Comput Syst Sci & Eng (2018) 3: 169–185

International Journal of
Computer Systems
Science & Engineering

A Risk Poker Based Testing Model For Scrum

Siti Noor Hasanah Ghazali[1], Siti Salwah Salim[1]*, Irum Inayat[2], Siti Hafizah Ab Hamid[1]

[1] *Department of Software Engineering, Faculty of Computer Science and Information Technology, University of Malaya, Malaysia*
[2] *Department of Computer Science, FAST-National University of Computer and Emerging Sciences, Pakistan*

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, risk prioritization often relies on individual expert judgment. Therefore, Risk Poker, a lightweight risk-based testing methodology in which risk analysis is performed through group discussion that outperforms the individual analyst's estimation is introduced in agile methods. Keeping in view aforementioned benefits Risk Poker can offer, unfortunately, no study has been conducted to empirically prove its ability to improve the testing process to date. Therefore, this research is aimed at closing this research gap by (i) deploying Risk Poker technique as a risk-based strategy in the agile development lifecycle, and (ii) empirically evaluating improvement of the proposed test process. For this purpose, Risk Poker technique is coupled with test coverage for an innovated testing process in an agile project following Scrum in order to provide adequate test coverage for testing activity. A case study was conducted with 6 teams of undergraduate 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 for risk analysis and estimate test coverage outperformed the averaged statistical estimation of risk analysis for user stories

Keywords: Software Engineering, Agile, Scrum, Risk Analysis, Software Testing, Estimation, Risk Poker

1. INTRODUCTION

Agile has been a very popular choice for the ever-changing software projects in the software industry. It is reported that Scrum is the most followed agile method, as reported in a survey that around 58% of the respondents working on agile projects used Scrum [1]. Therefore, in this research, we explore within the scope of Scrum management process for agile projects.

Agile methods explicitly practice risks-based strategies for prioritizing, estimating and analyzing tasks. However, in most of the situations, prioritization and analysis often rely on individual expert opinions. In spite of this, the most important characteristics of an agile team are that it is self-managed and emphasizes on group discussions or team decision in carrying out software development activities. Therefore, we aim to propose a testing model that makes use of group discussion to achieve consensus in prioritizing and analyzing tasks in Scrum methodology. Our proposed model is based on a technique called Risk Poker derived from one of the popular techniques used to combine expert opinions in order to determine planning estimation, called Planning Poker [2], which is widely being used in the scrum methodology [3, 4]. 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 [5]. Thus, a structured technique that supports agile environment is needed to properly address and control risks identified in agile development.

Risk Poker technique proposed by [6] utilizes characteristics of Planning Poker i.e. group discussion in providing lightweight risk analysis technique. Risk Poker is designed to identify and to analyze risks for user stories (requirements) by achieving group consensus. It is a team-based activity in which decisions are made by achieving an agreement between team members. Thus, by acknowledging that Risk Poker and test coverage will provide adequate testing in agile, this research demonstrates the implementation of Risk Poker in planning meetings in Scrum. Therefore, this concept is utilized in this research study and Risk Poker technique is combined with test coverage to provide estimation on how much testing is adequate. Assuming that the

*Corresponding Author. E-mail: salwa@um.edu.my

specified test coverage estimated by Risk Poker is able to provide adequate testing, which is crucial for small iteration-based agile projects, 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.

Following research questions are answered accordingly throughout this research:

RQ1: Is the test coverage provided by Risk Poker-based proposed model adequate as compared to the statistical combination of individuals?

RQ2: *How does Risk Poker-based proposed model estimation differ from the averaged statistical combination of individual estimations?*

2. LITERATURE REVIEW

2.1 Software Testing

Software testing in a software development lifecycle is meant to expose defects of product development and coding errors. Software testing is an essential activity in the software development lifecycle to determine and improve software quality over time. In overall, 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. International Software Testing Qualification Board (ISTQB) [7] has defined some characteristics of software testing to be adhered to; 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 [7]. Studies reveal that agile projects success rate depends on effective software testing process [8–10].

Unfortunately, software testing is often mistakenly treated as a single activity to be executed after coding to reveal code defects. The truth is that software testing is a process that runs continuously in parallel with other software lifecycle processes. It comprises of planning phase, analysis and design phase, execution phase, exit criteria evaluation phase and ends with reporting. The planning phase is to determine scope, analyze and review test items, assign resources and estimate required effort, to identify test approach to design test case and to execute the test. The most commonly adopted approaches are risk-based strategy, requirement-based strategy, and model-based strategy. The risk-based strategy involves test planning, estimating, and prioritizing tests based on the risk analysis performed using project documents and stakeholder's 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. Lastly, model-based strategy involves building mathematical models of the critical system behaviors and then executing tests to confirm whether the system is working as predicted by the model. While in analysis and design phase, a test item is designed and reviewed before execution. The test item is then tested in execution phase according to the test technique or test strategy defined in the planning phase. Afterward, exit criteria are evaluated based on the test coverage defined in the planning phase and the testing activity is concluded with a test report that contains evaluation on how testing activity is performed and lessons learned for future release.

Software testing can be directly implemented in a traditional software methodology, in the design and requirement phase, where static testing is carried out to review or inspect the design and requirement of the project to prevent early mishaps of design and data flow. Following, dynamic testing such as unit testing, integration testing, and system testing is executed on the code to uncover defect using a set of techniques and test cases. In order to carry out dynamic testing, test leaders define and plan the software test design technique and test coverage for the tester. Once testing activity is executed as planned, defects detected will be fixed by the developers and regression testing is carried out to unmask any hidden defects. Lastly, the code under test is evaluated to make sure that the tested program has met the completion criteria and to decide whether the program is ready to be delivered to customer. However, literature reveals that most of the times agile teams modify existing process, strategies or techniques to fit in agile environment [11]. Likewise, there is not much work done on identification of software test strategies that work best for Scrum. Therefore, following are the research motivations for this paper:

1. The agile environment has budget constraint and follows time-boxed iterations that push team members to prioritize level of testing needed for testing activity and prioritization of testing level is actually a risk-based testing strategy [12],

2. A research by [13] 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 methodologies in which has produced result that most of the agile methodology practices adopted risk-based strategy.

