Faculty of Science and Engineering

School of Electrical Engineering, Computing and Mathematical sciences

Gamifying Software Testing – A Focus on Strategy & Tools Development

Navid Memar

This thesis is presented for the Degree of
Doctor of Philosophy
of
Curtin University

September 2019

Declaration

To the best of my knowledge and belief this thesis contains no material previously published by any other person except where due acknowledgement has been made. This thesis contains no material which has been accepted for the award of any other degree of diploma in any university.

Signature: …….

Date: …….

Navid Memar
Faculty of Science and Engineering
School of Electrical Engineering, Computing and Mathematical Sciences
Curtin University
September 2019

In loving memory of my Brother:
Payam Memar 31 May 1980 – 15 February 2005

My beautiful brother, you may be gone but you will never be forgotten.

Life can never be the same for me without you!

Rest in peace my brother.

Abstract

Software testing is an essential activity within the software development life cycle, because bugs and occasional failures in software systems can cause significant challenges for industry and the economy. However, software testing is a costly activity and is often an undervalued job. Among existing software testing techniques, unit testing is one that developers often use to detect potential bugs within the code. However, unit testing is known as a challenging activity which could potentially affect testers' decision in performing testing activity using this technique. In contrast, gamification is a concept that uses game elements in a non-game context. Factors such as high cost, low appreciation, and the monotonous environment in software testing make it an excellent target for gamification.

This thesis aims to provide new testing strategies and tools that facilitate software testing to be more efficient, while providing the tester with an engaging and rewarding testing environment. For this purpose, gamification has been selected as a potential solution to raise tester engagement, as it can potentially remedy the high degree of repetition and ensuing boredom for a tester in the testing phase.

The results emerged from various empirical studies carried out are presented. These studies helped in validating the key factors required to develop a gamified software system, in which the software testing process is gamified to improve the engagement and experience of the software testers. Furthermore, through focus group sessions, the final gamified platform was validated to ascertain the usefulness of the proposed tool in increasing tester motivation levels and providing them with an engaging and rewarding environment. The findings from these studies indicate that the developed tool could be a solution that helps improve software tester engagement and testing experiences. The results also helped to identify the key factors required for improving the motivation, engagement, and experience of software testers.

The gamified platform allows the researcher to study the effect of gamification as an additive method that could help improve the performance of software testers. Results obtained from the evaluation sessions indicates that gamification could be helpful to increase software testers' performance. After conducting parallel studies, the importance of introducing new metrics that can help in quantifying the performance of software testers fairly and more accurately has been identified. Thus, new metrics were proposed to help achieve this aim. By using the proposed metrics, software testers' performance could be evaluated based on the importance of identified bugs. Results indicate that the proposed metrics can better analyze and compare the performance of software testers in a gamified testing environment. The results also suggested that time restriction may be an element that compromises the performance of software testers and the quality of the written software test code. Moreover, the results from the final study assists in understanding whether prior knowledge of the evaluation metric would lead to software testers identifying more difficult bugs that exists in the code. The results of this study indicate that introducing the gamified metric prior to the evaluation session significantly helps the testers in achieving a higher score compared to the rest of the testers (who did not know about the proposed gamified evaluation metric).

Acknowledgments

Undoubtedly, the process of completing my doctoral degree has been long, but it has been a fruitful journey. When I started my PhD, I had very little information about what this journey would involve. Through this process, I was able to improve many aspects of which I had little knowledge. This helped in improving my knowledge, problem solving, professional and academic communication, and many other related and useful skills. I am certainly grateful to many individuals for assisting me over these past five years.

First, I would like to express the deepest of gratitude to my supervisors, Associate Professor Aneesh Krishna, and co-supervisors Professor Tele Tan and Dr. David McMeekin, whose vision, encouragement, patience, and useful guidance have helped me through my PhD journey at the school of Electrical Engineering, Computing and Mathematics Sciences, Curtin University. I would like to thank Associate Professor Aneesh for believing in my abilities and for his motivation and encouragement when needed it the most. I would like to thank Professor Tele for his great technical feedback that has greatly helped in this thesis writing process. I would also like to thank Dr. David for his invaluable comments and suggestions throughout the course of this study.

I am indeed very grateful to Professor Ling li, Head of School of Electrical Engineering, Computing and Mathematics Sciences, for her valuable support through the most difficult times of my PhD journey.

My greatest gratitude goes to my wonderful family. I would like to thank my parents for providing me this opportunity to study in this beautiful country and for providing me with tremendous support during these years. Special thanks to my brother Dr. Amir Memar and sister in-law Dr. Ghazaleh Pourfallah for their outstanding support, patience, and understanding during the course of this journey.

Many thanks go to my cousin Mr. Ali Masoum, who was always the first to help during difficult times. There are other people who been of great assistance in one way or another during my doctoral degree journey, and I am thankful to them all.

Credits

Portion of the material in this thesis have previously appeared in the following publications:

1. **Navid Memar**, Aneesh Krishna, David A McMeekin and Tele Tan (2017). Gamification of Information System Testing-Design Consideration through focus group discussion. The 26th International Conference on Information Systems Development (ISD 2017), Larnaca, Cyprus.

   The author's contribution includes conducting literature review, conducting focus group and evaluation sessions, implementation of the gamified software-testing platform, preparing survey questionnaires, analyzing the data and drafting the paper. The co-authors contributed by way of providing feedback on each step of the project, reviewing results, editing survey questionnaires and editing the paper.
2. **Navid Memar**, Aneesh Krishna, David A McMeekin and Tele Tan (2018). Gamifying Information System Testing-Qualitative Validation through Focus Group Discussion. The 27th International Conference on Information Systems Development (ISD 2018), Lund, Sweden.

