TESTING MODEL USING RISK POKER TECHNIQUE FOR SCRUM-BASED SOFTWARE DEVELOPMENT PROJECTS

SITI NOOR HASANAH BINTI GHAZALI

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2017

Universiti Malaya

TESTING MODEL USING RISK POKER TECHNIQUE FOR SCRUM-BASED SOFTWARE DEVELOPMENT PROJECTS

SITI NOOR HASANAH BINTI GHAZALI

DISSERTATION SUBMITTED IN FULFILMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SOFTWARE ENGINEERING

FACULTY OF COMPUTER SCIENCE AND INFORMATION TECHNOLOGY
UNIVERSITY OF MALAYA
KUALA LUMPUR

2017

Universiti Malaya

UNIVERSITY OF MALAYA
ORIGINAL LITERARY WORK DECLARATION

Name of Candidate: SITI NOOR HASANAH BINTI GHAZALI

Matric No: WGC130003

Name of Degree: MASTER of SOFTWARE ENGINEERING

Title of Project Paper/Research Report/Dissertation/Thesis ("this Work"): TESTING MODEL USING RISK POKER TECHNIQUE FOR SCRUM-BASED SOFTWARE DEVELOPMENT PROJECTS

Field of Study: SOFTWARE TESTING

I do solemnly and sincerely declare that:

(1) I am the sole author/writer of this Work;
(2) This Work is original;
(3) Any use of any work in which copyright exists was done by way of fair dealing and for permitted purposes and any excerpt or extract from, or reference to or reproduction of any copyright work has been disclosed expressly and sufficiently and the title of the Work and its authorship have been acknowledged in this Work;
(4) I do not have any actual knowledge nor do I ought reasonably to know that the making of this work constitutes an infringement of any copyright work;
(5) I hereby assign all and every rights in the copyright to this Work to the University of Malaya ("UM"), who henceforth shall be owner of the copyright in this Work and that any reproduction or use in any form or by any means whatsoever is prohibited without the written consent of UM having been first had and obtained;
(6) I am fully aware that if in the course of making this Work I have infringed any copyright whether intentionally or otherwise, I may be subject to legal action or any other action as may be determined by UM.

Candidate's Signature Date:

Subscribed and solemnly declared before,

Witness's Signature Date:

Name:

Designation:

Universiti Malaya

ABSTRACT

In agile software development, project estimation often depends on group discussion and expert opinions. Literature claims that group discussion in risk analysis helps to identify some of the crucial issues that might affect development, testing, and implementation. However, risks prioritization often relies on individual expert judgement. Therefore Risk Poker, a lightweight risk based testing methodology which helps to achieve consensus through group discussion in risk analysis that outperforms the individual analyst's estimation is introduced in agile methods. Apart from the above-mentioned benefits Risk Poker can offer, unfortunately no study has been conducted to prove its ability to improve testing process to date. Therefore, this research is aimed at closing this research gap by (i) deploying Risk Poker technique as risk-based strategy in the agile development lifecycle, and (ii) empirically evaluating the proposed test process in providing adequate testing. This research will guide software practitioners in implementing this technique in Scrum-based software development projects. For this purpose, Risk Poker technique is coupled with test coverage for an innovated testing process for Scrum-based software development projects. A case study was conducted with 6 teams of students to estimate test coverage using Risk Poker for an e-commerce system. Three teams estimated their user stories using Risk Poker, while the rest estimated individually and used an average to obtain the statistical combination. The results showed that the proposed usage of Risk Poker mean *BRE* is 0.24, which is lesser compared to the mean *BRE* 0.50 for averaged statistical estimation.

Universiti Malaya

ABSTRAK

Di dalam pembangunan perisian yang menggunakan kaedah agile, penganggaran projek seringkali bergantung pada perbincangan kumpulan. Pelbagai penulisan berpendapat bahawa perbincangan kumpulan membantu mengenalpasti beberapa isu-isu penting yang boleh memberi kesan kepada proses pembangunan, ujian, dan pelaksanaan. Malah, perbincangan secara berkumpulan memberikan hasil analisa yang lebih baik berbanding penganggaran oleh pakar secara individu, selain memastikan kesamarataan dari segi penyertaan di dalam proses analisis. Namun, pengkelasan risiko bagi pembangunan perisian megikut tahap kesukaran sering diputuskan oleh seorang individu profesional. Oleh itu, berdasarkan faedah perbincangan kumpulan, Risk Poker membantu mencapai persetujuan melalui perbincangan kumpulan di dalam penganalisaan risiko di mana hasilnya mengatasi anggaran seseorang penganalisis individu. Selain daripada faedah-faedah Risk Poker yang disebut di atas, malangnya tiada kajian yang telah dilakukan bagi membuktikan kemampuan Risk Poker dalam menambahbaik proses pengujian yang ada pada masa kini. Oleh itu, kajian ini mensasarkan untuk mengisi jurang tersebut melalui (i) perlancaran teknik Risk Poker di dalam kitaran hayat pembangunan agile, dan (ii) penilaian penambahbaikan tersebut dalam analisis risiko secara empirik. Kajian ini akan menjadi panduan bagi pengamal pembangunan perisian bagi mengimplementasikan teknik ini di dalam projek pembangunan perisian berasaskan Scrum. Untuk itu, teknik Risk Poker digandingkan dengan liputan ujian untuk menghasilkan suatu inovasi proses ujian dalam projek agile yang menggunakan Scrum. Sebuah kaji selidik telah dijalankan di mana ia melibatkan 6 kumpulan pelajar, untuk menganggarkan liputan ujian menggunakan teknik Risk Poker ke atas sebuah sistem e-dagang. Tiga kumpulan menggunakan Risk Poker dalam kaedah penganggaran mereka, manakala kumpulan selebihnya menganggar secara individu. Hasil kajian menunjukkan bahawa penggunaan Risk Poker menperolehi purata BRE

Universiti Malaya

sebanyak 0.24, iaitu lebih kurang berbanding purata BRE kaedah penganggaran statistik secara individu sebanyak 0.50.

Universiti Malaya

ACKNOWLEDGEMENTS

I would first like to thank my supervisor Professor Dr. Siti Salwah Salim of the Faculty of Computer Science and Information Technology at University of Malaya. The door to Prof. Dr. Siti Salwah Salim’s office was always open whenever I ran into a trouble spot or had a question about my research or writing. She consistently allowed this research to be my own work, but steered me in the right the direction whenever he thought I needed it.

I would also like to thank Assoc. Prof. Dr. Siti Hafizah Binti AB Hamid who gave lots of help and cooperation in providing her final year students to participate in the experiment validation for this research. Without the students’ passionate anticipation and input, the validation could not have been successfully conducted.

Finally, I must express my very profound gratitude to my parents and my beloved husband Luqman Hakim Bin Baharudin for providing me with unfailing support and continuous encouragement throughout my years of study and through the process of researching and writing this dissertation. This accomplishment would not have been possible without them. Thank you.

Universiti Malaya

TABLE OF CONTENTS

Universiti Malaya

Universiti Malaya

Universiti Malaya

LIST OF FIGURES

Universiti Malaya

LIST OF TABLES

Universiti Malaya

CHAPTER 1: INTRODUCTION

Agile has been a very popular choice for the ever-changing software projects in software industry. It is reported that most agile projects implemented Scrum management process, where according to (One, 2010) 58% of the respondents of agile projects used Scrum, followed by Scrum with Extreme Programming hybrid and others. Thus, this research explores within the scope of Scrum management process for agile projects.

Although agile methodology has been widely chosen as software development methodology in industrial practice, unlike traditional software methodology, agile methodology is lacking a well-documented software testing process as a level of standard that can be used like a text-book guide that could be directly adopted in agile process without any alteration to fit agile process (Karhunen, 2009; Khalane & Tanner, 2013). Currently, agile team needs to modify the established traditional software testing process to fit agile characteristics (Khalane & Tanner, 2013). Compared to agile, traditional method refers to extensive planning, codified processes, and rigorous reuse to make development an efficient and predictable activity (B. Boehm, 2002), in which (Biju, 2010) also translates traditional method as waterfall, spiral and iterative methods.

Upon exploring agile environment in depth, it is revealed that agile methodology explicitly practices risks-based strategies in prioritizing, estimating and analyzing tasks. However, in most situations, prioritization and analysis often relies on individual expert opinions despite one of the most important agile characteristic being self-managed team where it emphasizes on the importance of group discussions or team decision in carrying out software development activities. Thus, upon proposing a testing model that is able to be implemented directly in agile methodology without the need of

Universiti Malaya

modification to fit agile and Scrum self-managed team members, this research identified a technique which is able to make use of group discussion characteristic in achieving consensus in prioritizing and analyzing tasks. The identified technique is called Risk Poker and it is derived from one of the popular techniques used to combine expert opinions in order to determine planning estimation, called Planning Poker (Grenning, 2002), which is widely used in the scrum methodology (Schwaber, 2004; Schwaber & Beedle, 2002).

Recently, Risk Poker technique proposed by (Van de Laar, 2012) has utilize one of the benefit of Planning Poker characteristics, which is group discussion in providing lightweight risk analysis technique. Risk Poker is designed to identify and analyze risks for user stories by achieving group consensus. It is a team-based activity in which decisions are made by achieving an agreement between team members. However, as suggested by (Van de Laar, 2012) there is a need to combine Risk Poker technique with test coverage to provide estimation on how much testing is needed to provide adequate testing.

Thus, by acknowledging that Risk Poker and test coverage will provide adequate testing in agile, this research can support software industry personnel to implement Risk Poker with Planning Poker in planning meeting for Scrum methodology in the future. In addition to that, it is assumed that the specified test coverage estimated by Risk Poker is able to provide adequate testing which is crucial for small iteration-based agile projects. Where at the end of software testing activity, the estimated test coverage can be used as one of the acceptance criteria in claiming that adequate testing has been performed for the agile product.

Universiti Malaya

The use of Risk Poker is a new concept among software practitioners and there is no empirical evidence proving the efficiency of Risk Poker technique for agile-based process. Therefore, this research aims to materialize this technique together with a combination of test coverage through experiment where at the end of this research, a testing model integrated in Scrum is produced with proof of concept through experiment validation. This research will integrate Risk Poker technique with test coverage in the planning phase in agile lifecycle and evaluating this approach to identify whether Risk Poker can provide adequate test coverage estimation for agile based projects. A properly designed study can provide preliminary evidence about this approach's strengths and weaknesses, thus reducing the risks accompanying its adoption in practice.

1.1 Problem Statement

Most of existing ready-to-use software testing process is tailored on fitting into well documented specification-based project. Thus, there is not much study focused on how to fit software testing model in the lightweight agile characteristics effortlessly, without the need of modification (Itkonen, Rautiainen, & Lassenius, 2005). A study by (Khalane & Tanner, 2013) agreed that most of the time, agile team members are required to modify existing process or technique to fit it into agile environment.

The fixed and short duration of agile project development requires a definite answer to how much testing is needed to ensure quality. Hence, test coverage is needed to be used as acceptance criteria for short iterations to define testing completeness. Therefore, this research is combining test coverage with a suitable test strategy as a proposed solution to perform as a ready-to-use testing model for agile projects.

Universiti Malaya

In most cases, risk analysis and tasks prioritization often rely on individual expert estimation where, a study by (Conboy & Coyle, 2009) has discussed risk management thoroughly, in which it also stated that the most dominant sources of risk identification is from Senior/Project Management. This contradicts with one of agile crucial characteristics which is a self-managing team, because in agile, decision should be made by team members through group discussion whereby it is able to promote optimized knowledge and expertise sharing among team members and consequently improve analyses (Moløkken-Østvold & Jørgensen, 2004) and estimation. Thus, this research is going to implement Risk Poker as a risk analysis technique that could perform as a testing model and provide test coverage estimation based on group consensus in risk analysis. However, despite the importance of group discussion in agile environment, literature claims that group discussion could add influence upon decision-making, which will increase optimism in estimation (Armstrong, 2006).

In order to propose a testing model that emphasizes group communication as one of its strengths, this research has identified that Risk Poker, as a risk-based strategy, has high potential to be suitable for agile environment as it promotes group consensus in analyzing risks based on various experiences of experts coming from different backgrounds. Unfortunately, despite the fact that Risk Poker is derived from a popular and widely used technique called Planning Poker, this research has learnt that through extensive literature search, there is no evidence of research study previously has been conducted to prove that Risk Poker is successful to perform as risk analysis.

Universiti Malaya

1.2 Research Motivation

The lack of well-documented standardized testing process that could fit into agile project without modification has motivated this research to explore more on testing in agile scope. Furthermore, software testing courses, techniques and study are more often introduced and validated in traditional environment compared to agile environment. Thus, there is a need to provide more analysis and study on testing process in agile environment as a guide for industrial professional and other interested researcher to explore and discover more research gap that needs to be filled in the future.

Apart from that, this research is motivated to explore and propose a software test strategy that fully utilizes agile characteristics in the proposed model. This is to give distinction and advantage to the proposed model compared to the existing software test strategy that is generally designed for traditional methodology, whereby the proposed model could use the benefit of agile characteristic to improve agile process as a whole. Thus, this research focused on one of the basic agile characteristics, which is a self-managed team wherein group discussion plays important role upon making decision. Based on the principle that many heads think better than one, it is expected that expert group judgment provides better analysis than individual expert does, by making sure all team members participate equally in the analysis process, regardless of their positions.

Lastly, literature has proven that there is a successful staffing effort estimation technique which emphasized on group discussion benefit to provide expert group judgment specifically for Scrum. This fact has continued to motivate this research to focus on providing a testing model that could successfully fit into agile like the staffing effort estimation technique.

Universiti Malaya

1.3 Objectives and Research Questions

The main goal of this research is to propose a testing model which will optimize the group discussion characteristic of agile environment to provide risk analysis and estimate test coverage. It is expected that the proposed testing model will provide adequate testing which will be the acceptance criteria for the product developed. Following are the objectives set to be achieved in this research:

1) To identify a suitable testing strategy for agile projects following Scrum.
2) To identify a suitable test coverage technique to provide adequate test coverage for the identified test strategy in 1.
3) To construct a testing model for agile project following scrum using the research findings in Objectives 1 & 2.
4) To validate the proposed model in terms of better risk analysis to provide test coverage estimation in agile projects.

The first objective will ensure the important criteria in agile environment are taken into account upon identifying relevant technique to provide a suitable software test strategy that can fit into agile projects. Following that, this research identifies suitable test coverage technique which can be combined with the technique found in Objective 1 to provide test coverage for the lightweight Scrum environment. This research constructs a testing model which integrates the proposed test strategy with suitable test coverage that could easily fit into agile projects following Scrum. Lastly, this research validates the proposed model in Scrum project with students as the test subjects in order to collect data and evaluate the effectiveness of proposed model towards providing adequate testing for agile projects.

Universiti Malaya

Below are the research questions for each of the objectives defined above:

Objective 1:

RQ1: How does software test strategy in agile methodology differ from traditional methodology?

This question helped this research to explore agile software development characteristics, the main differences of software development process in agile in comparison to traditional software development process, and identify suitable software test strategy to be used in agile project following Scrum without the need to modify the software test strategy to fit into agile team. It will address the issues and problems faced by software testers in agile environment. Once the differences between executing testing in agile methodology and traditional methodology is identified, the scope of identifying the right software test strategy for agile is narrowed down.

Objective 2:

RQ2: Which test coverage technique could provide adequate testing and suitable for the identified test process in RQ1?

This question explored on software test coverage application in order to find suitable test coverage technique that is able to provide adequate test coverage for the lightweight Scrum project environment. The identified test coverage technique is then combined with the software test strategy found in RQ1 to provide adequate testing estimation for Scrum.

Universiti Malaya

Objective 3:

RQ3: How to integrate the identified testing method with test coverage into a model to provide adequate testing estimation for agile project following scrum?

This research formulates and constructs a testing model for agile projects following Scrum by integrating the identified software test strategy together with test coverage into Scrum process. The affected scrum process is identified and this research designs the proposed model inside the aforementioned Scrum process.

Objective 4:

RQ4: How does risk poker perform better risk analysis and provide adequate test coverage?

This question proved whether the proposed model is able to promote better risk analysis and provide adequate test coverage for agile projects. In order to get the evidence required to prove the proposed model, a group of final year students from Software Verification and Validation class are used to implement the proposed model in an agile project following Scrum methodology. The following two sub-questions are answered respectively;

RQ4.1: Is the test coverage provided by Risk Poker-based proposed model adequate compared to the statistical combination of individuals?

Risk estimation provided by student teams is analyzed through comparison. The Risk Poker estimates are compared with the averaged statistical combination of students' estimates to determine the accuracy of risk analysis.

Universiti Malaya

RQ4.2: *How does Risk Poker-based proposed model estimation differ from the averaged statistical combination of individual estimations?*

Risk estimation provided by student teams is analyzed using SPSS tool to provide evidence that Risk Poker technique does improve test coverage estimation statistically.

1.4 Research Significance

This research creates a significant impact on providing a suitable ready-to-use testing process for agile project following Scrum. Its main contribution is to offer a testing model which could provide adequate test coverage that could fit into agile environment while fully utilizing group discussion in risk analysis as one of agile's important characteristic. The risk analysis provides prioritized tasks and test coverage estimation which are achieved from group consensus instead of relying on risk analysis and estimation by individual expert judgment which could overlook some important aspects while analyzing and calculating the risk factor.

This research also produces a Risk Graph that prioritizes the tasks to be developed and test coverage estimation to help developer and tester identify high-risk items and test coverage for the sprint.

Lastly, the analysis from the validation experiment is expected to be beneficial to professionals in software industry and researchers interested on software testing in agile projects.

Universiti Malaya

1.5 Scope of Research

The proposed model implement a risk-based software test strategy called Risk Poker. Risk analysis obtained from Risk Poker technique is used to estimate test coverage. The proposed model is then integrated in Scrum process as a testing model which provides adequate testing throughout the project.

Next, this research evaluates the proposed model's effectiveness by analyzing the test results of the validation experiment. The validation is done in a small-scale project at university stage, where the experiment environment involves 6 teams of final year students in Software Engineering course taking Software Verification and Validation class. The project duration is 9 weeks which is divided to two-weeks sprints for 3 iterations, for a total of 34 user stories provided by an e-commerce client. The teams of students are able to construct test cases and execute testing accordingly. Even though the validation is done in a small scale of student environment, effort was made to make sure the setting and execution of the experiment is very closely similar to an industry environment like what has been done in previous studies by (Mahnic, 2011; Mahnič & Hovelja, 2012; Molokken-Ostvold & Haugen, 2007; Moløkken-Østvold, Haugen, & Benestad, 2008; Williams, Gegick, & Meneely, 2009). The test coverage estimation conducted in the validation experiment focused on unit testing phase only to measure exposed fault compared to seeded fault in the system under test.

Universiti Malaya

1.6 Outline of the Dissertation

The dissertation is organized into six chapters for easy read-out. The list of chapters in this dissertation and their descriptions are as follows:

Chapter 1 Introduction is the chapter which provides brief descriptions of the research topic, objectives, research approaches and research contribution.

Chapter 2 Literature Review focuses on the research study and literature review of existing researches on agile methodology, testing strategies, test coverage and other studies or researches which adopted similar group discussion estimation or group discussion decision making in order to understand how to design the proposed model in agile project effectively.

Chapter 3 Research Methodology describes the processes, actions and steps that have been taken throughout the research in order to integrate the proposed testing model into the planning meeting and validate its effectiveness for agile projects.

Chapter 4 Model Building describes and explains the proposed testing model in details through design and diagrams for further understanding.

Chapter 5 Validation describes how the experiments and implementation of the proposed model is carried out throughout the research for an agile project. Based on the experiment and model implementation, this chapter also evaluates the effectiveness of integrating risk poker into the planning meeting in order to analyze risk and provide test coverage for agile project.

Chapter 6 Conclusion focuses on the conclusion, contribution and future works of this research.

Universiti Malaya

CHAPTER 2: LITERATURE REVIEW

In this chapter, this research reviews literatures related to the research domain which is agile method software development lifecycle to explore agile characteristics and to identify suitable framework to be integrated with the proposed solution. Next, this research explores on Scrum process to identify which Scrum process that is going to be affected upon integrating the proposed solution. Subsequently, this research reviews literatures of software testing to identify suitable software test strategy to achieve objective 1, and lastly test coverage literatures are reviewed to identify suitable test coverage technique to achieve objective 2.