3. Agile process implicitly applies risk-driven strategy when sprints are defined and tasks are assigned [14] 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 [15], 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, analyze technology risks and achieve consensus on certain decisions on their own [16]. The 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.

Figure 1 Risk Management Process.

2.2 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 how important it is to identify risks in software project management and act towards it in order to prevent disasters, cancellation, and frustration [17]. 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 [18] which is illustrated in Figure 1. 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 are analyzed [19]. Once the risk analysis is complete, the risks are prioritized according to their importance. The risk prioritization allows project manager and the team to execute actions described previously starting from the highest risk item first [20]. 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.

Similar to a software project risk management, ISTQB [7] 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 the testing point of view, risk-based testing strategy guides how many testing activities and how much effort to spend based on the risks assessment where; high-risk items will need serious testing compared to the low-risk items [21]. 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 ensured that testers have spent the budget in a well-organized approach [12].

Risk management in this testing strategy requires the 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 characteristics, a study [17] 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 [5] 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, understands and contributes to the risks mitigation strategy and be able to execute it as planned. They also reported in their research that, many agile projects implicitly managed risks which have 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 a risk-based testing strategy that could overcome this issue. In accordance with identifying a suitable risk-based strategy for the proposed testing model, the following section explored on previous research about risk-based strategy techniques for comparisons.

Risk-based testing has proven its practicality in managing and mitigating risks but [22] 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. Moreover, [23] also agrees that in practice, many organizations faced difficulties in integrating risk-based testing into an existing test process. Thus, 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 [24] and following 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 [6] 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.

2.3 Risk Poker Technique

Literature reveals one of the main challenges faced in agile software testing is the difficulty in estimating the testing activities [25]. In addition, developing test strategy and test plan for the agile project is also a problem because task development is given higher priority compared to testing of tasks. Hence, there is a need for risk and priority-based testing to overcome this issue in an agile project [26].

Risk Poker is held in a planning meeting to discuss risks associated with the requirement for the sprint and prioritize the risks for the sprint. The prioritized risks define the intensity of the development and testing required for the sprint. 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 [29].

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 analysis is also used to estimate the required testing effort for the product. The 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.

2.4 Test Coverage

Test coverage could be a good indicator to measure software quality by giving information of coverage adequacy for the system under test, thus making it an important step in software testing process [30]. 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 [31] also quoted that test coverage has become a way to relate how much testing 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 [32].

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 the 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 requirements to be fulfilled or achieved which acts as a stopping rule to mark when it is enough to stop testing [33]. Apart from becoming a stopping rule, test criterion could also be defined to determine the observations that should be made during the testing process [34].

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 a 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 [35].

3. Structural testing coverage measurement works well with incomplete requirements as in the agile environment. This type of testing is also useful in exposing unwanted program code or functionality since such testing inspects program codes; looks for statements not executed by any test cases [36].

Much research focuses 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 [37] for ASM specifications provides a formula called test predicate which is defined for coverage criterion to determine if a particular testing goal is reached when each of the test predicates is true. This formula is good to determine and measure which specification and testing goal are not covered yet and summarize test coverage percentage, however, it is unable to serve as stopping rule to indicate when to stop testing.

On the other hand, [33] has introduced the concept of a spanning set of entities for coverage testing where the generated size-reduced 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 as well as evaluating test-suite thoroughness more effectively.

Nevertheless, all these techniques are not lightweight techniques 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 [34] 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 it could serve as a stopping rule criterion for testing activity closure. 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 [38].

3. PROPOSED TESTING MODEL

3.1 Foundations of The Proposed Testing Model

A detailed literature analysis led us to use risk-based strategy as software test strategy for Scrum. Furthermore, the literature analysis helped us to reach a conclusion that Risk Poker technique should be implemented as a risk-based testing strategy combined with test coverage technique in the proposed testing model. Table 1 summarized the literature finding that became the basis for this work.

3.2 Risk Poker as a Risk-Based Testing Technique

In Risk Poker technique, risks are identified, discussed with the team members, and prioritized through group consensus. In a traditional risk prioritization, risks exposure is calculated to prioritize risks. The formula is as follows: $RE = P \times C$, 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 the impact that will affect the project if the risk happens.

Usually, an expert professional in the project management is responsible for assigning a score or weigh factor to the probability (P) and cost (C) of the identified risk according to his or her

Table 1 Findings from the Literature Review.

Category	Available methods or tools	Selected method or tools	Reason
Software Test Strategies			
	• Risk-based strategy • Requirement-based strategy • Model-based strategy	• Risk-based strategy	• Literature revealed that most of agile projects adopt risk-based approach [13]
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 [6] • Can be adopted into agile environment without modification
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 [34]

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 for rating the probability of risks using colored rating card, which is called likelihood factor in this research. Meanwhile, for the cost factor, product owner and stakeholders are responsible for discussing and estimating 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 (PO) involves stakeholders to formulate a feature list to be developed for the product in a form of user stories. PO translates the user stories into product backlog items. Once all required features are collected and listed in the product backlog, PO discusses the product backlog items with stakeholders to identify risks and costs that will affect the project. Thus, at the end of the discussion, PO and stakeholders decide whether the risk level is appropriate for the listed item that will impact business value, user's needs and project as a whole, which is known as impact factor risk identification.

Rating Likelihood Factor Risks

When product backlog items are ready, a sprint planning meeting is called by the PO to discuss the selected user stories, estimate staffing effort, and assign tasks. Since we are deploying Risk Poker technique in this research, everyone is required to identify risks related to the items or tasks under discussion and discuss them thoroughly from the developer, tester, and user perspec-

tives. Risk Poker technique ensures that the team understands the risks thoroughly and is able to rate and manage them. This risk level assignment is called likelihood factor identification for tasks to be developed and tested in the sprint.

