having a complex requirement that must be prioritised in order to increase test efficiency.

The third challenge is the measurement of effort without using any approach in order to estimate effort level. It is important to estimate level of effort to increase the reliability and quality of the product. This study therefore determines the level of test effort based on time lag from the detection and correction of faults that occur in the system. In actual fact, the time taken for bug correction will become shorter if the test effort focuses more on the allocation between the detection and correction of faults. However, in the development life cycle, it is nearly impossible to make the system bug-free but attempts by the developer to reduce the percentage of fault detection will ensure a high reliability for the software.

1.5 Objectives of the Study

The goal of this research is to develop an approach for test case generation for the retesting process using a predictive and prioritisation model to reveal more severe faults and to improve customer-perceived software quality. Therefore, there are three objectives of this study that need to be achieved:

i. To estimate the optimum test case size required to detect the faults in the system
ii. To prioritise test cases based on the requirement specification to increase test effectiveness
iii. To evaluate the test effort and assess its effectiveness by implementing an industrial case study

1.6 Scope of the Study

In order to achieve the objectives stated, the limitations below bind the scope of this study:

a) This study focuses on improving the test case optimisation that can reduce the time and cost for the retesting process.
b) Test prediction is applied to assess the optimisation of the test case based on 669 test case sizes.
c) Test prioritisation is integrated with requirement specifications to determine the adequate test planning before retesting is executed.
d) A Plantation Management System (PMS) that focuses on the labour and payroll module is used in this study to demonstrate the proposed model.
e) Historical data from the User Acceptance Test of the PMS system are taken into consideration for the evaluation phase of this study.
f) The test effort is evaluated via 37 test suites with 651 fault detections in the system.

1.7 Significance of the Study

The significance of this study can be divided into three different categories, which are: (i) System performance; (ii) Software tester; and (iii) In-house software organisation. The benefit of these respective categories is simplified in Figure 1.1:

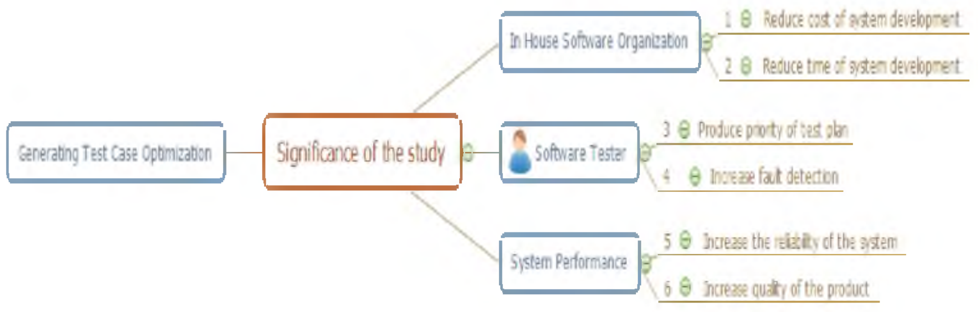
Generating Test Case Optimization
Significance of the study
In House Software Organization
1 Reduce cost of system development
2 Reduce time of system development
Software Tester
3 Produce priority of test plan
4 Increase fault detection
System Performance
5 Increase the reliability of the system
6 Increase quality of the product

Figure 1.1 Significance of the study

1.8 Organisation of the Thesis

This thesis is organised into seven chapters. A brief description of each chapter is given as follows:

i. Chapter 1 defines the challenges, problems, current methods, objectives, scope, and significance of the study.

ii. Chapter 2 reviews the main issues of interest, which include requirement traceability, test prediction model, test case prioritisation, and test effort estimation techniques.

iii. Chapter 3 presents the design of the computational method that supports the objectives of the study. This includes the research framework, data collection, and instrumentation and analysis.

iv. Chapter 4 describes the scheme for developing a prediction model to estimate the reliability of the retesting process and to determine the test case size using failure rate so as to improve fault detection in the system.

v. Chapter 5 discusses the implementation of the test case prioritisation model by considering four factors of requirement complexity.

vi. Chapter 6 evaluates the test effort based on fault detection and correction model by showing the relationship between the flow of test cases, fault response time, and fault resolution time, which are demonstrated using three types of comparative analyses.

vii. Chapter 7 draws the overall conclusions from the obtained results and presents the contributions of the study as well as recommends potential directions for future study.