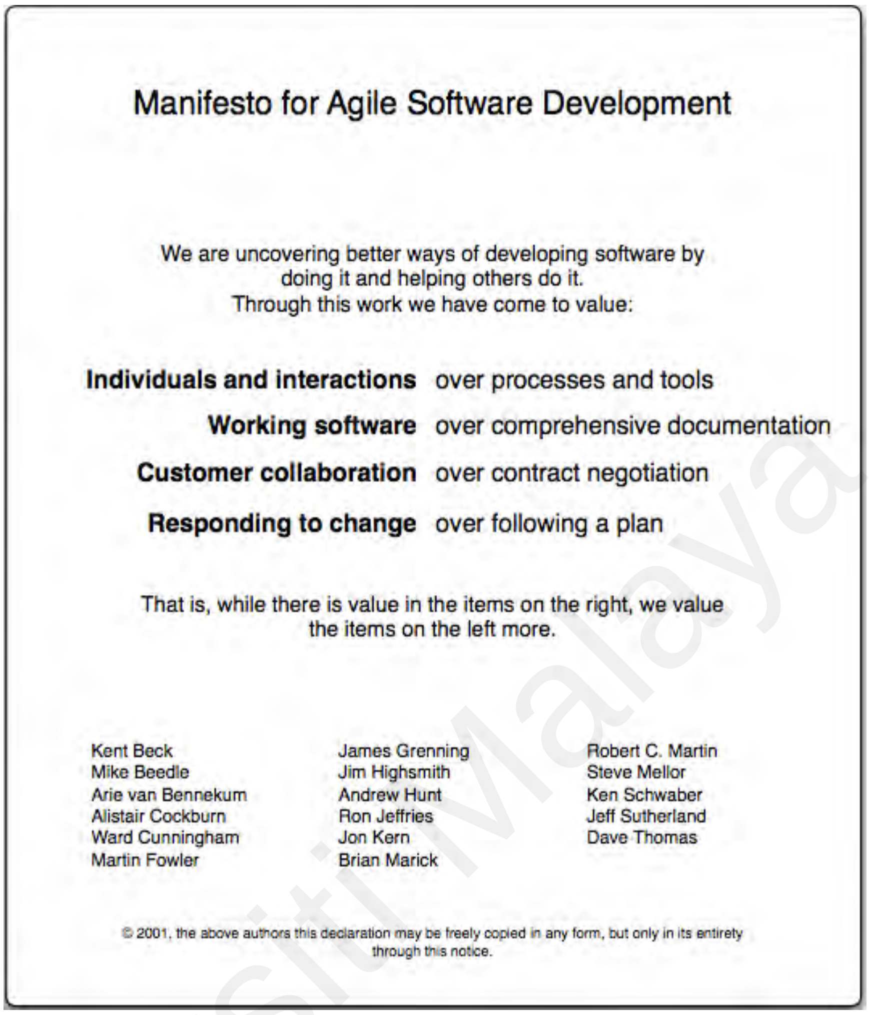
2.1 Agile Software Development Methodology

Agile methodology is another type of software development lifecycle that stresses out the importance of short development with more customer interaction compared to traditional methodology which is waterfall software development lifecycle. It has become an increasingly popular pick in industrial environment (Felker, Slamova, & Davis, 2012) which has been proven when the number of agile adoption in practice has started to multiply in a short time (Eloranta, Koskimies, Mikkonen, & Vuorinen, 2013). The adoption is encouraged by agile method's promising flexibility in terms of ability to accept and adapt to changes throughout a project cycle (Karhunen, 2009), coupled with its behavior as a time-boxed development cycle which guarantees workable software within short period of time compared to the fixed, long-period traditional methodology. These facts have influenced this research to choose agile methodology as domain of interest to work on, as agile popularity should be captured in a proven record for industrial guidance about opting agile methodology as software development cycle. Following, agile software development methodology is described briefly in this section of this thesis.

Universiti Malaya

Agile in a software development environment refers to a type of process or methodology used as a guideline in order to develop a software product, which is also treated like a software development lifecycle. In the early stages of agile discovery, this method has been used in many projects but has never been officially standardized as a recognized software development methodology to be practiced as a guideline in software industry. Thereupon, in 2001, a group of software engineering experts whom have worked and experiences with agile method has gathered and discussed to agree to officially recognize agile methodology in software development. They have come up with an Agile Manifesto which has define a set of values for agile methodology (Fowler & Highsmith, 2001). Along with the defined Agile Manifesto, they have developed 12 principles to further explain the values to be used as guidance in software development projects. There are many studies such as (Cho, 2008), (Hu, Yuan, & Zhang, 2009), (Karhunen, 2009) and (Karlsson & Martensson, 2009) which has deliberately explained and elaborate these 12 agile principles for better understanding. Figure 2.1 shows Agile Manifesto and the principles defined by the 17 software engineers during the meeting.

Universiti Malaya



Manifesto for Agile Software Development

We are uncovering better ways of developing software by doing it and helping others do it.
Through this work we have come to value:

Individuals and interactions	over processes and tools
Working software	over comprehensive documentation
Customer collaboration	over contract negotiation
Responding to change	over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Kent Beck	James Grenning	Robert C. Martin
Mike Beedle	Jim Highsmith	Steve Mellor
Arie van Bennekum	Andrew Hunt	Ken Schwaber
Alistair Cockburn	Ron Jeffries	Jeff Sutherland
Ward Cunningham	Jon Kern	Dave Thomas
Martin Fowler	Brian Marick	

© 2001, the above authors this declaration may be freely copied in any form, but only in its entirety through this notice.

Figure 2.1: Agile Manifesto (Fowler & Highsmith, 2001)

Tasks in traditional methodology is coordinated and assigned by Project Manager or Team Leader, unlike in agile methodology where tasks are coordinated by a self-managing team, which means team members decide and discuss tasks amongst themselves (Moe, Dingsøyr, & Dybå, 2010). This is aligned with agile characteristics which concentrate on; 1) Individual and interaction, 2) Working software, 3) Customer collaboration, 4) Respond to change rather than following a plan (Cho, 2008).

Traditional methodology assumes a Project Manager is able to plot a project schedule complete with predictable disaster and then assumes project will work as planned. But in reality, development project is full of surprises, unpredictable matters

Universiti Malaya

and changes, which fortunately could be handled by agile methodology as it is able to deal with the challenge of handling unpredictable project obstacles, such as unexpected change request from time to time (Moe & Dingsøyr, 2008). This is because agile attribute allows numerous unlimited changes throughout iterations in order to support product evolvement. Furthermore, agile methodology promotes less documentation while enhancing knowledge and skills sharing amongst team members. This is to ensure every team members have equal knowledge on the systems so that if a person of the team leaves, there is still a lot of shared knowledge that has gotten around among other team members (Cho, 2008). And as a final point; what makes agile an attractive methodology to be adopted, is the basic agile practices such as pair programming, continuous integration, short release and simple design, making the development process and procedure look much more promising (Harichandan, Panda, & Acharya, 2014). There are many empirical studies that have been made on agile approach and its benefits which (Eloranta et al., 2013) has summed up as following; 1) Better control on managing changing priorities, 2) Improved visibility and team morale, 3) Quicker time to market, and 4) Increased productivity.

There are many frameworks for project management process available to support agile software development; amongst them are Scrum, Extreme Programming (XP), Lean, Crystal, Feature Driven Development (FDD), Agile Unified Process and Dynamic Systems Development Method. Details of the comparison of these agile methodologies are as shown in Table 2.1.

Universiti Malaya

Table 2.1: Comparison of Agile methodologies (Coffin & Lane, 2006)

Category	Methodology	Strength	Weakness
Agile	• Popular choice as software development approach, • Identified agile characteristics, one of them is to emphasize communication between team members and customer, • Identified popular agile methods which is:		
	Scrum	• The only methodology compared here that has certification • Allows improvements and modification in the framework	• Only provide project management process, which acts like an empty bucket that requires combination or implementation
	Extreme Programming (XP)	• Supports pair programming which is a very strong technical practices • Allow constant refactoring of product developed as it is release-based approach	• Not a structured process phases but more to coding and releases where product is improved based on a series of releases.
	Lean	• Provide strong Return on Investment element for project success	• An element called Theory of Constraints can be complex to adopt
	Feature Driven Development	• Design by feature and build by feature	• Activities and process in the iterations are not well defined
	Agile Unified Process	• Provide many techniques and disciplines to choose	• Documentation is much more formal and heavier
	Crystal	• Provide many methodologies (a list of prescribed flavor) designed to scale by project size and criticality • The only agile methodology that could support life critical project	• Adjustment is required according to project size and criticality to follow the prescribed flavor

Universiti Malaya

Category	Methodology	Strength	Weakness
	Dynamic Systems Development Method	• Business value is the highest priority for deliverable	• Heavyweight project process compared from the list

A study by (Itkonen et al., 2005) has stated that it is not clear how the testing activities are going to fit into XP, TDD, FDD, etc. whilst on the other hands, Risk Poker has been suggested by (Van de Laar, 2012) to be able to fit inside Scrum as a risk-based testing. Thus, this research chooses to innovate the suggested solution for Scrum framework. The following section describes more about Scrum method within the scope of this research.

2.2 Scrum

The Scrum process skeleton is formalized and presented by Ken Schwaber at OOPSLA (Object Oriented Programming, Systems, Languages and Applications) in 1995. Scrum provides a simple iterative and incremental framework for project management (Hossain, Babar, & Paik, 2009) where Scrum project planning promotes product backlog stacks as paper-less documentations and burn-down charts as lightweight techniques compared with too many formal documentation and Gantt charts in traditional software development (Sutherland, 2001). In addition to that, Scrum is able to fulfill agile characteristics in order to produce software early and continuously while still maintain high degree of flexibility for project success. Scrum process is an adaptive cycle which regularly review activities to see what is occurring and take them into account to produce predictable outcome (Caballero, Calvo-Manzano, & San Feliu, 2011). Since Scrum team is self-managed, team members have the decision-making authority that comes into play when problems occurred, in which case they can be solved without escalating or needing the superior personnel for approval in order to

Universiti Malaya

obtain quick solution (Moe et al., 2010). These features have made Scrum the most widely used agile framework, where almost half of existing software industry with agile projects choose to use Scrum by itself or a hybrid of Scrum and XP (Löffler, Güldali, & Geisen, 2010).

Many studies have reported successful stories of Scrum adoption and their benefits such as;

1) Team is self-responsible in planning smaller tasks for themselves which will lead to a correct estimation (Haugen, 2006; Molokken-Ostvold & Haugen, 2007; Moløkken-Østvold et al., 2008),
2) Team members are forced to be exclusively committed to task as planned during the sprint planning because of the burn-down chart (Cho, 2008; Deemer, Benefield, Larman, & Vodde, 2010),
3) Team members are rewarded with high team spirit, enjoyable feeling and satisfaction when they are able to deliver end product at the end of each sprint (Caballero et al., 2011),
4) Daily meetings have shown that everyone has better general view of work progress and able to come up with solution together (J. Li, Moe, & Dybå, 2010),
5) Customer needs are addressed in every sprint, compared to traditional approach where customer needs are only addressed at the end of the project in user acceptance (Cho, 2008),
6) Better process control and quality because of controllable-sized tasks (Schwaber, 1997).

Additionally, (Selvi & Majumdar, 2013) have come up with top 10 reasons for adopting Scrum in agile project as shown in Table 2.2.

Universiti Malaya

Table 2.2: Scrum Benefits (Selvi & Majumdar, 2013)

Top 10 Reason to use Scrum
Scrum enables rapid reaction to changing customer requirements
Scrum teams possess all the required skills to get the job done
Scrum teams incur less technical debt
Scrum improves communication
Scrum leads to better client relationships
Scrum improves personnel satisfaction and commitment
Scrum reduces time taken to get product to market
Scrum produces higher quality product
Scrum succeeds by giving the customer what they need
Scrum increases productivity and lowers costs

The above-mentioned studies have influenced this research to focus on Scrum process improvement for agile project. Even though Scrum has been popular in agile project, (Cho, 2008) has reported some issues upon adopting Scrum in real practice such as;

1) No proper documentation traceability of tasks management when the system has become too big with too many releases. Even though Scrum promotes knowledge sharing amongst team members so that everyone is equal in knowledge, it is not feasible if the system is too large with distributed agile team,
2) Scrum promotes more communication with customer, but often customer is unable to communicate as required to obtain feedback of what they want for future system,
3) It's hard to gather everyone at the right time for daily review meeting,

Universiti Malaya

4) Potential to adopt wrong practice from the actual recommended Scrum practice, such as unintentionally keeping the waterfall process in some of the product development.

Now that the environment of Scrum framework has been described and related issues are highlighted as above, in the following section this research describes more on team members' role in a Scrum team. This is essential as Scrum team plays important role to make sure Scrum project is adopted with correct practice.

2.2.1 The Three Roles in Scrum

There are three important roles in a Scrum team, which are defined as; Product Owner, Scrum Master and The Team. Brief descriptions for each roles and responsibilities in Scrum are as follows.

Product Owner is the person who is responsible to make sure customer gets what they want. Product Owner is the main person who gathers and interacts with customer to collect requirements or features from customer before starting with a Scrum project. The collected requirements or features are written as a simple user stories which is understandable to customer as well, and it is kept in Product Backlog for reference. Product Owner is the only person who is responsible to prioritize the product backlog (Schwaber & Beedle, 2002) before Sprint Planning Meeting, where items that give the most value to the customer is placed at the top, thus granting Product Owner the authority to make decision on what item is important and what is not, based on customer's interest.

The next important role is Scrum Master; which is the person who is responsible in practicing the correct Scrum process throughout the project. Scrum Master has to make sure everyone in the team understands all the Scrum process so that they are able to

Universiti Malaya

follow the rules and practices. Scrum Master is also responsible in addressing and guiding team members to solve problem to prevent delays and bottleneck.

Finally, The Team consists of developers, testers, customer representative and other relevant personnel who make sure the development of product functionality is a success (Lewis & Neher, 2007). They are responsible for developing the product in a self-managed environment which means all team members must work together to produce workable product at the end of the sprint by sharing the knowledge, skills and concerns of the tasks assigned and cooperate accordingly as they are equally responsible for the end product (Moe et al., 2010). This characteristic of the team in Scrum team is proven to be able to improve productivity and able to integrate the product more efficiently (Moore, Reff, Graham, & Hackerson, 2007).

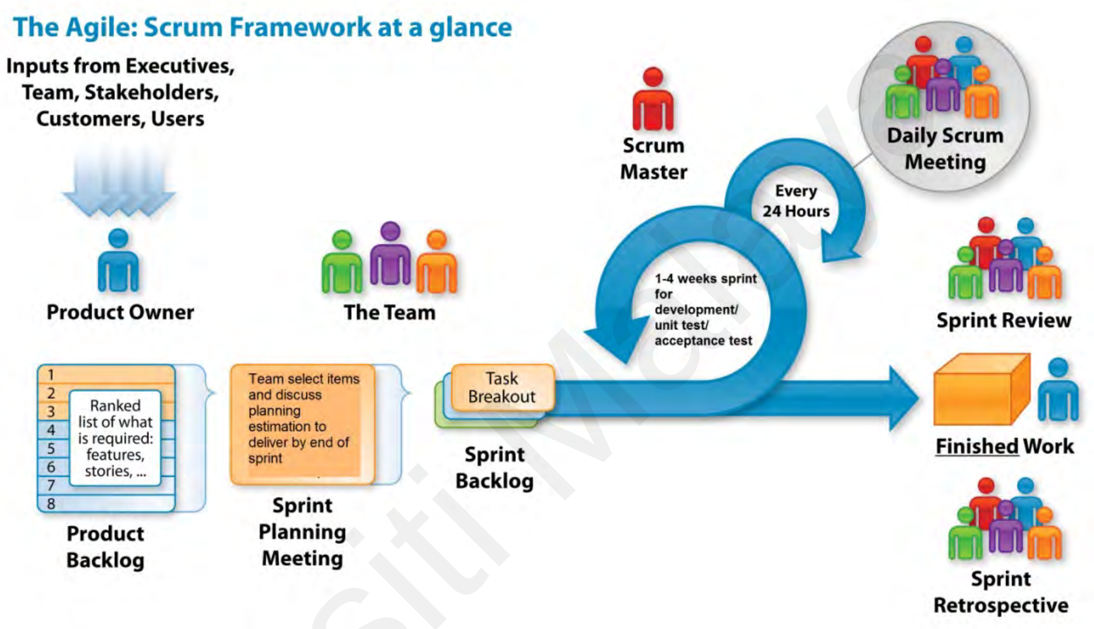
In the following section, this research explored on the process and work flows that make a Scrum framework works as project management process. Understanding these processes and activities helped this research to determine where to integrate the proposed method inside a Scrum framework.

2.2.2 The Scrum Work Flow

Scrum is a repetitive process of planning phase, development phase and review/closure phase for a software development project. At the high-level process overview, each iteration will start with planning, followed by development phase in a short period of time-boxed iteration called Sprint which last for one to four weeks and ends with reviewing/closure phase at the end of sprint. If there is room for the end product to evolve or if there are any changes uncovered during reviewing session, the changes and improvement will be brought to the planning phase for the next sprint. As shown in Figure 2.2, there are processes of product backlog collection, sprint planning meeting, sprint backlog creation also known as tasks prioritization, cycles of sprints

Universiti Malaya

where product is developed and tested, daily scrum meeting for progress update, and at the end of the sprint there are retrospectives in which the finished product is presented to the customer for feedback and discussion on issues is conducted.



The Agile: Scrum Framework at a glance
Inputs from Executives, Team, Stakeholders, Customers, Users
Product Owner
The Team
Scrum Master
Every 24 Hours
Daily Scrum Meeting
1-4 weeks sprint for development/ unit test/ acceptance test
Sprint Review
Ranked list of what is required: features, stories, ...
Product Backlog
Team select items and discuss planning estimation to deliver by end of sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Finished Work
Sprint Retrospective

Figure 2.2: Overall Scrum Process. Retold from (Hartman & Lawrence, 2013)

The sprint starts with Product Owner meeting and collecting requirements or what end user imagines they want for the end product. Requirements are collected as user story where end user or stakeholders list out all the things they want or think that a system should do in order to cater to their needs. User stories act as requirements and features documentation in Scrum (Karhunen, 2009), therefore Product Owner will clarify any issues of the listed user stories with stakeholders to grasp all the functionality needed and then prioritize the user stories in a stack of product backlog with those that value the most to the end user placed at the top. Product Owner is also responsible for determining the business value of the project that contributes to profit and loss to be prioritized in product backlog.

Universiti Malaya

Product backlog is a prioritized list of user stories which is rated by Product Owner according to business value for the proposed release (Schwaber & Sutherland, 2007). Items in product backlog is treated as the to-do list for the team to be executed in sprint, therefore they are often updated and available for the entire project duration. This backlog will continue to be refined and expanded by Product Owner to reflect changes and feedback by customer throughout the project lifetime. In addition to that, Product Owner is responsible to assign business value estimate or risk of impact value to each of the user story. Items ranked at the top of the product backlog list are rated as the most important to Product Owner, customer and end-user.

Once product backlog is ready and available to be explored to develop product functionality, a planning meeting will be held as a start. This meeting is also called Sprint Planning Meeting. Objective of the meeting is to discuss the selected product backlog item to decide, estimate and assign tasks to team members. In order to do so, team members select some user stories prioritized by Product Owner in product backlog and start discussing the user stories in depth with Product Owner, Scrum Master and The Team. Following that, team members will break the selected product backlog into a smaller workable task to be committed throughout the sprint and team members start to discuss the tasks in details. It is important for the team to fully understand the selected items before starting to work on it. During the discussion, Product Owner will explain the user stories in perspective of business value and what end user wants to the team and the team members will exchange knowledge and opinion on the feature to be developed in terms of development and testing wise. Once everyone is satisfied with the discussed items, it will be assigned to the respective person with estimated staffing effort such as tasks break-out, work hours, etc (Schwaber, 1997). At the end of Sprint Planning Meeting, the list of product backlog items which have been divided into smaller tasks

Universiti Malaya

with assignment estimation will be sorted in Sprint Backlog for team’s reference. A Sprint Backlog in Scrum stores a list of tasks broken-down from the user stories selected by the team in Sprint Planning Meeting which will be worked-on in the sprint.

Next, the team starts product development in order to produce the functionality planned within the estimated time-boxed sprint (Eloranta et al., 2013). Some people may call this sprint as development sprint (Eloranta et al., 2013). Objective of this sprint is to deliver a workable functionality at the end of sprint (Moore et al., 2007). During the development, the team will meet every day with Scrum Master to update status and problems faced in development. This meeting is called Daily Scrum Meeting and will last for 15 minutes only. No other issues should be discussed apart from what the team is doing and what obstacles are hindering the team members. This is to ensure the meeting is not a waste of time and interrupt team’s schedule of that day.

At the end of the sprint, a meeting called Sprint Review meeting is held to decide on the status of the end product. The Team will present the constructed product to Product Owner and stakeholders in order to gain feedback whether the product is declared as done or not done, or whether there is any improvement or changes needed for the functionality. If there is any change request made by stakeholders, the changes will be added into product backlog for the upcoming sprint. This activity gives the stakeholders the ability to keep track of the requirement changes and to see the product evolve from one sprint to another sprint. This will give them higher confidence in the team’s ability to deliver the requested functionality in the desired time-frame (Schatz & Abdelshafi, 2005).

Universiti Malaya

Lastly before starting and planning for a new sprint, a Sprint Retrospective meeting is held by Scrum Master with The Team. Objective of this meeting is to revise and evaluate the previous sprint. Scrum Master will record if there is any issue from previous sprint that should be addressed in the next sprint and if there is any improvement needed for the upcoming sprint. This is to gain better estimation during sprint planning and to obtain improved effectiveness and efficiency throughout the Scrum project.

The Scrum flow section has described all processes taken place in a Scrum team. These processes or framework of a Scrum methodology is like a bucket with rooms that need to be filled in with some other important process modified or implementation of another strategy to suit the agile development environment better (Khalane & Tanner, 2013). As an example, testing process could be integrated into this Scrum bucket to enhance quality assurance in the product development of agile methodology. Therefore, this research is going to identify software test strategy to go along with Scrum process to integrate testing process in a Scrum development environment. Following section touched in brief the discovery of software testing in agile methodology in order to understand more on software test strategy characteristics.

2.3 Software Testing

Software testing in a software development lifecycle is meant to expose defects of product development and coding errors, to measure product's reliability and dependability while convincing customers that the performance of the product is acceptable and increase customer's confidence that the product is able to perform correctly. On the whole, software testing helps to achieve the final goal of a software development process which is to produce high-quality software in an attempt to satisfy the requirements and meet the user's needs. Studies also reveal that the quality of

Universiti Malaya

testing processes translates into the level of quality and software development effort, where many agile's practices success rate depends on effective software testing process (Hellmann, Sharma, Ferreira, & Maurer, 2012; Khalane & Tanner, 2013; Winter, Rönkkö, Ahlberg, & Hotchkiss, 2011).

Software engineering has recognized that software testing has become an essential activity in the software development lifecycle in order to determine and improve software quality over time when a nonprofit association has established a recognized international organization of software testing called International Software Testing Qualification Board (ISTQB) offering certificates internationally. The certificate is to recognize and qualify software tester in order to offer assurance in quality control of the software tested to regulate the standard of software testing quality. ISTQB (Thomas Müller, 2011) has defined some characteristics of software testing to be adhered; 1) To ensure the program is meeting the business and technical requirements agreed for the program's design and development architecture, and 2) To deliver a program that will work as expected.