Risk Poker Activity Flow

Figure 2 shows the activity of Risk Poker technique in detail, stated as:

- PO and stakeholders decide on the impact factor of the user stories based on how much the user story would affect the end user, business value, and overall project.
- Following that, the PO presents the user stories to the team members during sprint planning meeting without disclosing the impact factor rating. The team members discuss the user stories presented to them thoroughly from developers and testers perspective until they are satisfied to identify and analyze the associated risks. These risks are classified as likelihood factors for a particular user story. The risks eventually affect the product quality, therefore, the team is required to understand all risks associated with the user story and share their concerns based on their expertise.
- When the team members are satisfied with the discussion on user stories and their associated 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. For this, the team members are provided with a set of cards consisting of a table of four colored (green, yellow, orange and red) boxes to rate the risk level of the likelihood for the user stories as shown in Figure 3. 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 finishes with the estimation individually, team members are required to simultaneously show their estimation in the group. On having a huge difference of estimation colors, the estimator explains the differences and conducts a discussion to clarify the situation. PO also has the right to assign the highest estimation value of the likelihood factor to the user story affected. When the team reaches a unanimous decision, the next set of estimation cards is provided for estimation.

3.3 Risk Poker and Test Coverage Estimation

At the end of sprint planning meeting, the team obtains risk ratings for each of the discussed backlog items. At this stage, the product backlog items are broken down into smaller tasks for development and testing. The tasks are stored as sprint backlog items that have rating color assigned to them as discussed in the planning meeting. The assigned rating color provides an estimation of test coverage for determining how much testing is needed throughout sprint. Testing activity will be executed based on the test coverage estimation to obtain a "done" criterion as a stopper for testing activity before presenting the finished product to the user at the end of sprint. Table 2 describes the code coverage definition for test coverage defined for unit test.

3.4 Risk Graph

Risk Graph component is developed separately to allow Scrum team to manage sprint backlog items in a database and provide risk prioritization graph for sprint backlog items. The risk graph component displays sprint backlog items in high, medium and low-risk level categories for Scrum team to commit to throughout the sprint. Each item displayed in the Risk Graph component is also associated with the test coverage definition for testing activity.

Risk Graph component is accessed at the end of the sprint planning meeting, where the discussed user story is updated in the Sprint Backlog database through the Risk Graph component as shown in Figure 4 together with the risk rating color assigned both for likelihood and impact factor. The risk rating will determine how thoroughly testing should be done in the sprint. The Risk Graph component allows the team members 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 the test in sprint.

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 5. 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 Risk Graph component pairs the rating color of likelihood factor and impact factor to prioritize risk exposure as shown in Figure 5.

Once the Risk Graph prioritization is exported, the team will choose to develop and test on the sprint backlog items placed in the high-risk area first, followed by medium-risk level items as plotted in the graph. The team member is able to view the test coverage estimated for the corresponding sprint backlogs as shown in Figure 6.

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.

3.5 Integrating the Proposed Testing Model Inside a Scrum Methodology

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 to provide a group consensus upon analyzing risks and estimating test coverage. Figure 7 shows the incorporation of the proposed testing model into the Scrum workflow. The proposed sections are highlighted with red circles in the Scrum processes. In a larger picture, Risk Poker technique will affect the following processes in Scrum, as shown previously in Figure 7; 1) Listing Product Backlog, 2) Sprint Planning Meeting, 4)

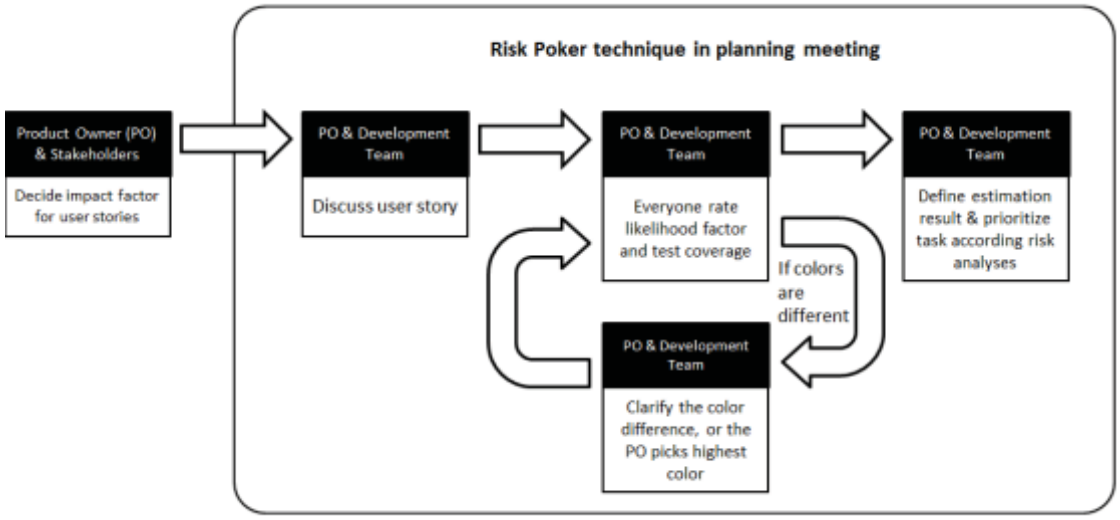

Figure 2 Detailed view of Risk poker activity flow diagram [6].