REFERENCES

Altinel, I. K., Sciences : Profiles A General Software Testing Model Involving, (2001). *Probability in the Engineering and Informational Sciences*, 15,519-533

Amasaki, S., Yokogawa, T. (2012). A study on predictive performance of regression-based effort estimation models using base functional components. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7343 LNCS*, 350–354.

Anand, S., Burke, E. K., Chen, T. Y., Clark, J., Cohen, M. B., Grieskamp, W., McMinn, P. (2013). An orchestrated survey of methodologies for automated software test case generation. *Journal of Systems and Software*, *86*(8), 1978–2001.

Averous, J. (2012). Large Complex Projects. LNCS 7343, 350–354,

Calzolari, F., Tonella, P., Antoniol, G. (2001). Maintenance and testing effort modeled by linear and nonlinear dynamic systems. *Information and Software Technology*, *43*(8), 477–486.

Chikh, A., Aldayel, M. (2012). A new traceable software requirements specification based on IEEE 830. *International Conference on Computer Systems and Industrial Informatics, ICCSII 2012*. New York, USA. pp. 301-307

Catelani, M., Ciani, L., Scarano, V. L., Bacioccola, A. (2011). Software automated testing: A solution to maximize the test plan coverage and to increase software reliability and quality in use. *Computer Standards & Interfaces*, *33*(2), 152–158.

Debroy, V., & Wong, W. E. (2011). On the estimation of adequate test set size using fault failure rates. *Journal of Systems and Software*, *84*(4), 587–602.

Fazlalizadeh, Y., Khalilian, A. (2009). Incorporating Historical Test Case Performance Data and Resource Constraints into Test Case Prioritization. T*ests and Proofs: Third International Conference*, TAP 2009, Zurich 43-57.

Gotel, O. C. Z., Finkelstein, A. C. W., Sw, L. An analysis of the requirements traceability problem (1994).*Journal Software Engineering & Applications*. 3, 869-874.

Hassouna, A., Tahvildari, L. (2010). An effort prediction framework for software defect correction. *Information and Software Technology*, *52*(2), 197–209.

Huang, R., Chen, J., Towey, D., Chan, A. T. S., & Lu, Y. (2015). Aggregate-strength interaction test suite prioritization. *Journal of Systems and Software*, *99*, 36–51.

Jiang, B., Chan, W. K. (2015). Input-based adaptive randomized test case prioritization: A local beam search approach. *Journal of Systems and Software*, *105*, 91–106.

Kapur, P. K., Goswami, D. N., Bardhan, A., Singh, O. (2008). Flexible software reliability growth model with testing effort dependent learning process. *Applied Mathematical Modelling*, *32*(7), 1298–1307.

Kvarnström, B. (2008). Traceability Methods for Continuous Processes. *Environmental Management*. 1402-1757

Kavitha, R., Kavitha, V. R., Suresh Kumar, N. (2010). Requirement based test case prioritization. *2010 International Conference on Communication Control and Computing Technologies*. Madurai, India 826–829.

Krishnamoorthi, R., Sahaaya Arul Mary, S. (2009). Factor oriented requirement coverage based system test case prioritization of new and regression test cases. *Information and Software Technology*, *51*(4), 799–808.

Law, C. C. H., Chen, C. C., Wu, B. J. P. (2010). Managing the full ERP life-cycle: Considerations of maintenance and support requirements and IT governance practice as integral elements of the formula for successful ERP adoption. *Computers in Industry*, *61*(3), 297–308.

Lin, C.-T., Huang, C.-Y. (2008). Enhancing and measuring the predictive capabilities of testing-effort dependent software reliability models. *Journal of Systems and Software*, *81*(6), 1025–1038.