Unfortunately, software testing is often mistakenly treated as a single activity to be executed after coding to uncover defects randomly while still expecting that it could verify the software to the characteristics defined by ISTQB. The truth is that software testing is a process that runs continuously parallel with other processes in software lifecycle development. It has planning phase, analysis and design phase, execution phase, exit criteria evaluation phase and ended with reporting. Planning phase is to determine scope, analyze and review test item, assign resources and effort required, and identify test approach to design test, and execute test is defined based on approaches suitable with the project nature. The most common adopted approaches are risk-based strategy, requirement-based strategy, model-based strategy and experience-based

Universiti Malaya

strategy. Table 2.3 shows the strengths and weaknesses comparisons of these software test strategies against agile characteristics. The analyses of these comparisons has shown that risk-based strategy suit agile characteristic where risks are analyzed and prioritized to discover problematic area early. Risk-based strategy involves test planning, estimating and prioritizing tests based on the risk analysis performed using project documents and stakeholder inputs. On the other hand, requirement-based strategy involves test planning, estimating and design tests based on the analysis performed using the requirement specification documents. And lastly, model-based strategy involves building mathematical models of the critical system behaviors and then executes testing to conform the system is able to be working as predicted by the model. While in analysis and design phase, test item is designed and reviewed to be ready for the next phase which is test execution. The test item is then tested in execution phase according to the test technique or test strategy defined in planning phase. When testing of the test item is finished, exit criteria is evaluated based on the test coverage defined during planning phase and the testing activity is concluded with test report that contains evaluation on how testing activity performed and lessons learned for future release.

Universiti Malaya

Table 2.3: Comparisons of software testing strategies (Thomas Müller, 2011)

Category	Software Test Strategy	Strength	Weakness
Software Testing	• Identified and understand software testing process • Identified software testing approach for software development project to learn which approach is suitable with agile:		
	Risk-based	• Tests are focused on most critical areas with risk analysis • Problem areas are discovered early, leads to better strategies and tests designed as preventive method	• Risks might be assessed as too low or too high • Risk assessment can be too subjective
	Model-based	• Automated testing of test generation and test result which is based on system model • Useful for structured analysis • Easy to understand between different teams in an organization	• The model designers should be an expert of the application area • Might encounter problem in determining whether test failure is caused by code or test script • Might not be suitable for testing all application
	Requirement-based	• Precise and details testing when it has clear, complete and correct requirements • provides well-structured testing with improved traceability and visibility	• There will be limitations and error when the requirement is not defined correctly • requires complete and correct requirement
	Experience-based	• Works well when there is no adequate requirement provided • Particularly useful for low-risk project with time constraint	• Intensity and completeness of this test design cannot be measured • Not a systematic approach with no record traceability

Universiti Malaya

The flow of a traditional software testing, which can be directly implemented in a traditional software methodology, starts from the beginning of a software development project, in the design and requirement phase, where static testing is carried out to review or inspect the design and requirement of the program to prevent early mishaps of program design and data flow. Following that, dynamic testing such as unit testing, integration testing and system testing is executed on the coded program to uncover defect using a set of techniques and test cases. In order to carry out dynamic testing, test leaders will define and plan the software test design technique and test coverage for tester. Once testing activity is executed as planned, defects detected will be fixed by developer and regression testing is carried out to unmask any hidden defect. Lastly the program under test is evaluated to make sure the tested program has met the completion criteria to decide whether the program has finished testing and ready to be delivered to customer.

Upon finishing testing, it is important to make sure the program was tested adequately before it is declared as "done" and handed-over to customer, as (Woodward & Hennell, 2005) has stated the impact of an inadequate infrastructure for software testing could affect business loss. Hence, it is suggested to couple testing with coverage criteria to cover the entire set of testing parameter be it conformance of specification or the accuracy of coding structure (Julius, Fainekos, Anand, Lee, & Pappas, 2007). Due to the important of testing process in a software development environment, much research is needed toward developing new, improved test methods (Briand & Pfahl, 1999) consistent with the adopted software development lifecycle.

Universiti Malaya

2.3.1 Software Testing in Scrum

Most of the adopted industrial software testing techniques are the existing techniques established for traditional software development lifecycle such as waterfall methodology (Itkonen et al., 2005). However, software testing in agile project environment requires testers to learn about the program under test thoroughly and always remember to concern themselves with customer's information as guidance for testing activities (Collins, Dias-Neto, & de Lucena, 2012). This research finds it important to invest in Scrum focusing on software testing as Scrum let the team members define their own quality strategy which is observed as more effective as they are self-managing and makes decision by themselves upon what they are working on rather than relying on management team (Caballero et al., 2011). Moreover, without fully changing basic tasks such as coding and testing, Scrum is aiming to deliver as much quality software as possible within the short time-boxed sprints which aligns with software testing goals (Lewis & Neher, 2007). Apart from that, during user stories discussion in planning meeting, testers could help the team to uncover complex business logic by identifying complex and negative test case scenarios and together discover which items of the program could be affected and significantly reduce potential defect and thus add more information as guidance to estimate the stories (Kayes, Sarker, & Chakareski, 2013).

However, despite the fact that software testing is an important and costly activity in software development lifecycle, and inadequate testing usually leads to major risks (Garousi & Zhi, 2013), Scrum process and most agile methods do not describe specific software test strategy to be taken into consideration (Itkonen et al., 2005). This is important because in agile, the team is not just responsible to identify failures, but they are also responsible to prevent it from happening at early stage. Therefore, agile testing

Universiti Malaya

is a challenge for traditional testers that used to be involved in testing activity at the end of a waterfall project (Collins et al., 2012).

Apart from the importance of early testing, (J. Li et al., 2010) also points out that knowledge sharing through group discussion helps in improving test efficiency, hence the importance of team concept in agile practices. In the team, it is not just testers who care about quality. The whole team is responsible to understand that quality is a part of the product development and how they fit into the process. Discussion amongst the team members in agile practices guides the team into understanding what quality attributes should be considered and together come up with the definition of “done” as the completion criteria (Talby, Keren, Hazzan, & Dubinsky, 2006). This means that testers in agile projects need to work closely and coordinate between business, management and developers.

In view of that, we can see that tester’s responsibility in Scrum does not focus on testing activity solely, but tester is also responsible to explain to developers and customer’s representative of the testing issues, review unit tests strategy, review stories and making them testable together with team members so they can understand the stories from inside out and together decide on the test strategy. In addition to that, the advantages of including professional testers in any agile development team are as follows; 1) Increase productivity, 2) Increase of cross competence within the team, 3) Promotes wider knowledge sharing which improves group dynamics, and 4) Could improve effort estimation and test strategy estimation (Karlsson & Martensson, 2009).

Universiti Malaya

2.3.2 Testing Issues in Scrum

Previous studies on software testing for agile projects specifically Scrum are reviewed to gain insight on the current situation of software testing issues in agile projects. Generally, agile methodology suggests no specific roles amongst team members. Everyone is responsible equally on the software development throughout iterations. Thus, quality assurance is every team member's responsibility. This leads to the raised issue of whether testers are needed in agile development at all (Kettunen, Kasurinen, Taipale, & Smolander, 2010). However, when dedicated tester is not assigned to an agile team, wherein developers test their own code, tester's attitude is hard to achieve within these developers and could lead to bias in producing quality software, hence the need of software tester professionals for software testing in agile (Itkonen et al., 2005).

Next in-line, many studies have raised concern that Scrum leaves too many aspects of quality management open. Common issue is how to fit software testing in the iterations (Caballero et al., 2011) as there is limited amount of studies in relation of agile methods and testing. Many structured text book software testing processes are for traditional software development environment. Testing practices used in these plan-driven methods may not be compatible with agile processes that do not have a structured defined requirement at the beginning of the development (Itkonen et al., 2005).

Apart from that, not much research in software testing for agile project focuses on test planning and the control over test coverage in order to provide effective test adequacy criteria for the short time-boxed iteration as scrum development process (Khalane & Tanner, 2013). Study by (Kettunen et al., 2010; Stolberg, 2009) has reported that it was difficult to test in parallel with development when applying the

Universiti Malaya

traditional testing approaches in agile environment and it has also cut short on time which resulted in unsatisfied testing coverage at the end of iterations. A study by (Petersen & Wohlin, 2010) also reported issues of test coverage and staffing effort when migrating from traditional test model to agile model. This issue has been identified throughout literature and continuous approach is proposed such as Extreme Programming (XP) which emphasizes the importance of building quality into the system with early testing, pair programming throughout the development lifecycle. However, it is not clear how the testing activities are related and synchronized with the other development tasks.

Lastly, as Scrum does not define proper test planning and how dedicated testers fit into Scrum process, this has left test engineers not knowing what to test, how much testing is needed, or what is the required output from the testing activity (Karhunen, 2009). Thus, as reported by (Khalane & Tanner, 2013), because agile and Scrum do not provide concrete guidance on testing strategies and how to fit them into the process, it has left the team to become innovative by adopting practices from other methodologies such as traditional software testing strategy for waterfall methodology and carefully redesigning the current scrum process structure to accommodate the adopted practices, hence the need to define what type of testing technique or metrics in order to decide what should be tested (Garousi & Zhi, 2013).

The listed issues were collected from previous studies where researches of the related studies have encountered those issues and highlighted them in their report as summarized in Table 2.4.

Universiti Malaya

Table 2.4: Summary of software testing issues in Scrum from previous studies.

Issues in Scrum	Research settings	Result
The need of professional Tester roles is arguable in agile environment	(Itkonen et al., 2005): A study on how agile method affects test processes on a number of organizations compared to the test processes in a plan-driven project.	A list of benefits observed on test processes in agile project which will be beneficial for software organization to address the issues beforehand in test process.
Difficult to test in parallel with development when applying traditional testing approaches. The short time-boxed iteration caused the unsatisfied testing coverage.		
Scrum leaves too many aspects of quality management open, thus the question of how to fit software testing in iterations	(Garousi & Zhi, 2013): A survey of software testing practices among practitioners in Canada to get the important and interesting findings about software testing practices.	Reveals the latest trends in software testing industry, identifying the areas of strength and weakness.
Issues with test coverage and staffing effort when migrating from traditional test model to agile	(Petersen & Wohlin, 2010): Investigate the effect of moving a test process from one model to the other. How the perception of bottlenecks, unnecessary work, and rework changes when migrating from a plan-driven to an incremental software development approach with agile practices.	Reveals the issues during migrating the test process, after effect of the migration process which related to testing lead-time and test coverage. Improvements were also identified where many issues commonly raised in plan-driven approach were not raised anymore in agile approach
The team needs to become innovative by adopting practices from other methodologies since agile and scrum do not provide concrete guidance on testing strategies	(Khalane & Tanner, 2013): Identify and present the concerns of stakeholders (in meeting user expectation) for Software Quality Assurance in Scrum. This study demonstrates the incompleteness of agile methods with particular attention to the lack of concrete guidance in Scrum, thus the need of the team to be innovative to adopt existing method into agile and Scrum.	This study draws on method tailoring literature to argue for customization of Scrum and concludes that Scrum needs to be viewed as a framework of 'empty buckets' which need to be filled with situation specific test practices and processes in order to meet user expectation.

Universiti Malaya

All of these issues have shown that Scrum methodology should be treated as an empty bucket framework which needs to adopt a reliable software testing process to improve software quality and meeting user expectation as suggested by (Khalane & Tanner, 2013). Thus, as proposed by (Van de Laar, 2012) to implement Risk Poker as a risk-based testing for Scrum, this research is focusing on Scrum as software development methodology. Therefore, discussion of appropriate software testing strategies is discussed within Scrum scope.

A qualitative study of survey conducted by (Kasurinen, Taipale, & Smolander, 2010), on the preferred testing process for software development has guided this research to narrow down which type of software test strategy suitable for agile practitioners. It is learnt that most of agile-based industry choose risk-based testing process due to good feedback and customer participations. The risk-based technique enables the team to focus on the most critical parts first which also improve the position of testing to start early and provide tester with better insight of the software since the team discussed the testing issues together at the beginning of the development. Thus, in the following section, this research explores in details what motivates this research to use risk-based approach as software test strategy.

2.4 Risk-based Strategy in Scrum

Literature reveals that, most of the time agile teams have to modify existing process, strategies or techniques to fit in agile environment (Stolberg, 2009). Moreover, not much articles and researches discussed which software test strategies works best in Scrum but some of the existing research did come out with enough facts to lead this research to proceed with risk-based testing as the best software test strategy for an agile project such as Scrum.

Universiti Malaya

Following is the justifications that lead this research to choose risk-based testing as software test strategy in agile project;

1) The agile environment itself such as budget constraint and time-boxed iterations has pushed team member to prioritize level of testing needed for testing activity in which, prioritization of testing level is actually a risk-based testing strategy (Stallbaum, Metzger, & Pohl, 2008),
2) A research by (Kasurinen et al., 2010) which has surveyed 31 software industry organizations, and interviewed 36 software professionals from 12 focus organizations in determining the preferred test selection strategy whether it is risk-based or design-based selection. The development approach for this focus group varies from plan-driven methodology to agile methodology and mixes of these two methodology at which has produced result that most of agile methodology practices adopted risk-based testing selection as testing strategy.
3) Agile process implicitly apply risk-driven strategy when sprint are defined and task are assigned (Nyfjord, 2008) which have the commonality with risk-based testing strategy method,
4) Even though agile process itself is a risk-driven process, it does not explicitly include risk management phases as in how to identify, analyze and mitigate risk (Paulk, 2002), thus a risk-based testing strategy could support risk management efficiently during project execution.

Therefore, based on the evidence listed above, risk-based testing as a software test strategy is best adopted in agile project since it is the nature of testing activity that there is always never enough time to test everything, especially in a time-boxed iteration like agile projects. Moreover, it is common that testing team often puzzled on how to assess user stories' business value, how to analyze technology risks and how to achieve

Universiti Malaya

consensus on certain decision on their own (M. Li, Huang, Shu, & Li, 2006). Following section describes risk management in software development project and risk-based testing as software test strategy to help understanding how risk analysis is conducted throughout a software project.

2.4.1 Risk Management and Risk-based Testing

In a software development project environment, risks are addressed in risk management discussion. Risk management is usually linked with project management planning which is carried out by project managers and stakeholders for a software development project. It is always emphasized on how important it is to identify risks in software project management and act towards it in order to prevent disasters, cancellation and frustration (Bannerman, 2008). The effect of project failure which is caused by unidentified and unmanageable risks can be controlled or minimized by having risk management composed of a collection of risk control methods. In general, risk management involves risk identification, risk analysis, risk prioritization and risk control (Hartmann, Fontoura, & Price, 2005) which is illustrated in Figure 2.3. In planning phase, any possible risks to the project are identified and this action is called risk identification. Following, an estimation of the probability of the risks happening and the consequence should the risk happens is analyzed (Hall, 1998). Once the risk analysis is complete, the risks are prioritized according to their importance. The risk prioritization allows project manager and team to execute actions described previously from the highest risk items first (B. W. Boehm, 1991). Afterward risk control is discussed; as an example, what are the strategies to deal with the risks and the risks resolution plan should any of the risks predicted occurred during project execution.

Universiti Malaya

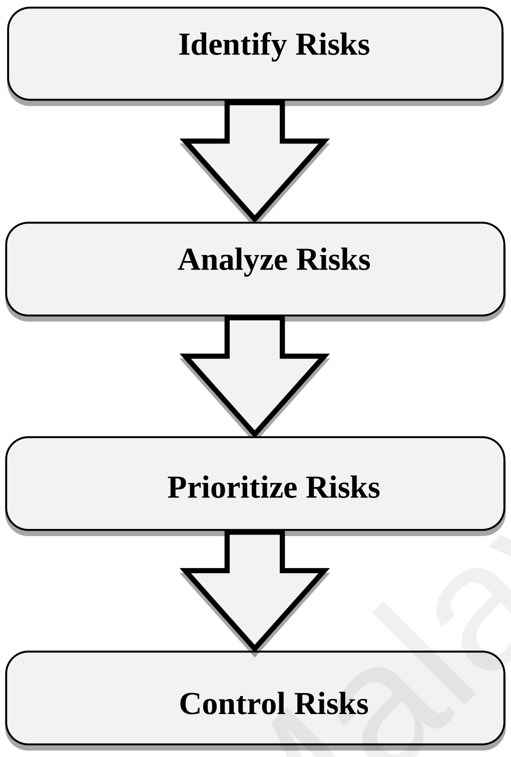


Figure 2.3: Risk Management Process

Similar to a software project risk management, ISTQB (Thomas Müller, 2011) has summed up that risk-based testing would help testing team to perform risk management, identify hazards that would lead to potential project risk, describe the risk that would threat project's objective, distinguish between project risks and product risks, use risk management element for test planning and define how testing would be carried out. From testing point of view, risk-based testing strategy guides how much testing activities and effort to spend based on the risks assessment where; high risk items will need serious testing compared to the low risk items. In short, the goal of a risk-based testing is to uncover the costliest and most important defects or faults as early as possible so that when a test is required to stop, risk-based testing has ensure that testers have spent the budget in a well-organized approach (Stallbaum et al., 2008).

Universiti Malaya

Risk management in this testing strategy requires testing team to continuously assess what might possibly go wrong that would lead to risks and identify which are the important risks to deal with, followed by the strategies to deal with those risks. Hence, from all these risk-based testing characteristic, a study (Bannerman, 2008) has concluded that risk analysis would improve estimation and reduce duplication of effort for the team.

Thus, in relation to the details explained about risk management and risk-based testing above, this research agrees with a study by (Nelson, Taran, & de Lascurain Hinojosa, 2008) which shows that it is effective and important to manage risks explicitly in an agile structure development so that everyone in the team is aware and understands every risk identified, understand and contributes on the risks mitigation strategy and able to execute it as planned. They also reported in their research that, many agile projects implicitly managed risks which has left team members hanging without knowledge and awareness of the possible risks, so when any of the risks happen, team members are unable to control the risk which leads to project failure or increased project cost. Hence the need to propose a testing model with risk-based testing strategy that could overcome this issue. In accordance to identify a suitable risk-based strategy for the proposed testing model, following section explored on previous research about risk-based strategy techniques for comparisons.

Universiti Malaya

2.4.2 Comparison of Risk-based Techniques

In this section, literature on existing risk-based techniques is explored to identify suitable risk-based techniques to be adopted as risk-based testing strategy for agile. From literature, many researchers have focused on risk management at project level while only a few concentrated on defining risks for product level risk control manipulation as below;

1) A research by (Black, 2003) on how to and why use risk analysis for test planning and quality guidance has shed some light that it is also beneficial to manipulate risk assessment at product development level. Risk analysis proposed is to estimate cost of testing, however without knowing how much test is needed, cost estimation would not be accurate.
2) A research article by (Hartmann et al., 2005) has proposed to manipulate risks criteria to help choosing the right development methodologies for a software project. The designer will perform a risk analysis of the project and use the tool to decide on the development methodology pattern to tackle the risks identified appropriately. This research could be enhanced to support decision on testing methodologies as well in order to cover all processes in a software development project.
3) A study by (Boness, Finkelstein, & Harrison, 2008) suggested to assess risk and provide risk control at an early stage by using the requirement analysis on goal graphs and judgments supplied by stakeholders or experts. However, if the requirement analysis is not clear, there is a possibility that some risks are missed out and unable to be controlled when it happens unexpectedly especially in agile project.
4) A useful tool of a model-based technique for risk-based system testing called RiteDAP has been proposed by (Stallbaum et al., 2008). This tool accepts risk analysis of an activity diagram and then automatically generates and prioritizes test cases for testing but no test coverage of how much testing is needed is provided to measure test adequacy especially in a time-boxed iteration like agile project. Furthermore, a detailed activity diagram is required to make use of this tool.

Universiti Malaya

5) A technique that implement risk-based strategy in agile is proposed by (Van de Laar, 2012) where risk analysis is executed by team members to help them prioritize the user stories of the product to be developed. The risk analysis could be used to estimate how much testing is needed throughout the development stage.

Table 2.5 summarizes the above risk-based technique researches and the comparisons whether the technique is able to fit in agile environment or not.

Table 2.5: Comparisons of Risk-based techniques.

Risk-based technique	Domain Application	Agile characteristics
Cost of Exposure (Black, 2003)	**Testing cost estimation** Risk analysis to estimate cost of testing.	None;
Pattern-based Methodology Tailoring (Hartmann et al., 2005)	**Development methodology estimation** Risk analysis which provides development methodology based on pattern and risk criteria.	None;
Requirement analysis using Goal Graph (Boness et al., 2008)	**Development risks estimation** Risk analysis to assess risk in requirements analysis using goal graphs and judgments from stakeholders to estimate project risk instead of product development risk.	None
RiteDAP (Stallbaum et al., 2008)	**Software test strategy estimation** A model-based technique for risk-based system testing which automatically generates and prioritizes test cases based on the test models that have been provided with information about risks.	None; this tool requires a complete use-case or activity diagram from a complete defined user requirement which is not always available in agile projects environment. Furthermore, any changes on the use-case or diagram will cause difficulty to be implemented in the tool as add-on or changed activity diagram.

University Malaya

Risk-based technique	Domain Application	Agile characteristics
Risk Poker (van de Laar, 2012)	**Software test strategy estimation** Risk analysis amongst team members to assess risks no matter how simple user stories provided. The risk analysis provides risk prioritization for the user stories.	Yes; achieving group consensus for risk analysis.

2.4.3 Discussion

Risk-based testing has proven its practicality in managing and mitigating risks but (Nyfjord, 2008) has made a good point by concerning on how to merge the lightweight process of agile with the standard industrial process without damaging the agility characteristics. From this point of view, this research has narrowed down the focus to adopt existing technique which has been proved successful in agile project and has been used widely in industrial project specifically Scrum methodology. This has led to an effort estimation technique used in Scrum called Planning Poker which has been identified as popular to Scrum project for staffing effort estimation (Williams et al., 2009) and following with its popularity, a risk-based technique has been derived; which is called Risk Poker, as mentioned in the previous section. The existence of Risk Poker as a risk-based technique which provides lightweight risk analysis technique described by (Van de Laar, 2012) could fit agile project following Scrum perfectly. The following section discussed how Risk Poker technique could perform as a software test strategy for an agile project in detail.

Universiti Malaya

2.5 Risk Poker as a Risk-based Software Test Strategy

As mentioned in section 2.4.2, only Risk Poker technique as a risk-based technique is able to be implemented in Scrum process without the need to modify the technique to fit agile characteristics. Furthermore, review of previous studies has shed some beneficial points to adopt Risk Poker technique as software test strategy in Scrum. A review by (Kettunen et al., 2010) has stated that one of the main challenges faced in agile software testing is that development team is unable to communicate with each other and fully understand about software testing thus making it difficult to estimate the testing activities. To top it up, test strategy and test plan for the agile project is also a problem when development tasks are given higher priority compared to testing tasks. Hence, (Garousi & Zhi, 2013) has pointed out the need of risk and priority-based testing to overcome this issue and to shorten time in agile project. In the same study, Garousi has raised questions as whether tester knowledge alone is enough to be relied on upon creating test-cases and how to decide on what should be tested and what should not, which is also a concern stated by (Jogu & Reddy, 2013) that if the team choose to use the conventional well-established testing process, it should be changed and modified to suit agile environment.