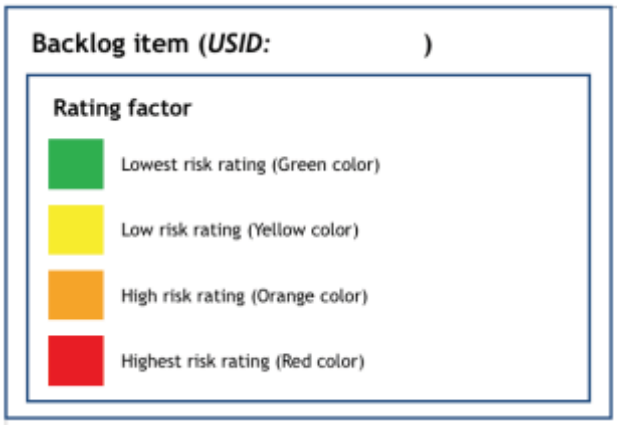

Figure 3 Rating card for impact factor and likelihood factor estimation.

Table 2 Coverage complexity for unit test [7]

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.

Update Sprint Backlog to manipulate Risk Graph prioritization, and 5) Testing activity in sprint.

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 8 shows the integrated testing model inside Scrum process.

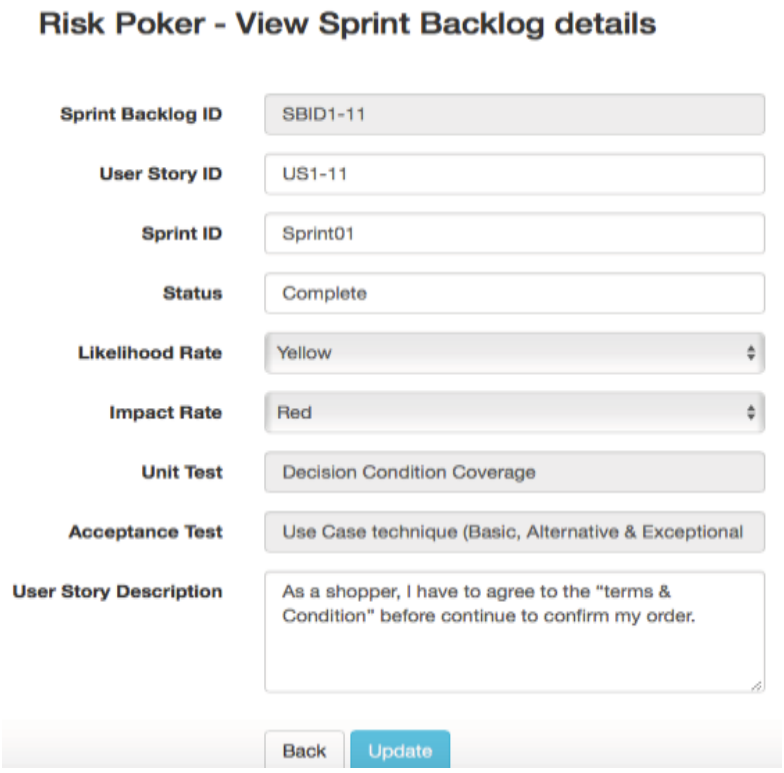

Figure 4 Update Sprint Backlog details.

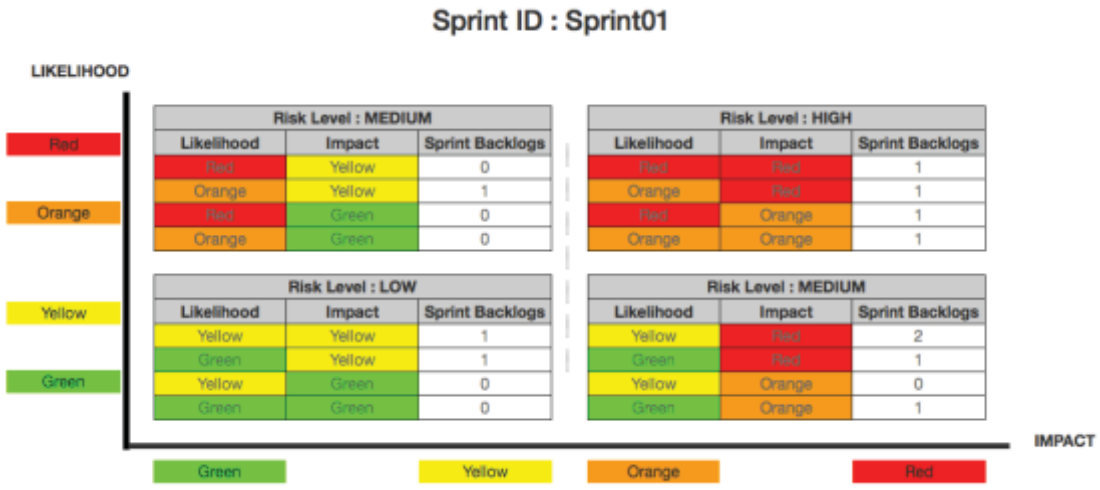

Figure 5 Risk Graph prioritization.

4. EXPERIMENTAL DESIGN

4.1 Objectives

The validation of experiment result is aimed to answer two research questions: *RQ1: Is the test coverage provided by Risk Poker-based proposed model adequate compared to the statistical combination of individuals?*, and *RQ2: How does Risk Poker-based proposed model estimation differ from the averaged statistical combination of individual estimations?*.

4.2 Participants

We conducted our experiment with final year undergraduate students of Software Engineering course. The students were already done with the Software Verification & Validation module and were assumed to be familiar with the test planning process, able to construct test cases for testing purpose and able to perform various types of testing technique throughout the software project. We had 3 experimental groups of students (who estimated risk level and test coverage using the proposed method), and the other 3 control groups of students (who used averaged statistical combination of individual estimates for further comparison). Data and result collected in the study are used to an-

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 6 Test coverage estimation for the risk level Medium.

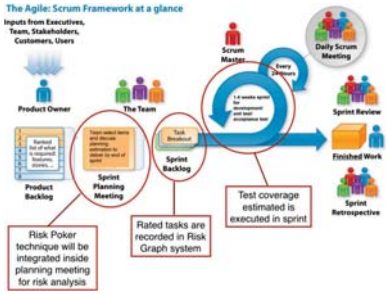

Figure 7 Proposed Software Testing Strategy incorporated in Scrum workflow.