Lo, J.-H., Huang, C.-Y. (2006). An integration of fault detection and correction processes in software reliability analysis. *Journal of Systems and Software*, *79*(9), 1312–1323.

Lunn, K., Sixsmith, A., Lindsay, A., Vaarama, M. (2003). Traceability in requirements through process modelling, applied to social care applications. *Information and Software Technology*, *45*(15), 1045–1052.

Mader, P., Egyed, A., Maeder, P. (2012). Assessing the effect of requirements traceability for software maintenance. *IEEE International Conference on Software Maintenance,* Austria, 171–180.

Marco, L. (2014). Security In Large, Strategic And Complex Systems. *First International Workshop on Signal Processing for Secure Communications* (SP4SC-2015), 25 August 2015, Rome. pp 13-18.

Matende, S., Ogao, P. (2013). Enterprise Resource Planning (ERP) System Implementation: A Case for User Participation. *Procedia Technology*, *9*, 518–526.

May, J., Dhillon, G., Caldeira, M. (2013). Defining value-based objectives for ERP systems planning. *Decision Support Systems*, *55*(1), 98–109.

Morgenshtern, O., Raz, T., Dvir, D. (2007). Factors affecting duration and effort estimation errors in software development projects. *Information and Software Technology*, *49*(8), 827–837.

Peng, R., Li, Y. F., Zhang, W. J., Hu, Q. P. (2014). Testing effort dependent software reliability model for imperfect debugging process considering both detection and correction. *Reliability Engineering & System Safety*, *126*, 37–43.

Rempel, P., Lehnert, S., Kuschke, T., Farooq, Q. U. A. (2013). A Framework for Traceability Tool Comparison. *Softwaretechnik-Trends*, *32*(3), 6–11.

Salem, A. (2010). A Model for Enhancing Requirements Traceability and Analysis. *International Journal of Advanced Computer Science and Applications*, *1*(5), 14–21.

Shahamiri, S. R., Kadir, W. M. N. W., Ibrahim, S., & Hashim, S. Z. M. (2011). An automated framework for software test oracle. *Information and Software Technology*, *53*(7), 774–788.

Sharma, A., Kushwaha, D. S. (2012). Estimation of Software Development Effort from Requirements Based Complexity. *Procedia Technology*, *4*, 716–722.

Simao, A., Petrenko, A. (2011). Generating asynchronous test cases from test purposes. *Information and Software Technology*, *53*(11), 1252–1262.

Srikanth, H., Banerjee, S., Williams, L., Osborne, J. (2014). Towards the prioritization of system test cases. *Software Testing, Verification and Reliability*, *24*(4), 320–337.

Srikanth, H., Drive, T. P., Cohen, M. B (2002). Reducing Field Failures in System Configurable Software : Cost-Based Prioritization. *Software Testing,* *Verification and Reliability*, 320-337

Srikanth, H., Williams, L (2000). Requirements-Based Test Case Prioritization. *Software Testing, Verification and Reliability,* 101-103

Srikanth, H., Banerjee, S. (2012). Improving test efficiency through system test prioritization. *Journal of Systems and Software*, *85*(5), 1176–1187.

Sundaram, S. K., Hayes, J. H., Dekhtyar, A., Holbrook, E. A. (2010). Assessing traceability of software engineering artifacts. *Requirements Engineering*, *15*(3), 313–335.

Temberger, M. I., Kovacic, A. (2008). The Role of Business Process Modelling in ERP Implementation Projects. In *Tenth International Conference on Computer Modeling and Simulation.* Cambridge, UK, 1-3 April 2008, pp. 260 – 265.

Zisman, A., Spanoudakis, G., Pérez-Miñana, E., & Krause, P. (2003). Tracing Software Requirements Artefacts. *The 2003 International Conference on Software Engineering Research and Practice (SERP'03). 2003. Las Vegas*, (August 2015), 1–7.