Thus, in searching for the most economically efficient way of performing software testing, Risk Poker could overcome this issue with its team-based discussion for decision and risk analyses upon achieving equal understanding and knowledge sharing amongst team members. Furthermore, (Moløkken & Jørgensen, 2004) has raised concerns that despite having group discussion as a high value in agile environment, most of risk analyses are based on individual expert-judgment estimates rather than group decision. This leads to hidden related issues when knowledge sharing and discussion is not implemented. Thus, group discussion amongst people with different

Universiti Malaya

knowledge and backgrounds helps to identify more issues that might affect development, testing, implementation, etc. This group discussion characteristic is one of the main key-point in Risk Poker technique, which will also promote optimized knowledge and expertise sharing amongst team members which will improve analyses.

Risk Poker is held in a planning meeting to discuss risks associated to the requirement for the sprint and prioritize the risks for development and the prioritized risks defined the intensity of the development and testing required for the sprint. Risk Poker is also considered as risk assessment activity as team members analyze and identify risks related to the user stories and prioritize the development task and testing task from the highest risk component to the lowest. The Risk Poker technique identifies risks for the development of product and estimate the required testing effort for the product. It is important to cater to the high-risk items first because if time or budget constrains might be disturbed, the team is able to let the low-risk items get forwarded into the following iteration and prevent great failure to the program.

Risk Poker is structured where team members are participating in the discussion and everyone is sharing their opinion regardless of their position or working experience. This way, for the team member that does not have much to say due to limited expertise, will implicitly learn and grasp new knowledge from the matter's expert and allowed to ask question for better understanding. Such a structured knowledge transfer could ensure everyone will have equal knowledge of the subject matter to estimate risks effectively and prevent misunderstanding of the subject discussed. Output of the risk poker session is a list of risk assessment where related risks of the requirement for the iteration discussed are identified and prioritized according to its importance.

Universiti Malaya

With all that being said, here’s to more reason why adopting Risk Poker would benefit an agile development team;

1) In reality, even without a formal risk assessment strategy, agile processes are managing risks and attempting to mitigate some risks implicitly, but since they are not organized or structured, those risks might be left untreated (Nelson et al., 2008). Thus, a structured technique that supports agile environment is needed to properly address and control risks identified in agile development.
2) Agile process consists of self-managing team members, therefore, the team is expected to share the authority of making decisions rather than having one person responsible to make decision for the team or even accept individual decision regarding their work without having other team members involve in their work (Moe et al., 2010).
3) Apart from the self-managing team characteristics, the team is expected to be a cross-functional team where members have the necessary knowledge to deliver working program because without this ability, a typical problem in an agile development is that the product is not ready at the end of the sprint (Eloranta et al., 2013).
4) When the items are prioritized according to the risk, it could ensure that nearly 50% of the feature developed is sufficient to meet the goal, and consequently, project manager or product owner could opt to drop the remaining requirements if necessary (Schatz & Abdelshafi, 2005).
5) In scrum environment, it is observed that product owner might not have enough time or did not have required competence to sort and prioritize sprint backlog accordingly (Eloranta et al., 2013).

Universiti Malaya

Thus, adopting Risk Poker would ensure team members could manage this matter themselves. Although implementing Risk Poker technique as a risk-based testing promises beneficial result, it still has some room for improvements. Following items are open for discussion and improvement available for risk poker technique:

1) Risk Poker as a risk-based testing has been suggested to be used by (Van de Laar, 2012) but there is no empirical study available on Risk Poker technique in software engineering context.
2) As introduced by (Van de Laar, 2012), risk analyses from Risk Poker can provide testing information needed, but there is no guide on test coverage selection; an estimation of how much testing is required for each prioritized items will prevent waste of time, effort and limited resources on the unnecessary low risk areas of the code that may already be adequately tested and has less hidden defect of a program failure.
3) Risk Poker can be easily combined with Planning Poker as they complement each other, but there is no empirical study to prove it.
4) Risk Poker is a lightweight technique which is quick to understand and could get fast result as it uses traffic-light colored poker cards instead of the conventional quantitative approach using risk factors (Van de Laar, 2012). However, there is no empirical study to measure this lightweight claim.

Having said the benefits of Risk Poker technique and the lack of evidence on Risk Poker as a risk-based testing has motivated this research to adopt Risk Poker in Scrum methodology, implement and evaluate the effectiveness of this technique as evidence for academic and industrial reference. Moreover, combining Risk Poker with test coverage estimation would ensure adequate testing is given to the program developed and simultaneously improve test process and software development quality. Thus, next

Universiti Malaya

section discussed more on test coverage to be selected to estimate how much testing is needed for testing.

2.6 Software Test Coverage

Test coverage could be a good indicator to measure software quality by giving information of coverage adequacy for system under test, thus making it an important step in software testing process (Shahid, Ibrahim, & Selamat, 2011). When test coverage is defined correctly, it ensures that testing is executed effectively according to the coverage criteria without missing the important areas of the system under test. (Dang & Nahhal, 2009) also quoted that test coverage has become a way to relate how much tests needs to be carried out. Furthermore, inadequate testing has become a major problem and it is an area that is still given much focus amongst researchers to explore (Lawrence, Clarke, Burnett, & Rothermel, 2005).

Adequate test coverage is a testing execution which is considered as "good enough" when it meets the defined criterion. Generally, when a test suite is able to detect every defect and verify correct behavior of the program, it is considered "good enough" and effective in measuring software quality. Unfortunately, in reality it is impossible to detect all defects in a program and claim the program is defect-free. Therefore we need to define a test criterion which is a set of requirement to be fulfilled or achieved as an adequacy measurement which acts as a stopping rule to mark when it is enough to stop testing (Marré & Bertolino, 2003). Apart from become a stopping rule, test criterion could also be defined to determine the observations that should be made during the testing process (Zhu, Hall, & May, 1997).

Universiti Malaya

There are many test coverage techniques that have been developed such as;

1) Counting how much program blocks are covered in statements, branches, conditions and number of dead mutants in mutation testing for structural source code testing,
2) Data-flow transition coverage in state machines and path coverage to satisfy all program's behavior from entry node to exit node (Walkinshaw, Bogdanov, Derrick, & Paris, 2010).
3) Structural testing coverage measurement works well with incomplete requirements as in agile environment. This type of testing is also useful in exposing unwanted program code or functionality since such testing inspect program codes; looks for statements not executed by any test cases (Woodward & Hennell, 2005).

Table 2.6 shows detail comparison of test coverage techniques mentioned above.

Table 2.6: Comparison of Test Coverage techniques.

Category	Summary	Strength	Weakness
Test Coverage	• Identified how to measure test adequacy • Identified various approaches to measure test coverage		
	Code Coverage	• Able to measure code coverage in a unit test where each written code is tested	• Only measure coverage of what has been written
	Requirement Coverage	• Allow positive and negative test on the product functionality	• Require complete list of requirements to define test coverage
	Structural Coverage	• Each element of the software is exercised during testing	• Will be hard to define coverage structure for a gradually change product
	Architectural Coverage	• Actual control and data links are utilized during testing	• Needs to define detailed architectural design to measure coverage

Universiti Malaya

Much research focus on how to measure the degree of coverage achieved by a test set and how much more is needed instead of determining how much is enough to be declared as stopping rules. A research by (Gargantini & Riccobene, 2001) for ASM specifications provides a formula called test predicate is defined for coverage criterion to determine if a particular testing goal is reached when each of the test predicate is true. This formula is good to determine and measure which specification and testing goal is not covered yet and summarize test coverage percentage, however it is not able to serve as stopping rule to indicate when to stop testing.

On the other hand, (Marré & Bertolino, 2003) has introduced the concept of a spanning set of entities for coverage testing where the generated reduced sized test suite set could guarantee the coverage needed. They have provided a method to derive a spanning set that is parameterized in the inclusion relation between entities which is useful for reducing and estimating the number of test cases and for evaluating test-suite thoroughness more effectively.

Nevertheless, all these techniques are not lightweight technique to be applied in a real software project as it takes some time for the tester to learn how to formulate these techniques into test coverage criterion in order to determine test adequacy, especially in a short time-boxed iteration such as agile project. Therefore a comprehensive study of test coverage by (Zhu et al., 1997) particularly for code coverage in unit testing is used as guidance in this research in order to define test coverage criterion for unit testing as;

i. It could serve as a stopping rule criterion for a testing activity closure (Zhu et al., 1997),

Universiti Malaya

ii. Test-cases are designed based on the internal structure of the program, thus less dependency on a well-documented requirement specification document (Chilenski & Miller, 1994).

iii. The low-level testing of individual components which is unit testing to verify the implementation of the software at code development level promotes early error detection (Gittens, Romanufa, Godwin, & Racicot, 2006).

This research has chosen to couple risk-based testing with test coverage adequacy measurement because it is observed that the number of failures revealed in testing is also related to how much coverage is set by the current test set (Cai & Lyu, 2007).

2.7 Summary

This chapter has described and discussed on literatures which contributed and are related to this research. This research focused on providing a suitable software test strategy for software testing domain which falls into agile methodology software development within the scope of Scrum management process. The summary of literature review within this chapter is illustrated in Figure 2.4 for better understanding.

Universiti Malaya

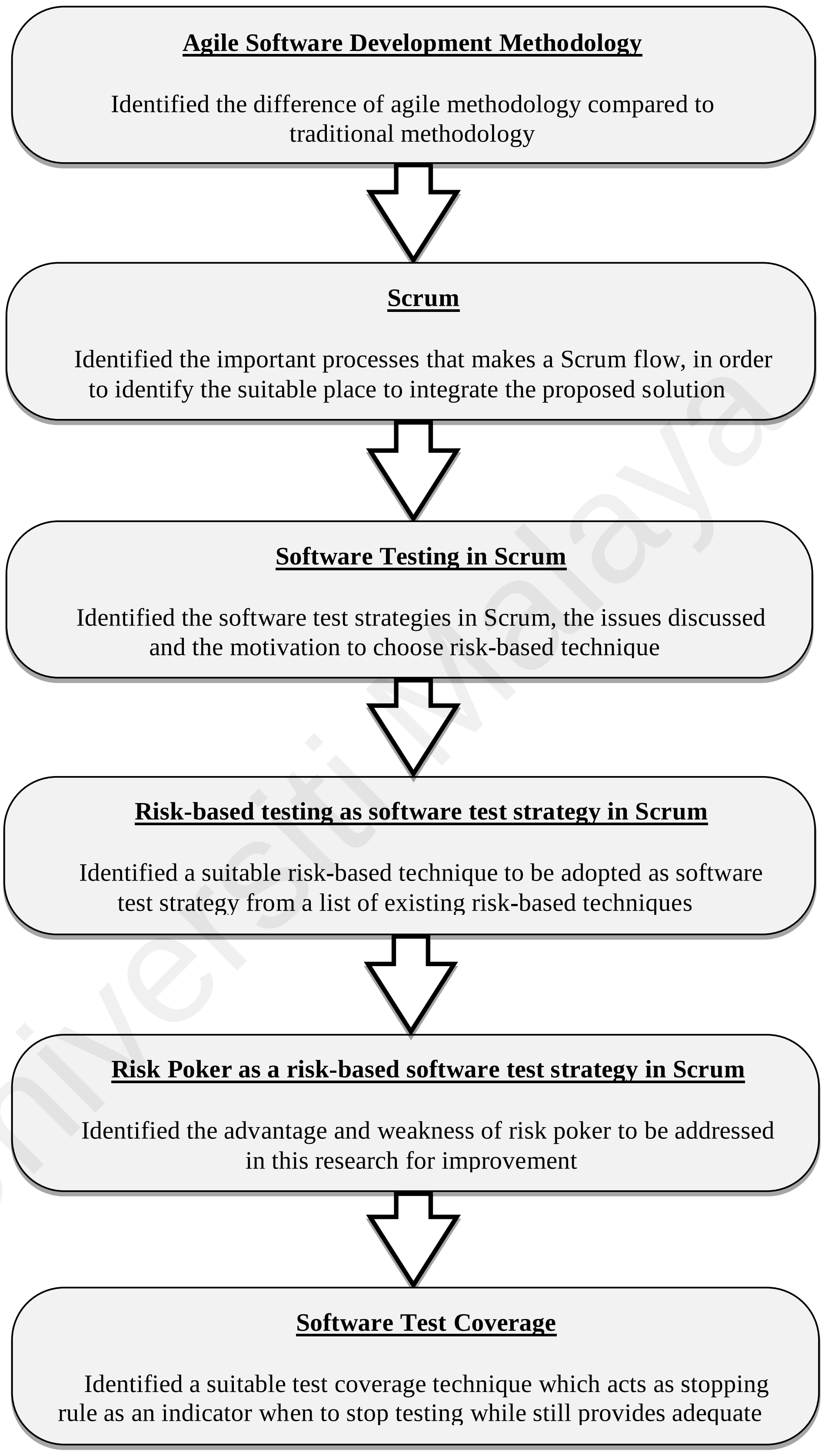

Agile Software Development Methodology
Identified the difference of agile methodology compared to traditional methodology
Scrum
Identified the important processes that makes a Scrum flow, in order to identify the suitable place to integrate the proposed solution
Software Testing in Scrum
Identified the software test strategies in Scrum, the issues discussed and the motivation to choose risk-based technique
Risk-based testing as software test strategy in Scrum
Identified a suitable risk-based technique to be adopted as software test strategy from a list of existing risk-based techniques
Risk Poker as a risk-based software test strategy in Scrum
Identified the advantage and weakness of risk poker to be addressed in this research for improvement
Software Test Coverage
Identified a suitable test coverage technique which acts as stopping rule as an indicator when to stop testing while still provides adequate

Figure 2.4: Summary of Literature Review Flow Diagram

Universiti Malaya

CHAPTER 3: RESEARCH METHODOLOGY

This chapter describes the process flow employed as research methodology used in completing this research. The process flow ensures this research is on track and able to achieve the objectives defined in chapter 1. The process that is also recognized as research methodology is explained in the next section.

3.1 Research Methodology

Research methodology used in this research consists of literature review phase, followed by model building of the proposed solution, developing the prototype and ended with validation phase. Each process has its own objective to achieve as illustrated in Figure 3.1 below. The details of these processes are explored in the following section.

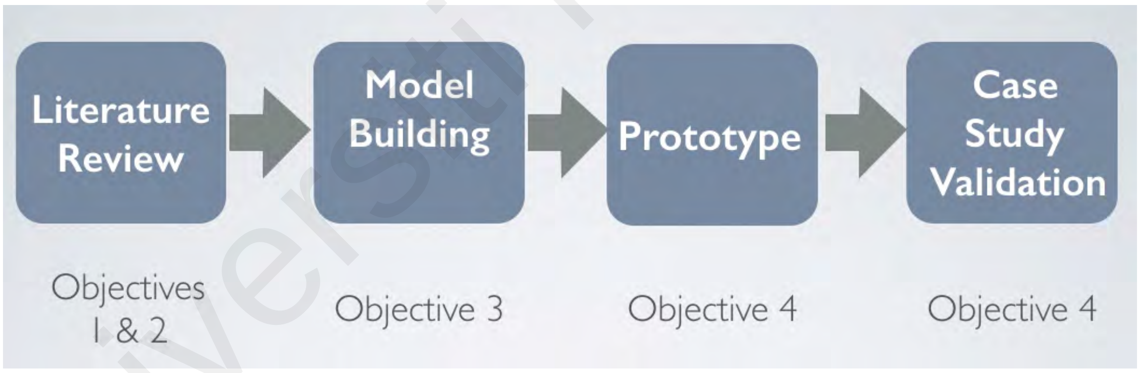


Figure 3.1: Research methodology process flow

3.1.1 Literature Review

In the process of literature review, this research executes this phase to make sure research objective 1 and 2 as defined in chapter 1 are achieved respectively. All literature related to this research is reviewed to help this research decides on the design of the proposed solution. In the following section, information about the literature review of this research is provided in the form of; 1) Sources of the literature, 2) Literature review, and 3) Findings of the literature review.

Universiti Malaya

3.1.1.1 Sources of the Literature Review

The sources of literature review for this research are from research articles, conference papers, journals, articles from periodical online magazines and books. Initially, searches of the most cited articles and studies on software testing current trend were grabbed from the following databases: ACM, IEEE Xplore, ScienceDirect and ISI Web of Science databases. After the search in the databases, a search for the keywords was performed in Google Scholar to collect any relevant papers falling short of the original search. From there, related research and studies are searched and reviewed for current trend focus.

3.1.1.2 Reviewing the Literature Review

This research's works started with reviewing previous studies of software testing research domain for agile methodology. A study by (Kettunen et al., 2010) on the comparison of agile process and traditional process development in industrial world has bring interest to this research to explore more on agile software development environment compared to traditional one as many of the software project out there adopted agile methodology, but there is not much research on software test process efficiency available for agile environment as reference.

This research reviewed a study by (Crispin & Gregory, 2009) at first as it provide meaningful information on agile testers as a start, and following, this research reviewed other related studies in this area for more insight. After many review of agile and software testing studies, this research has narrowed down the research domain to focus on software testing strategy for Scrum, as Scrum has been widely chosen as the popular project management for agile project and it implicitly apply risk-based prioritization for the tasks assigned in each sprint. Various Scrum studies are reviewed to understand how to manage software development in Scrum methodology, like (Schwaber, 2004) to

Universiti Malaya

name an example and how software testing fits into the development iteration (James, 2007).

3.1.1.3 Findings of the Literature Review

Having reviewed the studies and research areas mentioned in previous section, this research explored on existing agile methodology or framework that has been used in industry as listed in Table 2.1 in Chapter 2, Section 2.1. Next, this research moved forward by investigating the issues and problems arose specifically in software testing domain for Scrum project management. Current issues and challenges of agile software development with Scrum are reviewed, which is also discussed in Chapter 2, Section 2.2.1.

Next, this research concentrates on reviewing various studies of software test strategy in agile project relevant with the current issues in Scrum to gain the overall view of current issues and problems to be resolved, as highlighted in Table 2.3 and Table 2.4 in Section 2.3, Chapter 2.

To highlight, one of the issues discussed repeatedly is that literature claims that group discussion could add influence to the participants upon making decisions, despite one of the important agile characteristic being a self-managing team. In addition to that, (Karlsson & Martensson, 2009) did mention that in the self-managing team, it is required that team members participate equally in the analyses process, which is not the case in most situation in traditional software testing strategy due to the nature of defined process control model. Lastly, the reviewed studies also mentioned the problem of defining test adequacy for customer assurance, especially for a short period time-boxed iteration in agile project.

Universiti Malaya

From the literature reviewed, this research has come up with decisions to apply risk-based approach as risk-based testing strategy. Upon studying the Scrum project management studies, this research has come across a popular estimation technique that applies group discussion upon achieving decision and estimation which could address the problem of self-managing teamwork for software testing called Risk Poker. On resolving test adequacy matter on how much testing is enough or needed for the developed software, this research proposed to use the group estimation technique to analyze risks and assign relatively appropriate test coverage based on the risks prioritized. The reasons that lead this research to make the above decisions based on the thorough literature reviews, are as listed in Table 3.1. Once objectives 1 and 2 identified, the next step is to construct a model for the proposed solution as described in the following section.

Table 3.1: Decisions made based on the Literature Review

Category	Available methods or tools	Selected method	Reason
Software Test Strategies	• Risk-based strategy • Requirement-based strategy • Model-based strategy	Risk-based strategy	• Literature revealed that most of agile project adopt risk-based approach
Risk-based Testing Techniques	• Cost of Exposure • Pattern-based Methodology Tailoring • Requirement Analysis using Goal Graph • RiteDAP • Risk Poker	Risk Poker	• Group consensus in risk analysis fits important agile characteristic; Group discussion & self-managed team • Can be adopted into agile environment without modification

Universiti Malaya

Category	Available methods or tools	Selected method	Reason
Test Coverage	• Code Coverage • Requirement Coverage • Structural Coverage • Architectural Coverage	Code Coverage	• Able to measure fault exposure effectiveness • Can be used as stopping rule to define test adequacy

3.1.2 Model Building

The proposed solution focuses on proposing a testing model which suits agile environment especially Scrum. The risk analysis result of the proposed solution provides adequate test coverage estimation for testing activity. The proposed solution obtained risk analysis for the product under development by achieving group consensus amongst team members from various expertise which will implicitly promotes knowledge sharing for more accurate analysis and uncover hidden issues. The proposed solution also provides estimation of test coverage on how much testing is needed based on the risk analysis. To support the proposed solution, a prototype to produce a risk graph which prioritize tasks for the sprint based on the risk analysis result, and provide test coverage estimation information is developed as explained next in section 3.2.3. More details on model building for the proposed solution are described in chapter 4.

3.1.3 Prototype

A prototype to display the risk graph is developed for the proposed solution to display the prioritized items and the estimated test coverage. This prototype shows that the analyzed tasks listed in the risk graph produced from the implemented proposed solution is prioritized correctly with the estimated test coverage information for the sprint. More information on the risk graph produced by this prototype is explored in

Universiti Malaya

chapter 4. The remaining objective of this research is related to the case study validation which is explained in the following section.

3.1.4 Experiment Validation

This research reviewed previous studies as example to plan and design the experiment. Existing case study and research which can be used as an example to validate the proposed solution are mostly related to poker technique as an estimation technique. Thus, previous studies on poker technique in software engineering domain which is mostly about planning poker effectiveness as staffing effort estimation technique in agile project are studied.