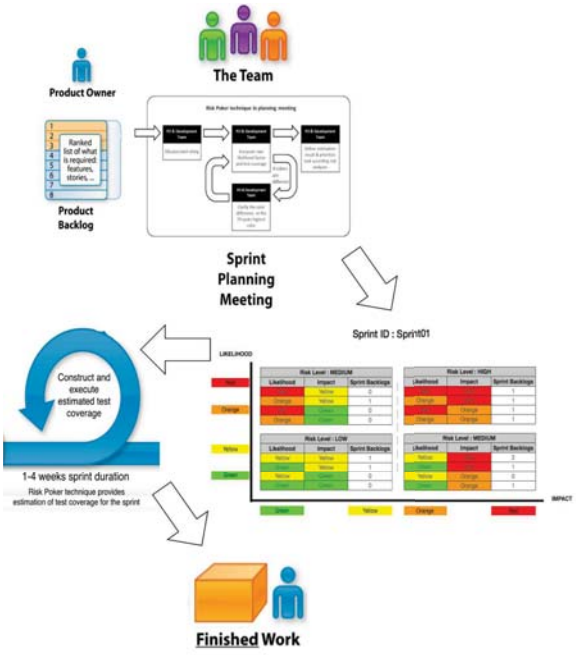

Figure 8 Integration of the Proposed Testing Model with Scrum Work Flow.

alyze the student group performance when using the proposed method compared to the averaged statistical combination of individual estimates. Table 3 listed the summary of the experiment participants' details.

Figure 9 Risk Poker Technique Affect Testing Activity in Sprint.

Table 3 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)

4.3 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 was 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 took 9 weeks to complete the estimation and testing. Each team consisted of 4 students who acted as a self-organizing and self-managing Scrum Team, responsible for analyzing risks and risk level, estimating test coverage and executing testing on an e-commerce system based on the given user stories.

The software project developed was an open source e-commerce system developed for a market-based client named Marvel Beads. The client provided the requirements and the primary researcher played the role of Product Owner in collecting requirements from client.

Each user story consisted of a short description of the required functionality to be discussed by team members during the planning meeting to analyze risks, prioritize the tasks and assign how much test coverage is needed. Figure 10 shows two examples of user stories and the risk identification obtained from the experimental student teams.

Testing activity was executed in a two-week sprint based on the test coverage estimation obtained during the planning meeting. Table 4 listed the summary of experiment materials and environment prepared to execute the experiment validation for the proposed testing model.

4.4 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.

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

1. This research conducted a briefing session (Step A; Table 5) 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 started the project with Sprint Planning Meeting. In Sprint Planning Meeting, the student teams

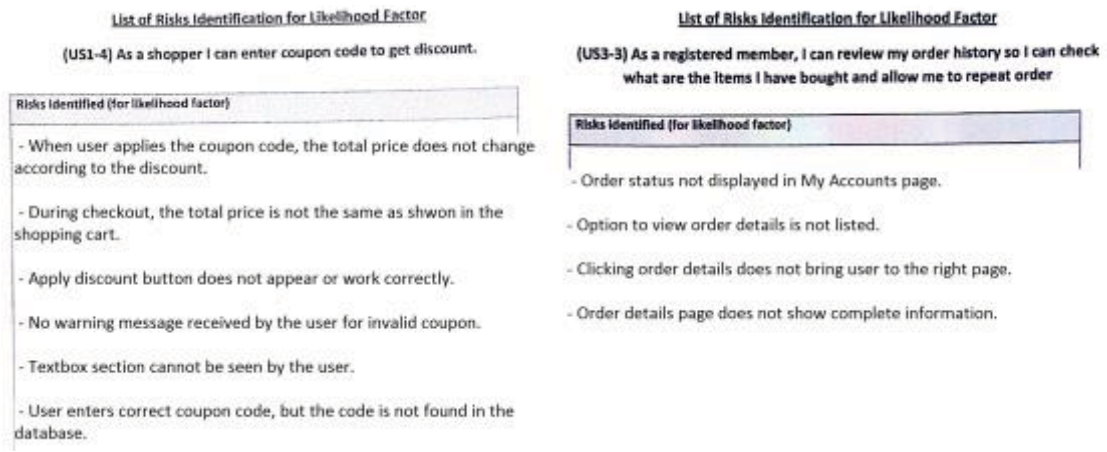

Figure 10 Sample of user stories and risks identification.

Table 4 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

were required to discuss user stories, identify risks and analyze risks associated with the user stories for prioritization later (Step C). The list of risks identified was recorded as shown in Figure x for risk assessment discussion. Once discussion took place and everyone was cleared with the related issues for the user story, student teams were provided with rating card to rate risk level for the discussed user story. The risk level assigned was associated with related test coverage for each level. The risk rating card consisted of four colored risks rating: Red as the highest risk, followed by orange, yellow and green. The highest risk level was associated with the most intense code coverage for a unit test, which is multiple condition coverage, followed by condition determination coverage, decision condition coverage, and decision coverage. The intensity of test coverage estimation assigned to each risk level is in accordance with unit test code coverage complexity defined in ISTQB, as discussed in Section 3.3 Risk Poker and Test Coverage Estimation.

- The experimental student group was 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 estimated test coverage and prioritized risks using averaged statistical combination of individual estimations where each team member was 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 members to reveal rating result. If there is a difference in color of rating, they will discuss the color difference and is-

Table 5 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 component to prioritize the highest risks level tasks to the lowest risks level
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 tests result for both experimental and control group.

sues related. 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 different again at this time around, the team uses the highest risk level rating. The rating is updated in Risk Graph component to prioritize and assign test coverage of the user story.