Previous study by (Haugen, 2006) and (Molokken-Ostvold & Haugen, 2007) are aimed to see whether planning poker technique could provide better accuracy on staffing effort estimation compared to unstructured group technique that does not apply group discussions for group estimation. The experiment in these studies is on measuring the accuracy difference between planning poker technique as group estimation and individual statistical group combination. (Moløkken-Østvold et al., 2008) and (Mahnič & Hovelja, 2012) both evaluate the effectiveness of planning poker technique as compared to individual statistical group combination which also applies group discussions, similar to planning poker, called hybrid Delphi technique. The difference between these two studies is; (Moløkken-Østvold et al., 2008) was to estimate tasks for current sprint, while (Mahnič & Hovelja, 2012) experiment was to estimate bigger scale development effort, which is user stories for all development sprints in the project.

Apart from planning poker studies, study on protection poker as a software security risk analyses technique by (Williams et al., 2009) offers the result of the effectiveness of applying protection poker in students project and in an industrial project where by

Universiti Malaya

applying protection poker for software security analyses, it has uncover hidden issues, broaden knowledge sharing and improved software security analyses amongst team member.

Hence, based on these studies, this research has designed an experiment to evaluate the effectiveness of risk poker as software test strategy for risk analyses and provide test coverage estimation by comparing the result of using risk poker technique to estimate test coverage with the averaged statistical combination of individual estimations as described in chapter 5. The experiment is also designed to be able to provide the answer of the research questions as defined in chapter 1.

Once the experiment is executed as planned, and the required data are collected, this research analyzes the result of the experiment to achieve research's final objective as explained objective 4 in chapter 1. This research compares the total of fault exposed during testing activity with seeded fault embedded in the system for both controlled group and experimental group. Thus, having reviewed previous studies on poker techniques, this research measures the accuracy of risk analyses and test coverage of risk poker compared to the averaged statistical combination of individual estimations using the calculation of balance measure of relative error (BRE), where;

$BRE = \frac{|seeded_{fault} - exposed_{fault}|}{\min(seeded_{fault}, exposed_{fault})}$ which is explained with further details in Chapter 5.

This research depends mainly on case study validation to see the effectiveness of the proposed technique. However, due to limitation of resources, it was not possible to apply this software testing technique or experiment in real life project and will be recommended as future work as discussed in chapter 6.

Universiti Malaya

3.2 Summary

In this chapter, this research described the flow of process that has been chosen to be followed in order to plan and execute current research. There are four main steps or process throughout this research methodology which are; 1) Literature review, 2) Model building, 3) Prototype, 4) Case Study Validation. In the following chapter, this research described the proposed solution in details which covers model building explanation and prototype construction.

Universiti Malaya

CHAPTER 4: MODEL BUILDING

This chapter explores the construction of testing model as the proposed solution for this research in detail and how the proposed model integrates into a Scrum methodology for agile project. Section 4.1.1 describes Risk Poker technique as the selected risk-based testing technique for the proposed testing model. In section 4.1.2, Risk Poker technique is combined with test coverage to provide test coverage estimation for the proposed model. While in section 4.1.3, the result of risk analysis performed using the proposed model is translated into a Risk Graph prototype which contains all information of the risk analysis result. Lastly, in section 4.1.4, the proposed testing model is integrated inside a Scrum methodology.

4.1 Testing Model Using Risk Poker Technique for Scrum-based Software Development Projects

Most of Scrum practitioners employ Planning Poker technique to estimate tasks and staffing effort for sprint (Haugen, 2006; Molokken-Ostvold & Haugen, 2007). Planning Poker is a popular choice because of the face-to-face group discussion and self-managing characteristics which is important for agile practitioners (Mahnič & Hovelja, 2012; Moløkken-Østvold et al., 2008). Planning Poker is a process of group estimation for user stories used in planning releases and iterations (Grenning, 2002) where it will be used to plan which features to be implemented and estimate staffing effort of the development team for the sprint. In Scrum, Planning Poker process is executed during Sprint Planning Meeting by The Team and Product Owner.

Recently, (Van de Laar, 2012) has proposed Risk Poker technique as a risk-based testing approach for agile projects which can be executed alongside with Planning Poker. Similarly, in Risk Poker, group consensus is achieved upon deciding the color

Universiti Malaya

cards risk level, instead of traditional individual risk assessment calculation for user stories. Risk poker technique is implemented as a risk-based software testing strategy in Scrum's Sprint Planning Meeting for risk analyses and this research combines code coverage technique to provide estimation of how much test coverage is needed for the tasks to be developed in the sprint as a testing model for Scrum methodology.

Like Planning Poker, Risk Poker technique is a face-to-face discussion and provide group consensus decision for determining risk level and estimate how much testing is needed for the user stories listed in product backlog to be developed in the sprint. As group discussion and self-managing team is one of the important agile characteristics, this technique is suitable for Scrum practitioners to be implemented in their process as a testing process for Scrum. Upon implementing Risk Poker technique, the team could also improve knowledge expertise, to be able to be responsible for the quality equally amongst team members, to be able to decide on how much test coverage is needed based on equal understanding amongst team member and to be able to practice cross functional job if needed during sprint as everybody is in the same level of knowledge sharing. Next, following section explores on how Risk Poker technique works as a risk-based testing in details.

Universiti Malaya

4.1.1 Risk Poker as a Risk-based Testing

In Risk Poker technique, risks are identified and discussed with team members and risks prioritization are achieved through group consensus. In a traditional risk prioritization, risks exposure is calculated to prioritize risks. The formula is as follows: $RE = P \times C$ (Amland, 2000; Bannerman, 2008; B. Boehm, 1989; Stallbaum et al., 2008), where;

- *RE* is the risk exposure,
- *P* is the probability of the risk to happen, also known as likelihood of the risk to occur, and
- *C* is the cost also known as impact that will affect the project, if the risk happens.

Usually, an expert professional in the project management is responsible to assign score or weigh factor to the probability (*P*) and cost (*C*) of the identified risk according to his or her judgment. The risk exposure (*RE*) is then calculated and prioritized accordingly. In Risk Poker, instead of relying on an individual expert judgment which might overlook some issues upon estimating risks regarding product development, team members are responsible to rate the probability of risks using colored rating card, which is called likelihood factor in this research. While for the cost factor, product owner and stakeholders are responsible to discuss and estimate the cost of the risk which is called impact factor in this research.

Rating Impact Factor Risks

In the Scrum process, upon listing product backlog items, Product Owner discusses with stakeholders to collect all required features for the product to be developed in a form of user stories. Product Owner translates these user stories into product backlog items. Once all required features are collected and listed in the product backlog, Product Owner discusses the product backlog items with stakeholders to identify risks and costs that will affect the project. Thus, at the end of the discussion, Product Owner and stakeholders decide on risk level appropriate for each of the product backlog item which will impact business, user's needs and project as a whole. This is called impact factor risk identification.

Universiti Malaya

Rating Likelihood Factor Risks

When product backlog items are ready, a Sprint Planning Meeting is called by Product Owner. In the meeting, the team and product owner discuss on the selected user story to estimate staffing effort and assign tasks. However, in this research, for a Scrum process which applied Risk Poker technique, during discussion amongst The Team and Product Owner, everyone is required to identify all risks related to the item in discussion and discusses the risks thoroughly in terms of developer perspective, tester perspective and user perspective. When everyone has equal knowledge and awareness, risks level assignment takes place using Risk Poker technique to ensure the team understands the risks thoroughly and able to rate and manage them equally as a team. These risk level assignment is called likelihood factor identification which would indicate how much testing is needed for unit test.

Risk Poker Activity Flow

Universiti Malaya

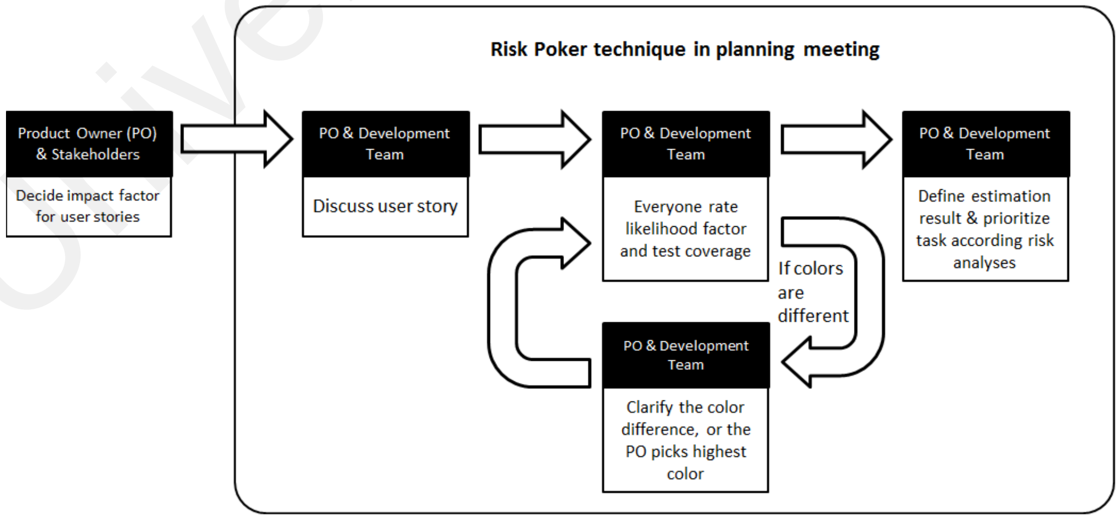

Risk Poker technique in planning meeting
Product Owner (PO) & Stakeholders
Decide impact factor for user stories
PO & Development Team
Discuss user story
PO & Development Team
Everyone rate likelihood factor and test coverage
If colors are different
PO & Development Team
Clarify the color difference, or the PO picks highest color
PO & Development Team
Define estimation result & prioritize task according risk analyses

Figure 4.1: Detailed view of Risk poker activity flow diagram (Van de Laar, 2012)

Figure 4.1 is drawn to show the activity of Risk Poker technique in detail. The flow starts with:

- Product Owner and stakeholders decide on impact factor for the user stories based on how much the user story would affect the end user, business and project.
- Following, Product Owner presents the user stories to the Team members during Sprint Planning Meeting without disclosing the impact factor rating. In the meeting, team members ask questions and discuss the user stories presented until they are satisfied. During the discussion, they identify and analyze what risks associated with the user stories; where should, if any of the risk happens, it will affect the quality of the product or even cause failure to the product. This risk is classified as likelihood factors for the discussed user story. In the discussion, the team discusses the user story thoroughly both from the eye of developer and tester. They are equally responsible for the product quality, thus they are required to understand all risks associated with the user story and together shares their concerns based on their expertise for the item to be considered and discussed.
- When everybody in the team is clear with the user story and their risks, the team is then required to estimate risk level for the user story and assign test coverage they think would be enough for testing activity. Thus, team members will be given a set of card which contains a table of four colored (green, yellow, orange and red) boxes to rate the risk level of likelihood for the user story as shown in Figure 4.2. The four colored risk factor is proposed by (Van de Laar, 2012). In addition to that, a study by (Noor & Khan, 2014) also classified and discussed defect prioritization in terms of four coloured priority level. Team members are required to rate the likelihood factor individually where 'red' represents the highest estimation factor and 'green' as the lowest estimation factor. When everyone has finished with the estimation

Universiti Malaya

individually, team members are required to simultaneously show their estimation in the group. If there is any huge difference of estimation color, the estimator should explain the difference and discussion will take place once again. When everyone is satisfied with the discussion, they are given a new set of estimation card and they are required to do the estimation individually once again and show the result simultaneously afterwards.

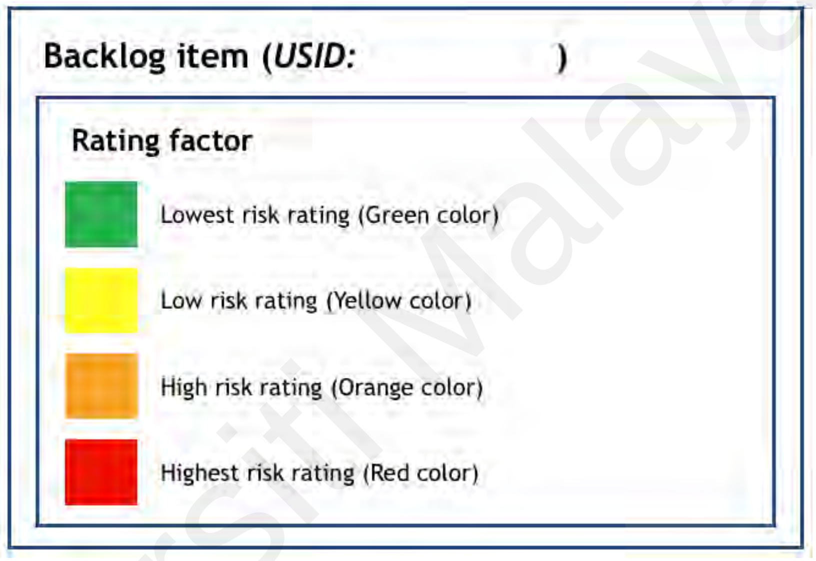

Backlog item (*USID:*)
Rating factor
Lowest risk rating (Green color)
Low risk rating (Yellow color)
High risk rating (Orange color)
Highest risk rating (Red color)

Figure 4.2: Rating card for Impact factor and Likelihood factor

- If the results still show difference of color assignment for the estimation factor, Product Owner has the right to assign the highest estimation for the likelihood factor for the user story affected. Otherwise if no difference of estimation showed up and group consensus is achieved, product owner records the rating of likelihood factor.

Universiti Malaya

In regard of impact factor, Product Owner is responsible to present them to the team members and justify the rating at the end of the discussion. Team members are allowed to ask questions and discussion will take place until everyone is satisfied over the impact factor rating. Risk analyses obtained through Risk Poker technique provides estimation of test coverage, which is explored in the following section.

4.1.2 Risk Poker and Test Coverage Estimation

At the end of Sprint Planning Meeting, the team has obtained risk rating for each of the discussed backlog items. At this stage, the product backlog items have been broken out into smaller tasks to be committed to by team members. The tasks are stored as Sprint Backlog Items which have rating color assigned to them as discussed in planning meeting. The rating color assigned provides estimation of test coverage to determine how much testing is needed throughout sprint. Testing activity will be executed based on the test coverage estimation to obtain a “done” criteria as a show stopper for testing activity before presenting the finished product to the user at the end of sprint. At the end of the sprint, the team will deliver the finished product with the test coverage report as one of the criteria acceptance for quality assurance upon delivering the finished product as promised.

A set of test coverage technique is assigned for each rating color associated with likelihood and impact factor to estimate how much testing is needed. Test coverage technique assigned for unit test is Code Coverage to test development coding to find error and expose fault. Table 4.1 describes the code coverage definition for test coverage defined for unit test.

Universiti Malaya

Table 4.1: Coverage complexity for unit test (Thomas Müller, 2011)

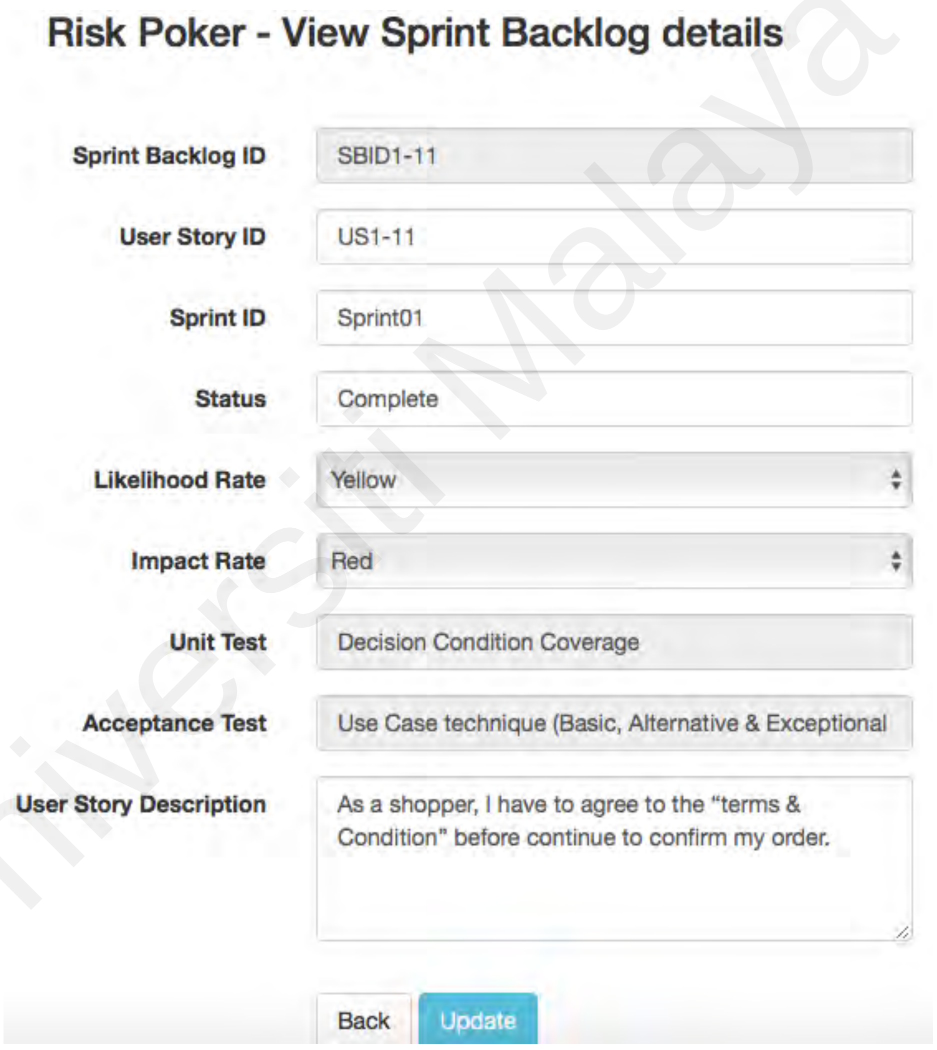
Rating	Test coverage complexity
Lowest (Green)	Decision coverage: 100% decision coverage implies both 100% branch coverage and 100% statement coverage.
Low (Yellow)	Decision condition coverage: 100% decision condition coverage implies both 100% condition coverage and decision coverage.
High (Orange)	Condition determination coverage: 100% condition determination coverage implies 100% decision condition coverage.
Highest (Red)	Multiple condition coverage: 100% multiple condition coverage implies 100% condition determination coverage.

Universiti Malaya

The test coverage complexity is defined in a test coverage table in the prototype system. At the end of Sprint Planning Meeting, team members are required to update Sprint Backlog Items with the rating color which will be matched to the estimated test coverage by the prototype system automatically. Team members are able to view Risk Graph exported by the prototype system for the prioritized Sprint Backlog Items as reference for the team to work on the tasks according to the highest priority. Details on the prototype system to display the Risk Graph are described in the following section.

4.1.3 Prototype System: Risk Graph

At the end of the Sprint Planning Meeting, the discussed user story is updated in the Sprint Backlog database through the prototype system interface as shown in Figure 4.3 together with the risk rating color assigned both for likelihood and impact factor. The risk rating will determine how thorough a testing will be done in the sprint. The prototype system allows the team member to insert, update and delete Sprint Backlog Items from and into the database. Each Sprint Backlog Items that has rating color assigned to it is then matched to the estimated test coverage needed for unit testing and acceptance testing for team member's reference in order to execute test in sprint.



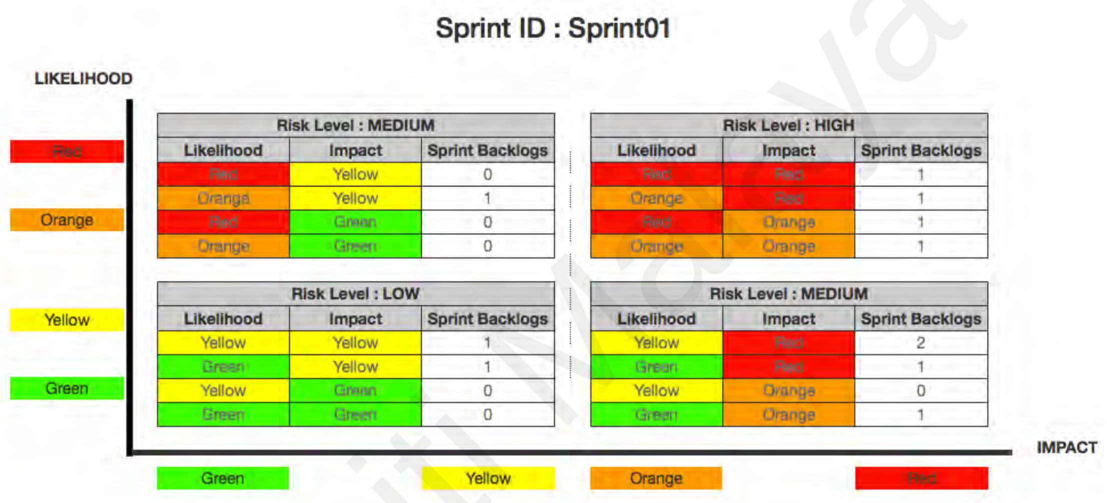
Risk Poker - View Sprint Backlog details

Sprint Backlog ID	SBID1-11
User Story ID	US1-11
Sprint ID	Sprint01
Status	Complete
Likelihood Rate	Yellow
Impact Rate	Red
Unit Test	Decision Condition Coverage
Acceptance Test	Use Case technique (Basic, Alternative & Exceptional
User Story Description	As a shopper, I have to agree to the "terms & Condition" before continue to confirm my order.

Universiti Malaya

Figure 4.3: Update Sprint Backlog details

Once Sprint Backlog database is updated for the sprint, the team is able to view Risk Graph prioritization to see which item is prioritized from the highest risk level to the lowest risk level as shown in Figure 4.4. Risk levels are categorized into High, Medium and Low grids. Each risk level grid shows a table that consists of rating color for both likelihood and impact factor assigned, and the total of related user stories. Instead of traditionally calculating the risk exposure for each of the sprint backlog items, the prototype system pairs the rating color of likelihood factor and impact factor to prioritize risk exposure as shown in Figure 4.4.



Risk Level : MEDIUM		
Likelihood	Impact	Sprint Backlogs
Red	Yellow	0
Oranga	Yellow	1
Red	Green	0
Orange	Green	0

Risk Level : HIGH		
Likelihood	Impact	Sprint Backlogs
Red	Red	1
Orange	Red	1
Red	Orange	1
Orange	Orange	1

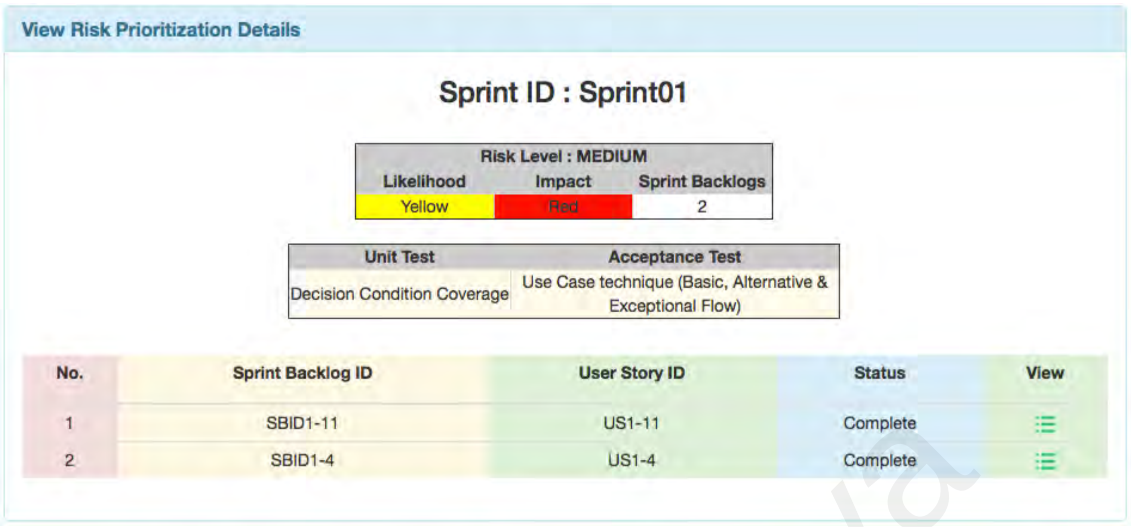
Risk Level : LOW		
Likelihood	Impact	Sprint Backlogs
Yellow	Yellow	1
Green	Yellow	1
Yellow	Green	0
Green	Green	0

Risk Level : MEDIUM		
Likelihood	Impact	Sprint Backlogs
Yellow	Red	2
Green	Red	1
Yellow	Orange	0
Green	Orange	1

Figure 4.4: Risk Graph prioritization

Once the Risk Graph prioritization is exported, the team will choose to develop and test on the sprint backlog items placed in high risk area first, followed by medium risk level items as plotted in the graph. As mentioned previously, the risk analysis obtained from Risk Poker provides test coverage definition both for unit test and acceptance test. Team members are able to see the details of test coverage estimated for sprint backlog items by clicking the total number of corresponding sprint backlogs of the risk level. Test coverage estimated for the corresponding sprint backlogs is as shown in Figure 4.5.

University Malaya


View Risk Prioritization Details

Sprint ID : Sprint01

Risk Level : MEDIUM		
Likelihood	Impact	Sprint Backlogs
Yellow	Red	2

Unit Test	Acceptance Test
Decision Condition Coverage	Use Case technique (Basic, Alternative & Exceptional Flow)

No.	Sprint Backlog ID	User Story ID	Status	View
1	SBID1-11	US1-11	Complete	☰
2	SBID1-4	US1-4	Complete	☰

Figure 4.5: Test coverage estimation for the risk level Medium

Thus, based on the estimated test coverage for the corresponding sprint backlogs, testers will construct and execute test accordingly. In this research, the efficiency of these test suites in detecting fault during experiment validation will determine the effectiveness of Risk Poker technique as risk-based testing in order to provide test coverage estimation. Details of this prototype system are explored in the following section for further understanding.

4.1.3.1 Prototype Functional Requirement

Similar to other system development, this prototype has a number of functional requirements as listed in Table 4.2:

Table 4.2: Functional Requirement of Risk Graph prototype

ID	Functional Requirement
FuncReq01	Prototype system should be able to provide a fully functional database for Sprint Backlog Items as record.
FuncReq02	Prototype system should be able to match test coverage for Sprint Backlog Items in the database automatically
FuncReq03	Prototype system should prioritize sprint backlog items in Risk Graph according to the rating color and display the Risk Graph accordingly.

Universiti Malaya

ID	Functional Requirement
FuncReq04	Prototype system should provide details of the estimated test coverage for unit test and acceptance test for sprint backlog items according to risk level

4.1.3.2 Prototype Non-Functional Requirement

Apart from functional requirements listed in the previous section, the prototype system also has some non-functional requirements to be adhered to in prototype development. The list of non-functional requirements is as shown in Table 4.3.

Table 4.3: Non-Functional Requirement for Risk Graph prototype

ID	Non-Functional Requirement
NonFuncReq01	Interface Requirement: The interface should be user friendly, easy database update and easy to understand data population.
NonFuncReq02	Scalability Requirement: The coding and development of the prototype system should be optimized, structured and cached systematically.

4.1.3.3 Programming Language

The Risk Graph prototype system is a web-based system developed using PHP scripting language for web development. The database support of the prototype system is MySQL database. The coding of PHP language and MySQL database for the prototype system is optimized and structured to fulfill the non-functional requirement of NonFuncReq02 as mentioned in the previous section.

Universiti Malaya

4.1.3.4 User Interface Diagram

The prototype system of Risk Graph has easy-to-navigate user interfaces as defined in non-functional requirement NonFuncReq01. The interfaces designed for this prototype are simple, clear and easy to understand as illustrated in Figure 4.6 below.

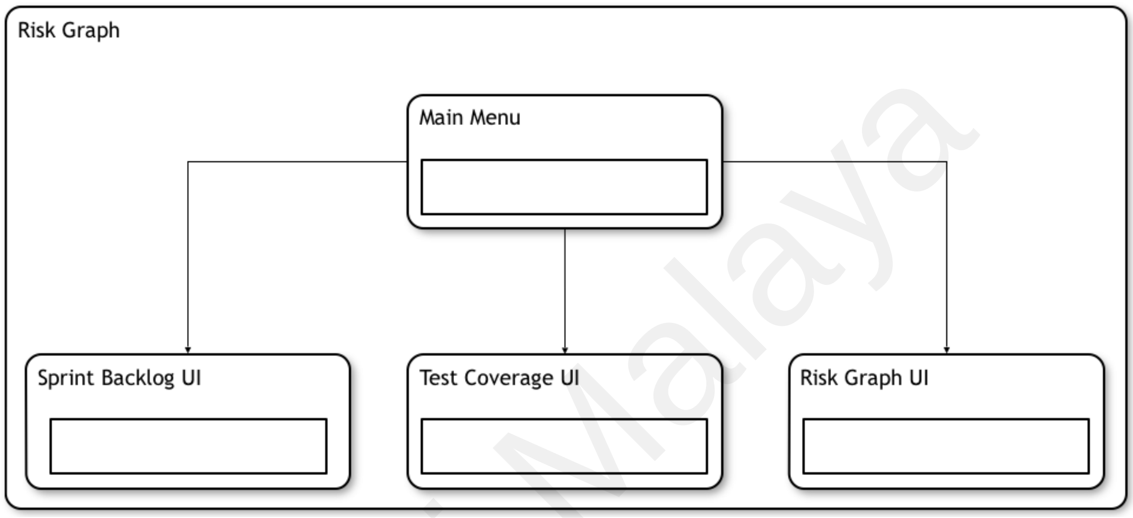


Figure 4.6: Risk Graph prototype interface structure

There are three main user interfaces designed for this prototype system to fulfill the functional requirement as mentioned in section 4.1.3.1. The Sprint Backlog UI should be able to address the FuncReq01, while Test Coverage UI addressed FuncReq04 and lastly Risk Graph UI addressed FuncReq03.

4.1.3.5 Operations in the Prototype

This section describes the use case diagram and the process flow of the prototype system in details. The prototype use case diagram is as illustrated in Figure 4.7.

Universiti Malaya

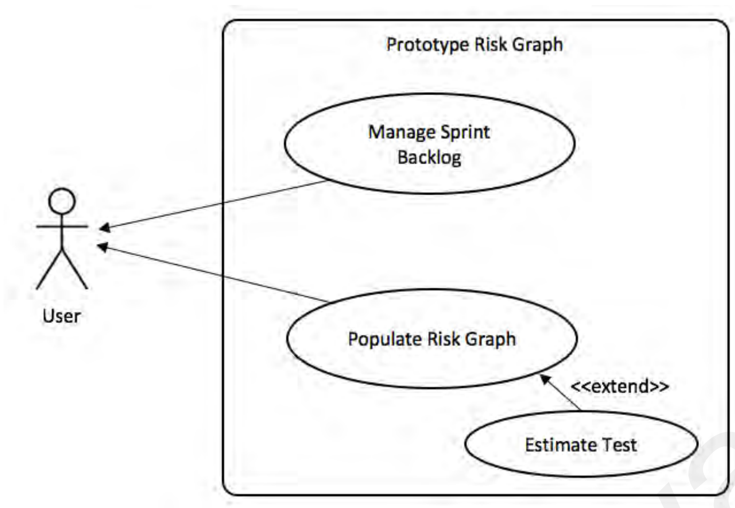


Figure 4.7: Use Case Diagram of the Prototype Risk Graph

"Manage Sprint Backlog" use case as defined in Figure 4.7 addressed the functional requirement FuncReq01 and FuncReq02 in order to provide fully functional database for Sprint Backlog Items. "Populate Risk Graph" use case addressed the functional requirement FunReq03 to provide prioritized Sprint Backlog Items in a risk graph with risk level defined accordingly. Lastly "Estimate Test Coverage" use case is only available if "Populate Risk Graph" use case is developed, as this use case provides the estimation of test coverage needed for the prioritized Sprint Backlog Items as required by FuncReq04.

The use case diagram defined has fulfilled all the required functions of the prototype system. Next, the prototype system process flow is defined as shown in the process flow design in Figure 4.8.

Universiti Malaya

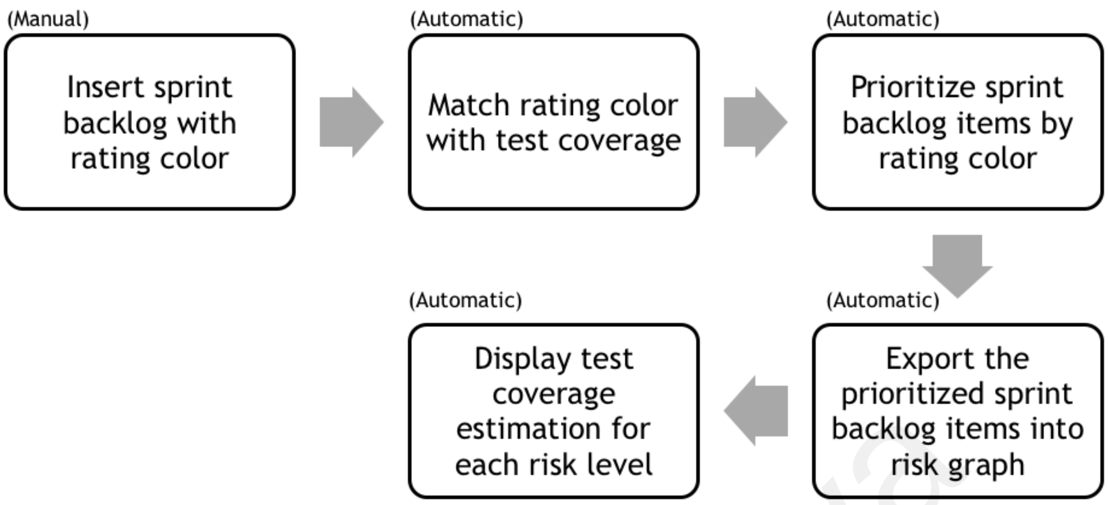

(Manual)
Insert sprint backlog with rating color
(Automatic)
Match rating color with test coverage
(Automatic)
Prioritize sprint backlog items by rating color
(Automatic)
Export the prioritized sprint backlog items into risk graph
(Automatic)
Display test coverage estimation for each risk level

Figure 4.8: Process Flow of Prototype Risk Graph

The process flow of the prototype system starts with user insert sprint backlog items into the database with the respective rating color at the end of Sprint Planning Meeting. Once the Sprint Backlog is updated, the prototype system matches the rating color with test coverage estimation automatically. After that, the prototype system automatically prioritizes sprint backlog items according to the rating color. This prioritization process is then translated into Risk Graph which is displayed in the Risk Graph UI for user's reference. Lastly, the prototype system allow user to display test coverage estimation for each risk level for testing in the sprint.

Universiti Malaya

4.1.4 Integrating the proposed Testing Model inside a Scrum Methodology

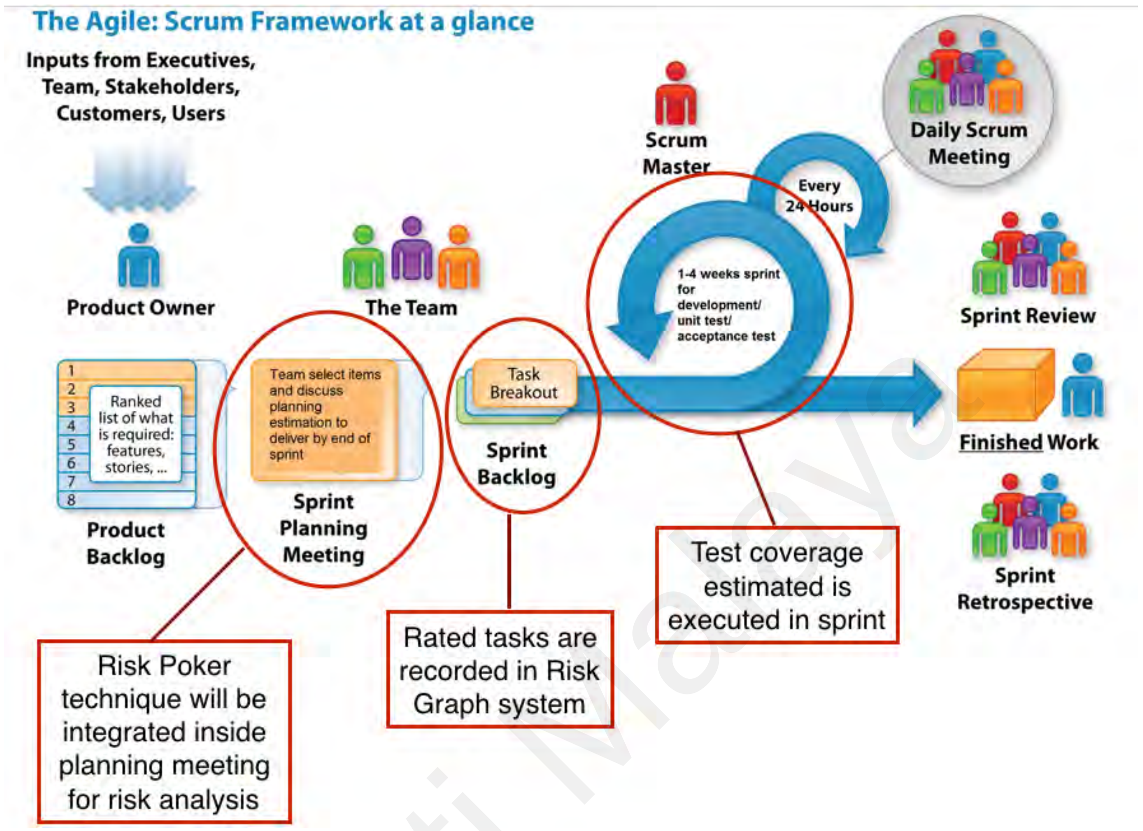

The Agile: Scrum Framework at a glance
Inputs from Executives, Team, Stakeholders, Customers, Users
Product Owner
The Team
Scrum Master
Daily Scrum Meeting
Every 24 Hours
1-4 weeks sprint for development/ unit test/ acceptance test
Sprint Review
Finished Work
Sprint Retrospective
Ranked list of what is required: features, stories, ...
Product Backlog
Team select items and discuss planning estimation to deliver by end of sprint
Sprint Planning Meeting
Task Breakout
Sprint Backlog
Test coverage estimated is executed in sprint
Rated tasks are recorded in Risk Graph system
Risk Poker technique will be integrated inside planning meeting for risk analysis

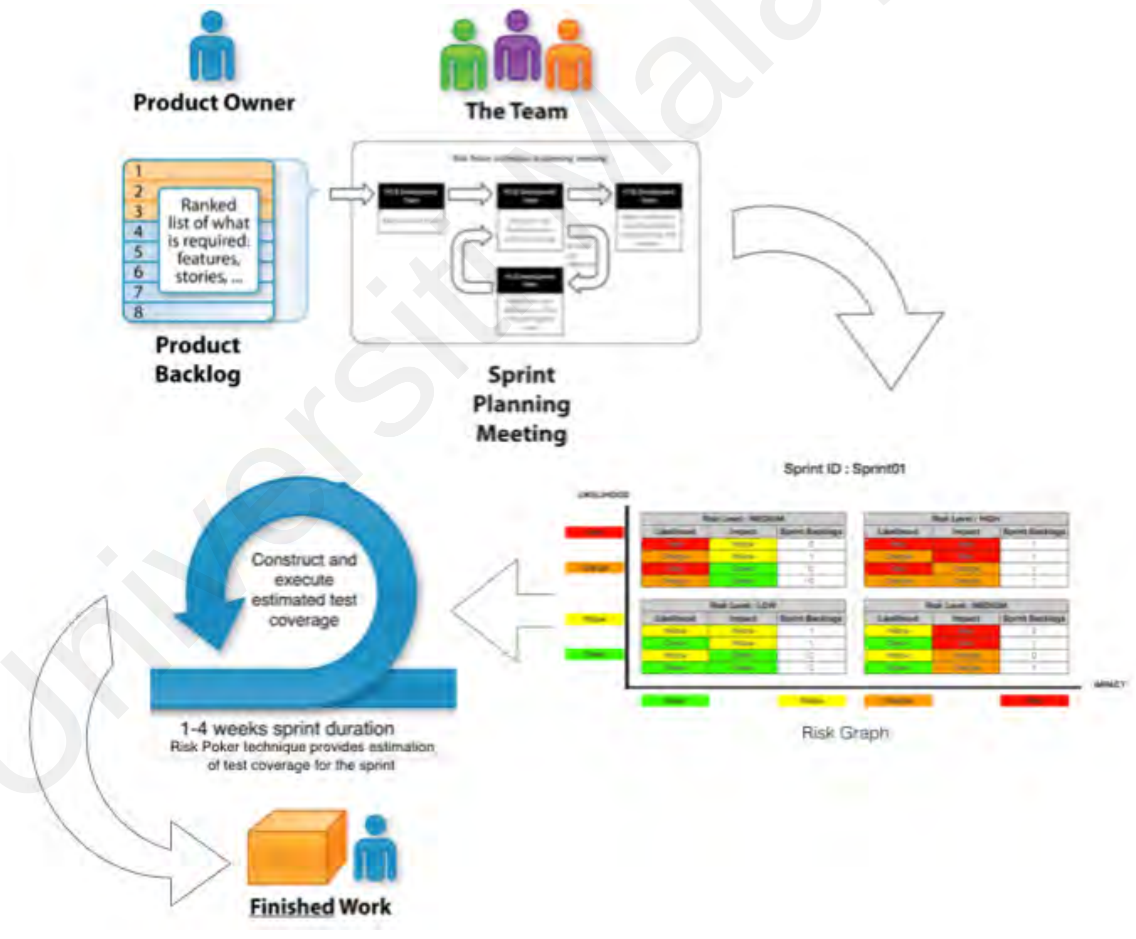
Figure 4.9: Proposed Software Testing Strategy incorporated in Scrum workflow

Figure 4.9 shows where the proposed testing model incorporated into the Scrum work flow. The proposed sections are mentioned through red circles that highlight the position where Risk Poker technique is integrated with Scrum processes. The proposed testing model is integrated into Scrum work flow and affected two particular processes: (i) Sprint Planning Meeting and (ii) Sprint Backlog. In Product Backlog listing and Sprint Planning Meeting, the Product Owner along with the team members discuss the user stories and focus on the risks involved. The team also decides the risks rating to estimate test coverage.

Universiti Malaya

This integration will provide a testing model for Scrum methodology in terms of;

1) The approach used for risk-analyses of user stories amongst team members which will uncover any possible hidden or unseen risks,
2) Better knowledge sharing between different background to improve decision on risk level and test coverage. Figure 4.10 shows the integrated testing model inside Scrum process.



Product Owner
The Team
Ranked list of what is required: features, stories, ...
Product Backlog
Sprint Planning Meeting
Sprint ID : Sprint01
Construct and execute estimated test coverage
1-4 weeks sprint duration
Risk Poker technique provides estimation of test coverage for the sprint
Risk Graph
Finished Work

Figure 4.10: Integration of the Proposed Testing Model with Scrum Work Flow

Integrating Risk Poker technique as a risk-based testing in Scrum would improve both risk analyses process and test coverage as risk poker is able provide a group consensus upon analyzing risks and estimating test coverage. In a big picture, Risk

University Malaya

Poker technique will affect the following processes in Scrum, as shown previously in Figure 4.10; 1) Listing Product Backlog, 2) Sprint Planning Meeting, 4) Update Sprint Backlog to manipulate Risk Graph prioritization, and 5) Testing activity in sprint.

For the testing activity inside sprint, testers are going to generate and execute test cases manually or using tools, and bug fixing is done in parallel in the sprint as shown in Figure 4.11. How much testing is needed and what type of test method to be applied has been estimated in Sprint Planning Meeting through Risk Poker, thus testers execute test as per estimated and provide test result as a report to be presented to customer as "done" criteria upon delivering finished work at the end of the sprint.