- On the other hand, the rating card for control group student teams contains score points for each color risk level to be averaged to get the statistical combination of individual estimations. Control students estimated the risk level individually on the rating card and then presented the rating result together with other team members to reveal result rating. The scores of the rating estimated by team members were accumulated amongst team members and then the average was calculated to get the risk level score. This average score is used to prioritize the user story and assigned with appropriate test coverage according to the average score using the Risk Graph component.

4. On starting the sprint formally after completing the planning phase, the experimental team started testing the user stories according to the test coverage assigned to the highest risk item, followed by medium risk allocated item and ended with the low-risk product (Step D) as shown in the Risk Graph component.

5. At the end of sprint, the teams provided a list of fault exposed and the test result (Step E).

6. Data were collected to evaluate the performance of Risk Poker and test coverage implementation in Scrum as compared to the averaged statistical combination of individual estimates (Step F).

5. EXPERIMENTAL RESULTS

The risk level and test coverage estimation of 34 user stories were performed to evaluate the proposed method as compared to the averaged statistical combination of individual estimations. Student teams in the experiment are required to discuss and analyze risks of the user stories provided to estimate test coverage. Experimental student teams provide test coverage estimation after analyzing risks using Risk Poker technique whilst the controlled student teams provide test coverage estimation based on the average scores of their individual judgment. Thus, the benchmark set in this paper is focused on test coverage estimation, where the exposed fault result obtained from student teams are measured against the seeded fault planted in the system.

For monitoring and observing the results of the proposed model, data were collected throughout the process and test results are collected at the end of each sprint. The test result is a report of how much fault is exposed throughout testing process particularly in each sprint for each team. The exposed fault was compared to the seeded fault to measure test coverage adequacy. Basic descriptive statistics for the test result of the teams are presented in Table 6.

5.1 RQ1: 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 the 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 estimation is to expose seeded faults. The *BRE* of both experimental student teams and control student teams are calculated as follows:

$$BRE = \frac{\left|\text{seeded}_{\text{fault}} - \text{exposed}_{\text{fault}}\right|}{\min\left(\text{seeded}_{\text{fault}}, \text{exposed}_{\text{fault}}\right)}$$

In order to answer RQ1, the mean *BRE* of the fault exposure was calculated 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 11 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 7 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 that 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). A straightforward analysis of the results has suggested that mean *BRE* of experimental student teams (0.24) is smaller than the mean *BRE* of the controlled student teams (0.50) as depicted in Figure 11. 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.

5.2 RQ2: How does Risk Poker-based proposed model estimation differ from the averaged statistical combination of individual estimations?

Referring to the descriptive statistic as stated previously in Table 6; 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 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 8 showed 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 use 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.

6. DISCUSSION

6.1 RQ1: Is the test coverage provided by Risk Poker-based proposed model adequate compared to the statistical combination of individuals?

Referring back to Figure 11, this research has learned that the *BRE* mean for the control group (Group B-3, B-5 & B-6) is greater than the experimental group, thus it indicates that the

Table 6 Statistics.

	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

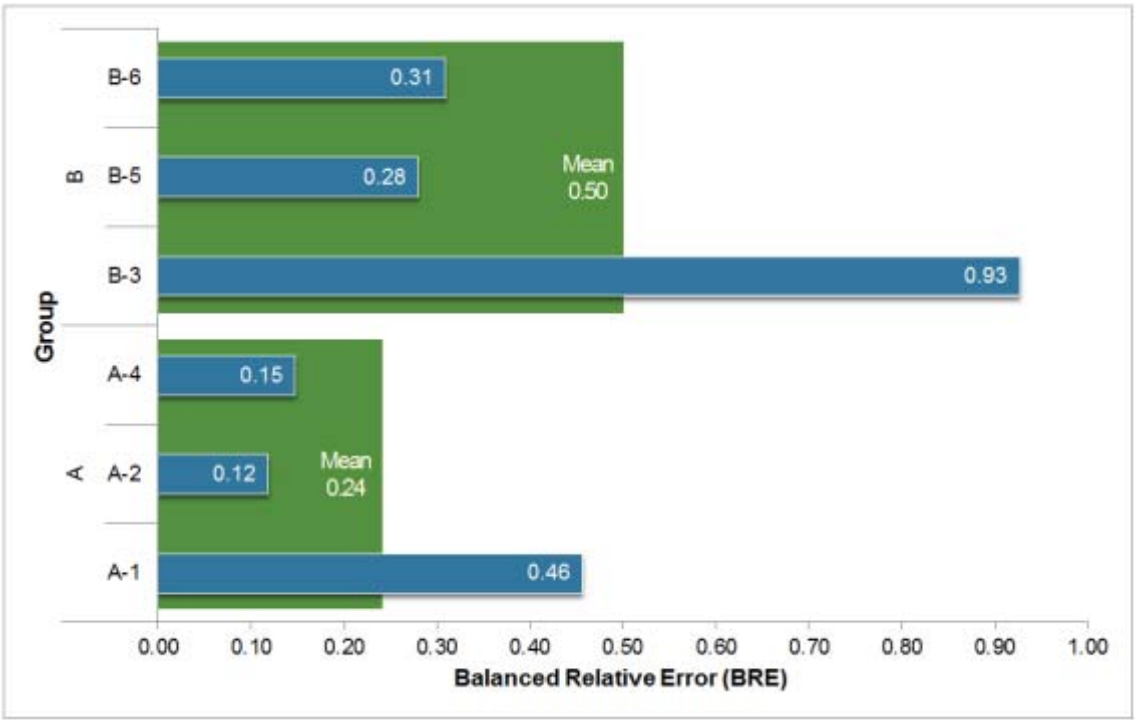

Figure 11 *BRE* mean for experimental and control group.