Universiti Malaya

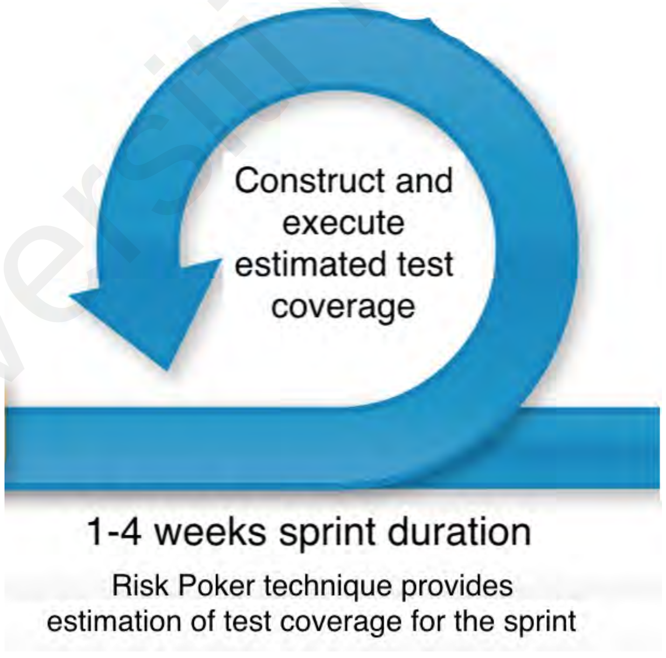

Construct and execute estimated test coverage
1-4 weeks sprint duration
Risk Poker technique provides estimation of test coverage for the sprint

Figure 4.11: Risk Poker Technique Affect Testing Activity in Sprint

Once the sprint duration is finished and end product is delivered to customer, the same scenario described in this section will be repeated again for the next batch of user stories defined in the product backlog items.

4.2 Summary

In this chapter, this research constructs the proposed solution’s model building in details, where explanation about Risk Poker technique is described, followed by combining test coverage with Risk Poker to provide test coverage estimation, design and develop system prototype of Risk Graph for the proposed testing model and lastly describes how the proposed testing model fits inside a Scrum methodology.

This chapter achieves this research’s objective, which is Objective 3 in order to construct a testing model for agile project following Scrum. Construction of model building for the proposed testing model is described in details in this chapter. To validate whether the proposed testing model described in this chapter would perform effectively, experiment validation is carried out as defined in the following chapter, which is chapter 5.

Universiti Malaya

CHAPTER 5: VALIDATION

In this chapter, validation of the proposed testing model is executed through an experiment on control group of student teams compared to the experimental group of student teams. The result of the experiment from both control group and experimental group student teams are analyzed statistically using SPSS tool to answer research questions defined in objective 4 for this research. The experiment details are described in the following section; such as experiment design which is described in section 5.1 with subsection of the experiment objective, experiment participants, experiment materials and experiment process. At the end of the experiment, the collected data and result are analyzed in experiment results in section 5.2. Lastly, section 5.3 discusses the study validity of this research for future reference.

5.1 Experiment Design

In this section, the experiment is designed thoroughly to make sure all elements are considered and available for experiment process. At this stage, experiment objectives is listed to make sure the research questions required to achieve objective 4 is addressed during the implementation of the experiment as explained in section 5.1.1. Next, participants of the experiment validation are identified and their characteristics are listed, as described in section 5.1.2. Next, experiment materials are prepared for this research's experiment validation as described in section 5.1.3. The materials prepared should fit student teams' knowledge and their ability to execute the experiment successfully to make sure the experiment is deliverable. Lastly, the experiment process or step by step of process flow of the experiment is planned as described in section 5.1.4. Details of the experiment designed to validate the proposed testing model are explained in the subsections below.

Universiti Malaya

5.1.1 Experiment Objective

In order to fill the gap that has been defined in Chapter 1 and to confirm whether the proposed testing model could perform in Scrum methodology, this research has taken an approach to implement the proposed testing model in an agile software development project following Scrum for a group of student team.

The study was conducted to observe how well the proposed method performed as a software test strategy for Scrum student team. The validation of experiment result is aimed to answer research questions defined in objective 4 for this research as described in Chapter 1. There are two research questions to be answered to achieve objective 4, which is RQ4.1 and RQ4.2 which were explained in detail in the experiment results section 5.2 in this chapter.

5.1.2 Experiment Participants

The experiment was conducted to observe how well the proposed method perform as a software test strategy for the student team in an agile software development project following Scrum. The conducted experiment requires 3 experimental group of students to estimate risk level and test coverage using the proposed method while the other 3 control group students were using the averaged statistical combination of individual estimates for further comparison. Data and result collected in the study are used to analyze the student group performance when using the proposed method compared to the averaged statistical combination of individual estimates. The student groups are made of final year undergraduate students of Software Engineering course. They have completed the Software Verification & Validation study syllabus for the semester and assumed to be familiar with test planning process, able to construct test cases for testing purpose and able to perform various types of testing technique throughout the software project. Table 5.1 listed the summary of the experiment participants' details.

Universiti Malaya

Table 5.1: Summary of experiment participants' details

Participants	
Scrum Team	Final year student of Software Engineering course
	Completed the Software Verification & Validation syllabus
	Assumed to be familiar with: 1) Test planning 2) Construct Test Cases 3) Execute testing
	3 Experimental group (4 students each group)
	3 Control group (4 students each group)

5.1.3 Experiment Materials

A set of 34 user stories were given to 6 groups of students to be analyzed, estimated and tested for an agile software project lifecycle following Scrum. Each team is required to prioritize and estimate test coverage for the same set of 34 user stories within 3 sprints with each sprint's duration lasting for 2 weeks. The whole project takes 9 weeks to complete the estimation and testing. Each group consists of 4 students and acts as a self-organizing and self-managing Scrum Team, responsible to analyze risks, risk level, estimate test coverage and execute testing on an e-commerce system based on the given user stories.

The software project prepared by this research is based on an open source e-commerce system for a client named Marvel Beads. The client provides the required user requirements and this research plays the role of Product Owner in collecting user story from client. An open source e-commerce system is developed and customized according to the user stories and passed to the student team for testing for each sprint.

Universiti Malaya

Each story consists of a short description of the required functionality to be discussed by team members during planning meeting to analyze risks, prioritize the tasks and assign how much test coverage is needed. Testing activity is executed in a two-week sprint based on the test coverage estimation obtained during planning meeting. The six groups of student teams are divided into two categories; 1) Three of the groups apply risk-poker technique to prioritize tasks and estimate test coverage, whilst the other 2) Three teams use averaged statistical combination of individual estimations to prioritize and estimate test coverage technique.

Seeded faults are planted in the system to suit the purpose of testing in order to measure how much fault is exposed at the end of the project to measure test coverage adequacy estimated using the proposed model. Table 5.2 listed the summary of experiment materials and environment prepared to execute the experiment validation for the proposed testing model.

Universiti Malaya

Table 5.2: Summary of experiment materials

Experiment Materials	
Software Project	1 complete software testing project (test plan, construct test cases, execute testing, report) Tamper coding for testing
	34 user stories for 3 sprints
	Sprint duration: 2 weeks
	Project duration: 9 weeks
Scrum process affected	Sprint Planning Meeting
	Sprint backlog prioritization
	Testing activity

Project Data	Data For Sprint Planning Meeting: 1) User stories 2) Risks identification list 3) Risks level rating 4) Test coverage

5.1.4 Experiment Process

Once the software project environment is ready for the experiment, this research starts the experiment according to the step-by-step process described in Table 5.3.

Table 5.3: Experiment Steps

Steps	**Sprint**	**Activities**
A	0	Brief and train student teams on the software project details and Scrum process
A.1		Train experimental student teams on how to implement the proposed method in Scrum
A.2		Train control student teams on how to estimate risks and test coverage using averaged statistical combination of individual estimations
B		Product Owner list user stories in the product backlog for 3 sprints respectively
C	1,2 and 3	Sprint Planning Meeting
C.1		Student team and Product Owner conduct a discussion session on the user stories for Sprint 1
C.2		Tasks are identified, associated risks are identified, risks are discussed and analyzed
C.3		Student teams estimate risks level and test coverage for user stories
C.3.1a		Experimental student teams use Risk Poker technique to estimate risks level and test coverage
C.3.1b		Control group student teams use the averaged statistical combination of individual scores to estimate risks level and test coverage
C.4		A list of risks level and test coverage estimation is collected for the user stories
C.5		Student teams insert the rating into Risk Graph prototype system to prioritize the highest risks level tasks to the lowest risks level

Universiti Malaya

Steps	Sprint	Activities
D	1,2 and 3	Sprint Test the prioritized items according to the assigned test coverage in the Risk Graph to expose fault
E		Report the list of fault exposed during testing activity
F		The researcher collects data and test result for both experimental and control group.

The detailed explanation of the step-by-step experiment process is as follows:

1) This research conducted a briefing session (Step A; Table 5.3) to train student teams on Scrum process. Briefed experimental student teams on how to implement the proposed method within Scrum. Next, student teams of the control group are briefed and trained to estimate risk level and test coverage using averaged statistical combination of individual estimates.
2) At the beginning of the project, a collection of the same user stories have been assigned for 3 sprints respectively for all teams (Step B).
3) Student teams start the project with Sprint Planning Meeting. In Sprint Planning Meeting, the student teams are required to discuss user stories, identify risks and analyze risks associated with the user stories for prioritization later (Step C). Once discussion has taken place and everyone is clear with the related issues for the user story, student teams are provided with rating card to rate risk level for the discussed user story. The risk level assigned is associated with related test coverage for each level. The risk rating card consists of four color risks: Red as the highest risk, followed by orange, yellow and green. The highest risk level is associated with the most intense code coverage for unit test which is multiple condition coverage, followed by condition determination coverage, decision condition coverage and decision coverage.

Universiti Malaya

- The experimental student group is required to implement Risk Poker technique in Sprint Planning Meeting to estimate test coverage and prioritize risks, while on the other hand,
- The control student group estimate test coverage and prioritize risks using averaged statistical combination of individual estimations where each team member is required to prioritize the user stories individually and the scores are then averaged to get the test coverage and prioritization scores. The rating card for control group students has scores where the highest risk scores 4 points, followed by high risk with 3 points, medium risk with 2 points and low risk with 1 point.
- Experimental student group estimated the risk level individually on the rating card and then present the rating result together with other team member to reveal rating result. If there is difference in color of rating, they will discuss the color difference and issues related. And then, once again they will estimate the risk level rating individually and present the result once again to achieve group consensus on risk rating. Should the rating color is difference again at this time around, the team use the highest risk level rating. The rating is updated in Risk Graph prototype system to prioritize and assign test coverage of the user story.
- On the other hand, the rating card for control group student teams contain score points for each color risk level to be averaged to get the statistical combination of individual estimates. Control student estimated the risk level individually on the rating card and then present the rating result together with other team member to reveal result rating. The score of the rating will be accumulated amongst team member and then averaged to get the risk level score. The user story is prioritized and assigned with appropriate test

Universiti Malaya

coverage according to the averaged score using the Risk Graph prototype system.

4) At the end of Sprint Planning Meeting, student teams obtained prioritized tasks for the Sprint Backlog using the Risk Graph prototype system.
5) Following that, upon starting sprint, student team start testing the user stories according to the test coverage assigned to the highest risk product first, followed by medium risk and ended with low risk product (Step D) as shown in the Risk Graph prototype system.
6) At the end of sprint, student teams provide a list of fault exposed as well as the test result (Step E).

Data is collected to measure how Risk Poker and test coverage implementation in Scrum performed compared to the averaged statistical combination of individual estimates (Step F).

5.2 Experiment Results

The result of experiment on estimating risk level and test coverage for 34 user stories is collected and analyzed to validate the proposed method compared to the averaged statistical combination of individual estimations. This research monitors and observes throughout the experiment process, and data is collected throughout the experiment and test result is collected at the end of each sprint. The test result is a report of how much fault is exposed throughout testing process in each sprint for each team. The exposed fault is compared to the seeded fault to measure test coverage adequacy. Basic descriptive statistics for the student teams test result are presented in Table 5.4.

Universiti Malaya

Table 5.4: Statistics of Student Teams' Test Result

	BRE averaged Individual Statistical Combination	BRE Risk Poker
User stories	102	102
Mean	0.5049	0.2402
Median	0.0000	0.0000
Std. deviation	0.88858	0.49116
Skewness	2.543	1.973
Std. error of skewness	0.239	0.239
Kurtosis	8.145	3.235
Std. error of kurtosis	0.474	0.474
Range	5.00	2.00

Universiti Malaya

5.2.1 RQ4.1: Is the test coverage provided by Risk Poker-based proposed model adequate compared to the statistical combination of individuals?

Exposed fault is used to measure whether the test coverage estimated in the sprint is adequate to expose the seeded fault in the system. Balanced Relative Error (*BRE*) is used to calculate performance of test coverage assigned in the sprint. Thus, the greater the *BRE* score is, the less adequate test coverage assignment performed in the related sprint as the *BRE* score represents how accurate test coverage estimated is to expose seeded fault. The *BRE* of both experimental student teams and control student teams are calculated as follows:

$$BRE = \frac{|seeded_{fault} - exposed_{fault}|}{\min\,(seeded_{fault}, exposed_{fault})}$$

In order to answer RQ4.1 this research calculates the mean *BRE* of the fault exposure by comparing the *BRE* of Risk Poker estimates (experimental student teams) and the *BRE* of the averaged statistical combination of individual estimates (controlled student teams).

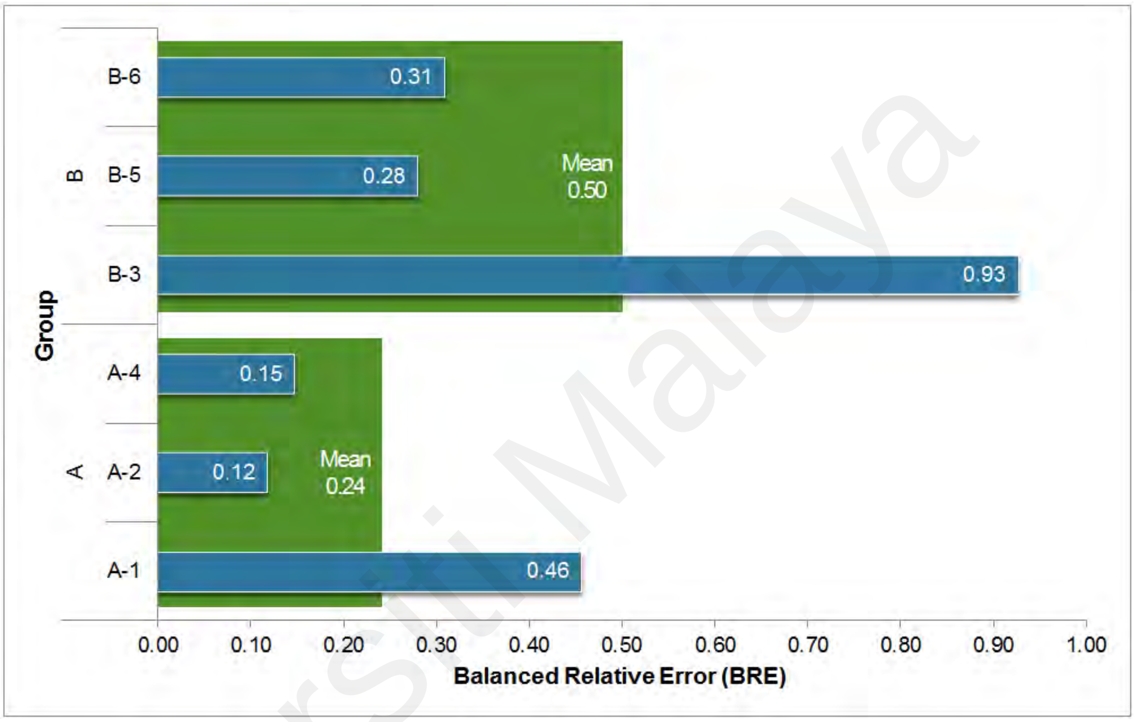


Figure 5.1: BRE mean for experimental and control group

Figure 5.1 shows the *BRE* mean for both the experimental and control student teams test result. The mean *BRE* of the seeded fault is 0.00 thus, the closer the mean *BRE* of the test result to 0.00, the lesser the relative error of the fault exposure. Table 5.5 summarized the *BRE* scores for both experimental group (student team A-1, A-2, A-4) and controlled group (student team B-3, B-5, B-6). It seems that experimental group student teams returned *BRE* scores are within 0.0 - 1.0, and the greatest relative error score for this group is within 1.1 - 2.0 (student team A-1). Whilst on the other hand, *BRE* scores for controlled group student teams are within 1.1 - 2.0 applicable for all participated teams (student team B-3, B-5, B-6). Straightforward analysis of the results

Universiti Malaya

has suggested that mean *BRE* of experimental student teams (0.24) is smaller than the mean *BRE* of the controlled student teams (0.50). Thus this research is able to conclude that the test coverage estimation provided by Risk Poker technique is more adequate in exposing the seeded fault compared to the averaged statistical combination of individual estimates.

Table 5.5: BRE scores

	BRE =					
	0.0	**0.1 - 1.0**	**1.1 - 2.0**	**2.1 - 3.0**	**3.1 - 4.0**	**4.1 - 5.0**
n = 34						
A-1	21	10	3	0	0	0
A-2	30	4	0	0	0	0
A-4	29	5	0	0	0	0
B-3	17	9	4	2	1	1
B-5	25	8	1	0	0	0
B-6	23	9	2	0	0	0

5.2.2 RQ4.2: How does Risk Poker-based proposed model estimation differ from the averaged statistical combination of individual estimations?

Referring to the descriptive statistic as stated in the previous Table 5.4, for the total of 102 user stories that were analyzed for both experimental and control group of student teams, the experimental group of student teams which estimate risk and test coverage using Risk Poker technique, have the *BRE* mean of 0.24 (*sd* = 0.49) compared to the control group of student teams BRE mean which is 0.50 (*sd* = 0.89). So, does the difference between the two *BRE* means is simply due to sampling variation, or does the *BRE* provide evidence that Risk Poker technique does, on average, improve test coverage estimation? The *p-value* obtained from an independent samples t-test answers this question. This research has run an independent t-test to test the hypothesis that both

Universiti Malaya

the experimental and control group were associated with statistically significantly different mean of balanced relative error of the test coverage estimation. Thus, the independent t-test was conducted to compare balanced relative error for test coverage estimation in using Risk Poker technique as risk and test coverage estimation and in averaged statistical combination of individual estimation conditions.

The result displayed in Table 5.6 has shown that there was a significant difference in the *BRE* scores for student teams that used Risk Poker technique to estimate risk level and test coverage (M=0.24, SD=0.49) and *BRE* scores for student team that did not used Risk Poker technique (M=0.50, SD=0.89) conditions; t(202)=(2.63), p=(0.009). Since the *p-value* is 0.009, therefore the difference between the two means is statistically significantly different from zero at the 5% level of significance. Thus, there is sufficient evidence to suggest that risk poker technique does change the mean *BRE* of test coverage accuracy. There is an estimated change of standard error of 0.1%. Hence, these results suggest that Risk Poker technique really does have an effect on estimating risk level and test coverage for testing of an agile project following Scrum.

Universiti Malaya

Table 5.6: Independent t-test result

		Lavene's Test for Equality of Variances		**t-test for Equality of Means**					**95% Confidence Interval of the Difference**	
		F	**Sig.**	**t**	**df**	**Sig. (2-tailed)**	**Mean Difference**	**Std. Error Difference**	**Lower**	**Upper**
BRE Scores	**Equal variances assumed**	17.139	0.000	-2.633	202	0.009	-0.26471	0.10053	-0.4629	-0.6649
	Equal variances not assumed			-2.633	157.448	0.009	-0.26471	0.10053	-0.4632	-0.6615

5.3 Discussion

RQ4.1 *Is the test coverage provided by Risk Poker-based proposed model adequate compared to the statistical combination of individuals?*

Referring back to Figure 5.1, this research has learnt that *BRE* mean for control group (Group B-3, B-5 & B-6) is greater than experimental group, thus it indicates that the control group’s test coverage estimation is less accurate to expose seeded fault. The range of control group’s mean is 5.00 compared to experimental group which is 2.00 as shown earlier in Table 5.4 Statistics. The higher range of unexposed seeded fault for control group indicates that test coverage estimation by experimental group is more adequate to detect seeded fault compared to the control group. In addition to that, the highest number of unexposed seeded fault (7 unexposed seeded fault) occurred in two out of three control group’s estimation as shown in Figure 5.2 also indicates inadequate test coverage estimation technique for testing to cover required functionality to detect fault. These issues have proven that Risk Poker technique is able to provide relevant estimation of test coverage when the group is allowed to discuss their rating and concerns, where hidden issues are able to be highlighted for the task rating and estimation. Furthermore, this result has shown significant difference in statistical tests presented previously in Table 5.4 which indicates that Risk Poker estimates provided by student teams is more accurate than the averaged statistical combination of individual estimations.

Universiti Malaya

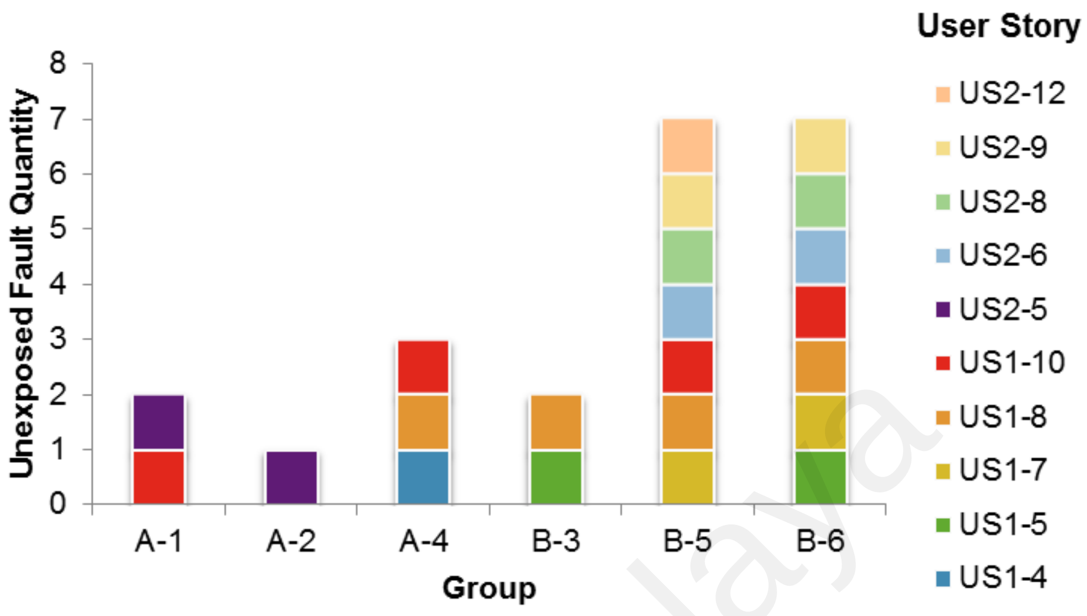


Figure 5.2: Unexposed fault quantity

RQ4.2: How does Risk Poker-based proposed model estimation differ from the averaged statistical combination of individual estimations?

Following, an independent t-test analysis conducted on the experiment test result has shown that there is a statistically significant difference between the proposed technique and the averaged statistical combination of individual estimations. Thus, it indicates that test coverage estimation provided by student teams that used Risk Poker technique is more accurate than the averaged statistical combination of individual estimations. In addition to that, the result of this statistical tests also disagree with (Armstrong, 2006) that face-to-face meetings are harmful for decision making. Considering the fact that Risk Poker estimates tended to be slightly better than the averaged statistical combination of individual estimates, it seems reasonable to continue the research on group processes in software estimation.

Universiti Malaya

5.4 Study Validity

The conclusions of this research are based on the results of statistical tests which has exposed that there is a statistically significant difference between experimental group and control group student teams. In this study, in order to increase study validity, 6 teams studied were working on the same problem. All teams estimated the same set of user stories; therefore, their estimates are directly comparable. However, in spite of the fact that the study was conducted within the framework of a group student project, every effort was made to increase its external validity by simulating an industrial environment as closely as possible. The experiment is controlled where;

i. The Sprint Planning Meeting is conducted within a certain time-frame with Product Owner and Customer is around to explain the user stories to the team.
ii. The user stories were defined on the basis of the e-commerce system that is actually used for an online store and the students were required to fully test the code with seeded fault.
iii. The test report are required to be handed to the researcher at the end of each sprint to ensure testing are executed as planned.

Nevertheless, the main threat to external validity remains that only one project was used.

On the basis of statistical analysis, the results can be generalized only to students of the last year of computer science course working on similar projects that require the testing of e-commerce systems, while more studies are needed on real projects of different size and complexity in order to generalize the findings to industry. In addition to that, the researcher is also the Product Owner, so it might not be comparable to an

Universiti Malaya

actual product owner in industrial environment. Furthermore, the experiment environment does not include full development project and bug fixes which could be another variable that would contribute to the effectiveness of implementing the proposed method in an agile project following Scrum.

Student teams were required to end their Sprint Planning Meeting within 60 - 80 minutes in each meeting where they discuss, analyze, prioritize and estimate test coverage for each user story. The experiment includes the time constraint in the project execution to make sure both the experimental and control group spend the same amount of effort and time to achieve decision on risk analyses and estimating test coverage.

Considering the aforementioned limitations, the results of this study can be used together with other studies as a stepping stone to further research, narrowing down the focus to a more experienced expert groups and searching for contexts where Risk Poker improves test coverage estimation accuracy by increasing commitment, sharing estimating expertise, promoting team growth and refining solution understanding. To second that, experimenting with student teams alone might not give a various statistic result to compare and measure the difference of implementing the proposed technique with other technique. Also, in a bigger scale, student would not be able to replicate a real situation as professional testers with their limited experience, thus the need for a bigger scale study to involve industrial project for more statistical comparable result. And lastly, the experiment is a comparison between experimental and control group of people, not a post and after post technique. Thus, a study reporting the statistic result of improvement of accuracy of before implementing the proposed technique with after the implementation the proposed technique would help.

Universiti Malaya

5.5 Summary

This chapter provides discussion and result of the fourth research objective, which is to validate the proposed model in a software project case study following Scrum in order to improve risk analyses in agile projects. Research questions 4.1 and 4.2 are answered respectively throughout this chapter to provide test result and data to measure the effectiveness of proposed method in the experiment conducted. The test result shows that there is a significant difference between the experimental group and controlled group to support that the proposed method would provide positive effect to the process, however, the study scale is small, which involves final project of student teams. Overall, the experimental group which implemented the proposed method performs better than the control group because the result analysis shows that Risk Poker test coverage estimation is able to expose seeded fault better than the controlled group, thus it is able to improve risk analysis in agile project following scrum. However, future case study or future researcher should consider all the study validity factors as listed in section 5.4 for better result and reliable measurement.

Universiti Malaya

CHAPTER 6: CONCLUSION

In this chapter, this research discusses in brief the conclusion of the research, research findings, research contributions and research limitations to give a clear head-point to any interested industry personnel, researcher and academician who would like to implement the proposed method in software development project. Also, this research summarizes potential future work and provides some recommendations for the use of other researchers who are interested to explore further on software testing and agile domain.

6.1 Fulfillment of Research Objectives

At the end of this research, all objectives defined previously in Chapter 1 are achieved, which are;

- Through intensive literature reviews, this research has identified a suitable testing strategy that could fit agile projects following Scrum effortlessly and able to performed effectively through the experiment validation described in Chapter 5,
- This research has also identified suitable test coverage technique to combine with the identified software test strategy, where the test coverage estimation provided by the proposed method shows significant difference in the statistical result analyzed in Chapter 5 compared to the individual estimations,
- This research has successfully constructed a testing model that would fit effortlessly in agile project, where the testing model is successfully implemented in the experiment process, experiment environment and materials section as described in Chapter 5,

Universiti Malaya

- Lastly, this research has successfully validated that the proposed testing model is able to provide better estimation of test coverage in agile project by answering the required research questions which resulted in the significant difference of the result analysis for the experiment result section, as described in Chapter 5.

6.2 Research Contributions

Based on the findings described above, this research helps to identify a risk analysis technique as software test strategy whereby Risk Poker, which strongly emphasizes on group discussion characteristics of agile method, is suitably integrated in the planning meeting and consequently yielded better risk prioritization as well as estimating adequate test coverage.

This benefits industry players who would like to implement ready-to-use software testing strategy for an agile project following scrum, in which they will be able to efficiently prioritize user stories and estimate the required testing effort and testing coverage.

Moreover, the results identified in this research will provide some guidance for practical practitioners to understand what to expect when trying to implement this technique in software development projects and helps other interested researcher to explore more on software testing in agile domain.

Universiti Malaya

6.3 Research Limitations

Although in general this research is able to meet the stipulated research objectives, there are several limitations to the applicability of the results. One of the limitations is that the case study carried out in this research may not speak for all levels of project scale due to the limited size scale of the experiment conducted. Besides, this research also did not have the suitable resources for experiment participants such as those from bigger scale projects in the software industry to implement and simulate this study in an almost realistic software project environment with expert personnel from various fields to come together and contribute their views and estimations which can serve as another method to measure the accuracy and efficiency of the proposed method. Like those in most researches of this level, this research also faced other common limitations such as time, people, money and real project environment. Other limitations of the experiment validation for this research is also described in chapter 5, section 5.4 for Study Validity.

6.4 Recommendation for Future Work

This research opens up opportunities to various potential future works and some of the future works highly recommended by this research are as follow;

- Integration of Risk Poker with Planning Poker in the planning meeting since both techniques share many similar characteristics to achieve group consensus in decision making process.

- Conduct similar case study but in a software project which has various levels of group members knowledge and field of expertise which is expected to produce higher accuracy in risk prioritization and test coverage estimation.

Universiti Malaya

- Study the outcome of a software development project that initially does not apply the method proposed by this research, and then apply the proposed method in order to identify and measure the improvements brought by the proposed method.
- Apply similar case study on a real software development project involving real industry personnel in order to verify the practicality of implementing the proposed method in the industry. In the same study, researchers may also identify the reception level and issues that would occur upon implementing the proposed technique to the existing team members who have established their own ways of estimating prior to the introduction of the new method.

Universiti Malaya

REFERENCES

Amland, S. (2000). Risk-based testing:: Risk analysis fundamentals and metrics for software testing including a financial application case study. *Journal of Systems and Software, 53*(3), 287-295.

Armstrong, J. S. (2006). How to make better forecasts and decisions: Avoid face-to-face meetings. *Foresight, 5,* 3-8.

Bannerman, P. L. (2008). Risk and risk management in software projects: A reassessment. *Journal of Systems and Software, 81*(12), 2118-2133.

Biju, S. M. (2010). Agile software development methods and its advantages *Technological Developments in Networking, Education and Automation* (pp. 603-607): Springer.

Black, R. (2003). Quality Risk Analysis. *USA: Rex Black Consulting Services[Online] Available: http://www/. rexblackconsulting. com/publications/Quality% 20Risk% 20A nalysis1. pdf.*

Boehm, B. (1989). *Software risk management.* Paper presented at the European Software Engineering Conference.

Boehm, B. (2002). Get ready for agile methods, with care. *Computer, 35*(1), 64-69.

Boehm, B. W. (1991). Software risk management: principles and practices. *Software, IEEE, 8*(1), 32-41.

Boness, K., Finkelstein, A., & Harrison, R. (2008). A lightweight technique for assessing risks in requirements analysis. *IET software, 2*(1), 46-57.

Briand, L., & Pfahl, D. (1999). *Using simulation for assessing the real impact of test coverage on defect coverage.* Paper presented at the Software Reliability Engineering, 1999. Proceedings. 10th International Symposium on.

Caballero, E., Calvo-Manzano, J. A., & San Feliu, T. (2011). Introducing Scrum in a Very Small Enterprise: A Productivity and Quality Analysis *Systems, Software and Service Process Improvement* (pp. 215-224): Springer.

Cai, X., & Lyu, M. R. (2007). *Software reliability modeling with test coverage: Experimentation and measurement with a fault-tolerant software project.* Paper presented at the Software Reliability, 2007. ISSRE'07. The 18th IEEE International Symposium on.

Chilenski, J. J., & Miller, S. P. (1994). APPLICABILITY OF MODIFIED CONDITION DECISION COVERAGE TO SOFTWARE TESTING. *Software Engineering Journal, 9*(5), 193-200.

Cho, J. (2008). Issues and Challenges of agile software development with SCRUM. *Issues in Information Systems, 9*(2), 188-195.

Universiti Malaya

Coffin, R., & Lane, D. (2006). A Practical Guide to Seven Agile Methodologies. *Part 1, XP, Scrum, Lean, and FDD.*

Collins, E., Dias-Neto, A., & de Lucena, V. (2012). *Strategies for agile software testing automation: An industrial experience.* Paper presented at the Computer Software and Applications Conference Workshops (COMPSACW), 2012 IEEE 36th Annual.

Conboy, K., & Coyle, S. (2009). A case study of risk management in agile systems development.

Crispin, L., & Gregory, J. (2009). *Agile testing: A practical guide for testers and agile teams*: Pearson Education.

Dang, T., & Nahhal, T. (2009). Coverage-guided test generation for continuous and hybrid systems. *Formal Methods in System Design, 34*(2), 183-213.

Deemer, P., Benefield, G., Larman, C., & Vodde, B. (2010). The scrum primer. *Scrum Primer is an in-depth introduction to the theory and practice of Scrum, albeit primarily from a software development perspective, available at:* http://assets/. *scrumtraininginstitute. com/downloads/1/scrumprimer121. pdf, 1285931497*, 15.

Eloranta, V.-P., Koskimies, K., Mikkonen, T., & Vuorinen, J. (2013). *Scrum Anti-Patterns--An Empirical Study.* Paper presented at the Software Engineering Conference (APSEC, 2013 20th Asia-Pacific.

Felker, C., Slamova, R., & Davis, J. (2012). *Integrating UX with scrum in an undergraduate software development project.* Paper presented at the Proceedings of the 43rd ACM technical symposium on Computer Science Education.

Fowler, M., & Highsmith, J. (2001). The agile manifesto. *Software Development, 9*(8), 28-35.

Gargantini, A., & Riccobene, E. (2001). ASM-based testing: Coverage criteria and automatic test sequence generation. *Journal of Universal Computer Science, 7*(11), 1050-1067.

Garousi, V., & Zhi, J. (2013). A survey of software testing practices in Canada. *Journal of Systems and Software, 86*(5), 1354-1376.

Gittens, M., Romanufa, K., Godwin, D., & Racicot, J. (2006). *All code coverage is not created equal: a case study in prioritized code coverage.* Paper presented at the Proceedings of the 2006 conference of the Center for Advanced Studies on Collaborative research.

Grenning, J. (2002). Planning poker or how to avoid analysis paralysis while release planning. *Hawthorn Woods: Renaissance Software Consulting, 3.*

Hall, E. M. (1998). *Managing risk: Methods for software systems development*: Pearson Education.

Universiti Malaya

Harichandan, S., Panda, N., & Acharya, A. A. (2014). Scrum Testing With Backlog Management in Agile Development Environment.

Hartman, B., & Lawrence, R. (2013). Intro to Agile. *Agile For All.* Retrieved from http://www.agileforall.com/

Hartmann, J., Fontoura, L. M., & Price, R. T. (2005). Using Risk Analysis and Patterns to Tailor Software Processes. *XIX Simpósio Brasileiro de Engenharia de Software, Uberlândia*.

Haugen, N. C. (2006). *An empirical study of using planning poker for user story estimation.* Paper presented at the Agile Conference, 2006.

Hellmann, T. D., Sharma, A., Ferreira, J., & Maurer, F. (2012). *Agile Testing: Past, Present, and Future--Charting a Systematic Map of Testing in Agile Software Development.* Paper presented at the Agile Conference (AGILE), 2012.

Hossain, E., Babar, M. A., & Paik, H.-y. (2009). *Using scrum in global software development: a systematic literature review.* Paper presented at the Global Software Engineering, 2009. ICGSE 2009. Fourth IEEE International Conference on.

Hu, Z.-g., Yuan, Q., & Zhang, X. (2009, 11-12 July 2009). *Research on Agile Project Management with Scrum Method.* Paper presented at the Services Science, Management and Engineering, 2009. SSME '09. IITA International Conference on.

Itkonen, J., Rautiainen, K., & Lassenius, C. (2005). *Towards understanding quality assurance in agile software development.* Paper presented at the Icam 2005.

James, M. (2007, 24 September 2007). Scrum and Quality Assurance. *Agile.*

Jogu, K. K., & Reddy, K. N. (2013). Moving Towards Agile Testing Strategies. *Cvr journal of science & technology*, 88.

Julius, A. A., Fainekos, G. E., Anand, M., Lee, I., & Pappas, G. J. (2007). Robust test generation and coverage for hybrid systems *Hybrid Systems: Computation and Control* (pp. 329-342): Springer.

Karhunen, J.-R. (2009). Scrum quality management: an empirical study.

Karlsson, E., & Martensson, F. (2009). *Test processes for a Scrum team.* Master's thesis, Lund University, Sweden.

Kasurinen, J., Taipale, O., & Smolander, K. (2010). *Test case selection and prioritization: risk-based or design-based?* Paper presented at the Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, Bolzano-Bozen, Italy.

Kayes, I., Sarker, M., & Chakareski, J. (2013). On Measuring Test Quality in Scrum: An Empirical Study. *arXiv preprint arXiv:1310.2545*.

Universiti Malaya

Kettunen, V., Kasurinen, J., Taipale, O., & Smolander, K. (2010). *A study on agility and testing processes in software organizations.* Paper presented at the Proceedings of the 19th international symposium on Software testing and analysis.

Khalane, T., & Tanner, M. (2013). *Software quality assurance in Scrum: The need for concrete guidance on SQA strategies in meeting user expectations.* Paper presented at the Adaptive Science and Technology (ICAST), 2013 International Conference on.

Lawrence, J., Clarke, S., Burnett, M., & Rothermel, G. (2005). *How well do professional developers test with code coverage visualizations? An empirical study.* Paper presented at the Visual Languages and Human-Centric Computing, 2005 IEEE Symposium on.

Lewis, J., & Neher, K. (2007). *Over the Waterfall in a Barrel-MSIT Adventures in Scrum.* Paper presented at the AGILE.

Li, J., Moe, N. B., & Dybå, T. (2010). *Transition from a plan-driven process to scrum: a longitudinal case study on software quality.* Paper presented at the Proceedings of the 2010 ACM-IEEE international symposium on empirical software engineering and measurement.

Li, M., Huang, M., Shu, F., & Li, J. (2006). *A risk-driven method for eXtreme programming release planning.* Paper presented at the Proceedings of the 28th international conference on Software engineering.

Löffler, R., Güldali, B., & Geisen, S. (2010). Towards Model-based Acceptance Testing for Scrum. *Softwaretechnik-Trends, GI.*

Mahnic, V. (2011). A case study on Agile Estimating and Planning using Scrum. *Electronics and electrical engineering, 111*(5), 123-128.

Mahnič, V., & Hovelja, T. (2012). On using planning poker for estimating user stories. *Journal of Systems and Software, 85*(9), 2086-2095. doi:http://dx.doi.org/10.1016/j.jss.2012.04.005

Marré, M., & Bertolino, A. (2003). Using spanning sets for coverage testing. *Software Engineering, IEEE Transactions on, 29*(11), 974-984.

Moe, N. B., & Dingsøyr, T. (2008). Scrum and team effectiveness: Theory and practice *Agile Processes in Software Engineering and Extreme Programming* (pp. 11-20): Springer.

Moe, N. B., Dingsøyr, T., & Dybå, T. (2010). A teamwork model for understanding an agile team: A case study of a Scrum project. *Information and Software Technology, 52*(5), 480-491.

Moløkken, K., & Jørgensen, M. (2004). Expert estimation of the effort of webdevelopment projects: Why are software professionals in technical roles more optimistic than those in non-technical roles. *Journal of Empirical Software Engineering*.

Universiti Malaya

Molokken-Ostvold, K., & Haugen, N. C. (2007). *Combining estimates with planning poker--an empirical study*. Paper presented at the Software Engineering Conference, 2007. ASWEC 2007. 18th Australian.

Moløkken-Østvold, K., Haugen, N. C., & Benestad, H. C. (2008). Using planning poker for combining expert estimates in software projects. *Journal of Systems and Software, 81*(12), 2106-2117.

Moløkken-Østvold, K., & Jørgensen, M. (2004). Group Processes in Software Effort Estimation. *Empirical Software Engineering, 9*(4), 315-334. doi:10.1023/B:EMSE.0000039882.39206.5a

Moore, R., Reff, K., Graham, J., & Hackerson, B. (2007). *Scrum at a fortune 500 manufacturing company*. Paper presented at the Agile Conference (AGILE), 2007.

Nelson, C. R., Taran, G., & de Lascurain Hinojosa, L. (2008). Explicit risk management in agile processes *Agile processes in software engineering and extreme programming* (pp. 190-201): Springer.

Noor, R., & Khan, M. F. (2014). Defect Management in Agile Software Development. *International Journal of Modern Education and Computer Science, 6*(3), 55.

Nyfjord, J. (2008). Towards integrating agile development and risk management.

One, V. (2010). The State of Agile Development. *State of Agile Survey 2010.*

Paulk, M. C. (2002). Agile methodologies and process discipline. *Institute for Software Research*, 3.

Petersen, K., & Wohlin, C. (2010). The effect of moving from a plan-driven to an incremental software development approach with agile practices. *Empirical Software Engineering, 15*(6), 654-693.

Schatz, B., & Abdelshafi, I. (2005). Primavera gets agile: a successful transition to agile development. *Software, IEEE, 22*(3), 36-42. doi:10.1109/MS.2005.74

Schwaber, K. (1997). Scrum development process *Business Object Design and Implementation* (pp. 117-134): Springer.

Schwaber, K. (2004). *Agile Project Management with Scrum*: Microsoft Press.

Schwaber, K., & Beedle, M. (2002). Agilè Software Development with Scrum.

Schwaber, K., & Sutherland, J. (2007). What is scrum. *URL:* http://www.scrumalliance.org/system/resource/file/275/whatIsScrum.pdf, *[Stand: 03.03. 2008]*.

Selvi, K., & Majumdar, R. (2013). Scrum: An Agile Process. *International Journal of Research in Engineering and Technology, 2*(3), 337-340.

Universiti Malaya

Shahid, M., Ibrahim, S., & Selamat, H. (2011). *An Evaluation of Current Approaches to Support Test Coverage Analysis.* Paper presented at the International Conference on Computer Engineering and Technology, 3rd (ICCET 2011).

Stallbaum, H., Metzger, A., & Pohl, K. (2008). *An automated technique for risk-based test case generation and prioritization.* Paper presented at the Proceedings of the 3rd international workshop on Automation of software test.

Stolberg, S. (2009, 24-28 Aug. 2009). *Enabling Agile Testing through Continuous Integration.* Paper presented at the Agile Conference, 2009. AGILE '09.

Sutherland, J. (2001). Agile can scale: Inventing and reinventing scrum in five companies. *Cutter IT Journal*, *14*(12), 5-11.

Talby, D., Keren, A., Hazzan, O., & Dubinsky, Y. (2006). Agile software testing in a large-scale project. *Software, IEEE, 23*(4), 30-37.

Thomas Müller, D. F., ISTQB WG Foundation Level. (2011). Certified Tester Foundation Level Syllabus. Version 2011. Retrieved from http://www.istqb.org/downloads/send/2-foundation-level-documents/3-foundation-level-syllabus-2011.html4

Van de Laar, J. (2012). Risk Poker: Risk based testing in agile projects. *Software Quality DayS*, 51.

Walkinshaw, N., Bogdanov, K., Derrick, J., & Paris, J. (2010). Increasing functional coverage by inductive testing: a case study *Testing Software and Systems* (pp. 126-141): Springer.

Williams, L., Gegick, M., & Meneely, A. (2009). Protection Poker: Structuring Software Security Risk Assessment and Knowledge Transfer *Engineering Secure Software and Systems* (pp. 122-134): Springer.

Winter, J., Rönkkö, K., Ahlberg, M., & Hotchkiss, J. (2011). Meeting organisational needs and quality assurance through balancing agile and formal usability testing results *Software Engineering Techniques* (pp. 275-289): Springer.

Woodward, M. R., & Hennell, M. A. (2005). Strategic benefits of software test management: a case study. *Journal of Engineering and Technology Management*, *22*(1), 113-140.

Zhu, H., Hall, P. A., & May, J. H. (1997). Software unit test coverage and adequacy. *Acm computing surveys (csur)*, *29*(4), 366-427.

Universiti Malaya