Table 7 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

control group's test coverage estimation is less accurate in exposing seeded fault. The range of control group's mean is 5.00 compared to the experimental group which is 2.00 as shown earlier in Table 6 Statistics. The higher range of unexposed seeded fault for control group indicates that test coverage estimation by the 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 faults) occurred in two out of three control group's estimation as shown in Figure 12 also indicates inadequate test coverage estimation

Table 8 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

technique for testing to cover required functionality to detect the fault. These issues have proven that Risk Poker technique is able to provide a relevant estimation of test coverage when the group is allowed to discuss their rating and concerns, where hidden issues can be highlighted for the task rating and estimation. Furthermore, this result has shown a significant difference in statistical tests presented previously in Table 6 which indicates that Risk Poker estimates provided by student teams are more accurate than the averaged statistical combination of individual estimations.

6.2 RQ2: 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 disagrees with [39] that face-to-face meetings are harmful to 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.

6.3 Study Validity

To increase the validity of this study, we studied 6 teams, working on the same project and 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 emulating an industrial environment as close as possible. 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 faults. Nevertheless, the main threat to external validity remains that only one project was used.

Based on statistical analysis, the results can be generalized only to students of the final 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 actual product owner in the 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 a 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. In addition, on a bigger scale, students 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 a more statistically comparable result. 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 the accuracy of before implementing the proposed technique with after implementing the proposed technique would help.

7. CONCLUSION AND FUTURE WORKS

This research has identified a suitable testing strategy that could fit agile projects following Scrum and able to perform through the experiment validation executed with undergraduate student teams in a software development project following Scrum. 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 where *BRE* mean for

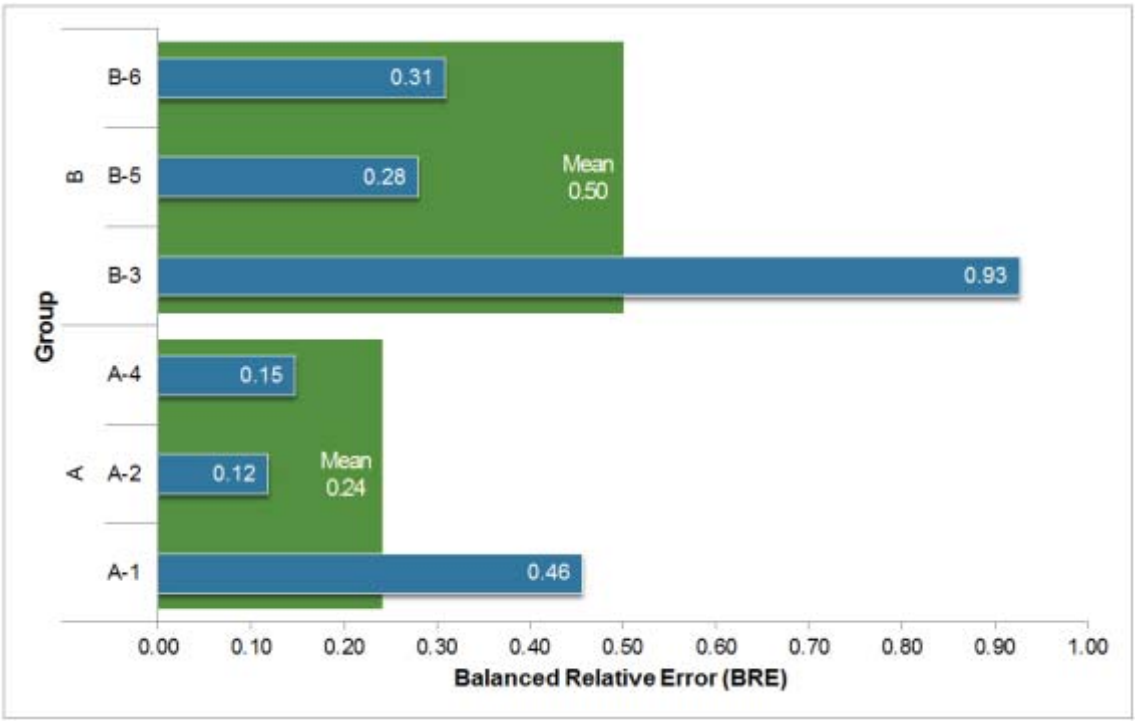

Figure 12 Unexposed fault quantity.

the averaged individual estimations (*BRE* mean=5.00) is greater than Risk Poker estimation (*BRE* mean=2.00).

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 the 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. At the same time, measure if there is any improvement over time in the risk assessments activity.

- Study the outcome of a software development project that initially does not apply the method proposed by this research, and then in the middle of the project, apply the proposed method in order to identify and measure the improvements brought by the proposed method compared to the initial process.

- Apply similar case study to 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.

- Conduct a similar case study on a real software project for agile which initially used group discussions for estimation. Then, to understands how exactly Risk Poker improved predictions, examine and analyze the risk assessment obtained from Risk Poker group compared to the initial group discussions result.

Acknowledgment

This research is supported by Fundamental Research Grant Scheme (FRGS) Grant No.: FP002-2016 from the Ministry of Higher Education, Malaysia.

REFERENCES

1. VersionOne.com, *The State of Agile Development*, in *State of Agile Survey 2010*. 2010: https://www.versionone.com/pdf/2010_State_of_Agile_Development_Survey_Results.pdf.
2. Grenning, J., *Planning poker or how to avoid analysis paralysis while release planning.* Hawthorn Woods: Renaissance Software Consulting, 2002. **3**.
3. Schwaber, K., *Agile Project Management with Scrum*. 2004: Microsoft Press.
4. Schwaber, K. and M. Beedle, *Agilè Software Development with Scrum.* 2002.
5. Nelson, C.R., G. Taran, and L. de Lascurain Hinojosa, *Explicit risk management in agile processes*, in *Agile processes in software engineering and extreme programming*. 2008, Springer. p. 190-201.
6. Van de Laar, J., *Risk Poker: Risk based testing in agile projects.* Software Quality DaySS, 2012: p. 51.

7. Thomas Müller, D.F., ISTQB WG Foundation Level. *Certified Tester Foundation Level Syllabus*. 2011 [cited 2015 24 January 2016]; Version 2011:[Available from: http://www.istqb.org/downloads/send/2-foundation-level-documents/3-foundation-level-syllabus-2011.html4.
8. Hellmann, T.D., et al. *Agile Testing: Past, Present, and Future–Charting a Systematic Map of Testing in Agile Software Development*. in *Agile Conference (AGILE), 2012*. 2012. IEEE.
9. Khalane, T. and M. Tanner. *Software quality assurance in Scrum: The need for concrete guidance on SQA strategies in meeting user expectations*. in *Adaptive Science and Technology (ICAST), 2013 International Conference on*. 2013. IEEE.
10. Winter, J., et al., *Meeting organisational needs and quality assurance through balancing agile and formal usability testing results*, in *Software Engineering Techniques*. 2011, Springer. p. 275-289.
11. Stolberg, S. *Enabling Agile Testing through Continuous Integration*. in *Agile Conference, 2009. AGILE '09.* 2009.
12. Stallbaum, H., A. Metzger, and K. Pohl. *An automated technique for risk-based test case generation and prioritization*. in *Proceedings of the 3rd international workshop on Automation of software test*. 2008. ACM.
13. Kasurinen, J., O. Taipale, and K. Smolander, *Test case selection and prioritization: risk-based or design-based?*, in *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*. 2010, ACM: Bolzano-Bozen, Italy. p. 1-10.
14. Mahnič, V. and T. Hovelja, *On using planning poker for estimating user stories.* Journal of Systems and Software, 2012. **85**(9): p. 2086-2095.
15. Paulk, M.C., *Agile methodologies and process discipline.* Institute for Software Research, 2002: p. 3.
16. Li, M., et al. *A risk-driven method for eXtreme programming release planning*. in *Proceedings of the 28th international conference on Software engineering*. 2006. ACM.
17. Bannerman, P.L., *Risk and risk management in software projects: A reassessment.* Journal of Systems and Software, 2008. **81**(12): p. 2118-2133.
18. Hartmann, J., L.M. Fontoura, and R.T. Price, *Using Risk Analysis and Patterns to Tailor Software Processes.* XIX Simpósio Brasileiro de Engenharia de Software, Uberlândia, 2005.
19. Hall, E.M., *Managing risk: Methods for software systems development*. 1998: Pearson Education.
20. Boehm, B.W., *Software risk management: principles and practices.* Software, IEEE, 1991. **8**(1): p. 32-41.
21. Felderer, M. and R. Ramler, *Risk orientation in software testing processes of small and medium enterprises: an exploratory and comparative study.* Software Quality Journal, 2016. **24**(3): p. 519-548.
22. Nyfjord, J., *Towards integrating agile development and risk management.* 2008.
23. Ramler, R. and M. Felderer. *A process for risk-based test strategy development and its industrial evaluation*. in *International Conference on Product-Focused Software Process Improvement*. 2015. Springer.
24. Williams, L., M. Gegick, and A. Meneely, *Protection Poker: Structuring Software Security Risk Assessment and Knowledge Transfer*, in *Engineering Secure Software and Systems*. 2009, Springer. p. 122-134.
25. Kettunen, V., et al. *A study on agility and testing processes in software organizations*. in *Proceedings of the 19th international symposium on Software testing and analysis*. 2010. ACM.
26. Garousi, V. and J. Zhi, *A survey of software testing practices in Canada.* Journal of Systems and Software, 2013. **86**(5): p. 1354-1376.
27. Jogu, K.K. and K.N. Reddy, *Moving Towards Agile Testing Strategies.* Cvr journal of science & technology, 2013: p. 88.
28. Moløkken-Østvold, K. and M. Jørgensen, *Group Processes in Software Effort Estimation.* Empirical Software Engineering, 2004. **9**(4): p. 315-334.
29. Schatz, B. and I. Abdelshafi, *Primavera gets agile: a successful transition to agile development.* Software, IEEE, 2005. **22**(3): p. 36-42.
30. Shahid, M., S. Ibrahim, and H. Selamat. *An Evaluation of Current Approaches to Support Test Coverage Analysis*. in *International Conference on Computer Engineering and Technology, 3rd (ICCET 2011)*. 2011. ASME Press.
31. Dang, T. and T. Nahhal, *Coverage-guided test generation for continuous and hybrid systems.* Formal Methods in System Design, 2009. **34**(2): p. 183-213.
32. Lawrence, J., et al. *How well do professional developers test with code coverage visualizations? An empirical study*. in *Visual Languages and Human-Centric Computing, 2005 IEEE Symposium on*. 2005. IEEE.
33. Marré, M. and A. Bertolino, *Using spanning sets for coverage testing.* Software Engineering, IEEE Transactions on, 2003. **29**(11): p. 974-984.
34. Zhu, H., P.A. Hall, and J.H. May, *Software unit test coverage and adequacy.* Acm computing surveys (csur), 1997. **29**(4): p. 366-427.
35. Walkinshaw, N., et al., *Increasing functional coverage by inductive testing: a case study*, in *Testing Software and Systems*. 2010, Springer. p. 126-141.
36. Woodward, M.R. and M.A. Hennell, *Strategic benefits of software test management: a case study.* Journal of Engineering and Technology Management, 2005. **22**(1): p. 113-140.
37. Gargantini, A. and E. Riccobene, *ASM-based testing: Coverage criteria and automatic test sequence generation.* Journal of Universal Computer Science, 2001. **7**(11): p. 1050-1067.
38. Cai, X. and M.R. Lyu. *Software reliability modeling with test coverage: Experimentation and measurement with a fault-tolerant software project*. in *Software Reliability, 2007. ISSRE'07. The 18th IEEE International Symposium on*. 2007. IEEE.
39. Armstrong, J.S., *How to make better forecasts and decisions: Avoid face-to-face meetings.* Foresight, 2006. **5**: p. 3-8.