   The author's contribution includes the development of the final gamified software-testing platform, conducting focus group and evaluation sessions, preparing survey questionnaires, analyzing the data and drafting the paper. The co-authors contributed by way of providing constant feedback on each step of the project, reviewing results, suggesting mathematical methods for analyzing the data, editing survey questionnaires and editing the paper.
3. **Navid Memar**, Aneesh Krishna, David A McMeekin and Tele Tan 2020. "Investigating information system testing gamification with time restrictions on testers' performance." Australasian Journal of Information Systems 24: 1-21.

To Whom It May Concern,

I, Navid Memar, contributed to the above listed publications as indicated therein.

_____

Navid Memar

_____

Supervisor: Associate Professor. Aneesh Krishna

Table of Contents

Chapter 1

Introduction

With our increasing dependence on software to manage our daily activities, improving software quality is becoming critical to society. The importance of software in almost every industry sector is dramatically increasing. As software innovation progresses rapidly, the number of software failures may unavoidably increase, and if a company's software failure rate is too high, the future of the company would be destroyed (Charette, 2005). Software testing is therefore a vital practice in the software development process, and one way to reduce the number of software failures is through such testing. It is estimated that almost 70% of software costs and resources are related to maintenance of existing software systems (Pfleeger & Atlee, 2010; Swanson & Beath, 1989). Software systems are an important part of our daily lives, and they are rapidly taking over operations and control in industries such as enterprise management, financial processes, and automated vehicles. Thus, it is vital to provide assurances that these complex systems can meet the requirements of proper operation before being released to the owners and intended users.

Our research focuses on developing new testing strategies and tools that will enable software testing to be performed more efficiently while providing an engaging and rewarding test environment. To achieve this, the current monotonous nature of testing will be addressed through the gamification of software testing.

1.1 Research objectives

In this study, we have quantified the influence of gamification as an additive tool on the performance of software testers. Thus, we have proposed and developed a gamified software testing platform that could help remedy some of the issues in the software testing domain. We have also proposed a gamified performance evaluation metric to

boost software tester motivation to identify more important bugs within the software during testing. The main objectives of this thesis are as follows:

A. To study and understand the process of testing and inspection of software code for defects.
B. To apply human computer interaction (HCI) principles to help model the decision-making process of testers and thereby determine the most efficient testing methods.
C. To explore gamification for improving software testing quality and efficiency.

The aim of the thesis is to improve the engagement of software testers and to improve the software testing quality by understanding the behavior of software testers and by using gamification in software testing process. Gamification is used for improving tester engagement and testing quality. In this project, gamified mechanics is used in a continuous integration environment to increase engagement, encourage software testers, and avoid monotonous and repetitive environment and tasks. The gamification mechanism could be used to engage software testers. By engaging software testers, the testing quality and the productivity of testers would increase. Gamification could engage and motivate software testers to improve their efficiency and reliability.

1.2 Hypothesis

The performance of software testers can be influenced by the boring, monotonous, and repetitive nature of software testing. It is hypothesized that gamification could help assist in improving the engagement and motivation levels of software testers. Furthermore, it is hypothesized that gamification could also boost tester performance, efficiency, and reliability.

1.3 Thesis structure

This thesis consists of seven chapters. An outline of each chapter is provided below.

Chapter 1 (current chapter) provides information about the research objectives, hypothesis, and thesis structure.

Chapter 2 provides some background information that includes an overview of the software testing, gamification, examples of software failures due to lack of testing, issues in software testing and linkage of software testing and gamification, literature review on similar studies, and finally a summary of the background study.

Chapter 3 presents the design consideration of the gamified software testing platform through results obtained from various focus group sessions with software developers and software testers. This chapter consists of an introduction section that describes information related to the content of the chapter. Furthermore, the methodology section presents the prototype development methods and the application of each game element in the selected software testing application. For this purpose, a chosen relative methodology is discussed to obtain optimal results. The initial developed prototype has been used in focus groups sessions to study the essential game-based design elements that could help software testers and businesses to achieve the desired outcome. The results obtained from these focus groups sessions are presented in this chapter. This chapter was published in the 26th international conference on Information Systems and Development (ISD) in 2017.

Chapter 4 presents the development and evaluation of the final gamified software testing platform. In this study, the aim is to understand whether the developed features and selected core elements within the developed gamified platform could help in fulfilling the objectives of the gamified software testing platform. Furthermore, the chosen method for the evaluation of the developed gamified software testing is discussed. Finally, the results obtained from the various focus group sessions with software developers and software testers are presented. This chapter was published in the 27th international conference on Information Systems and Development (ISD) in 2018.

Chapter 5 focuses on the evaluation of the developed gamified software-testing tool to enhance the software-testing experience with the aim of providing a more engaging and rewarding environment for software testers. This chapter also introduces a new gamified metric that assists in evaluating the effectiveness of software tester performance fairly and more accurately. Lastly, this chapter studies the importance of time pressure on the software testing practice. Results of this study are obtained by conducting various focus group sessions with software developers and software testers. This study has been submitted to the Australasian Journal of Information Systems.

Chapter 6 presents the effects of using the gamified software testing-platform on the performance of software testers with regards to detecting different levels of code defects and the quality of the participants' written test codes. Furthermore, this chapter investigates the significance of introducing the gamified evaluation metric, which is introduced in Chapter 5, to the participants prior to the testing activity. This study helps to discover whether the software testers could detect more difficult bugs when performing the testing task with prior knowledge of the evaluation metric. The results of this study are provided after evaluating software tester performance using the developed gamified software testing platform.

Chapter 7 briefly discusses all the experimental results presented in this thesis and forms conclusions. Moreover, this chapter presents proposals for future work. A schematic diagram of the thesis structure can be found in Figure 1.1.

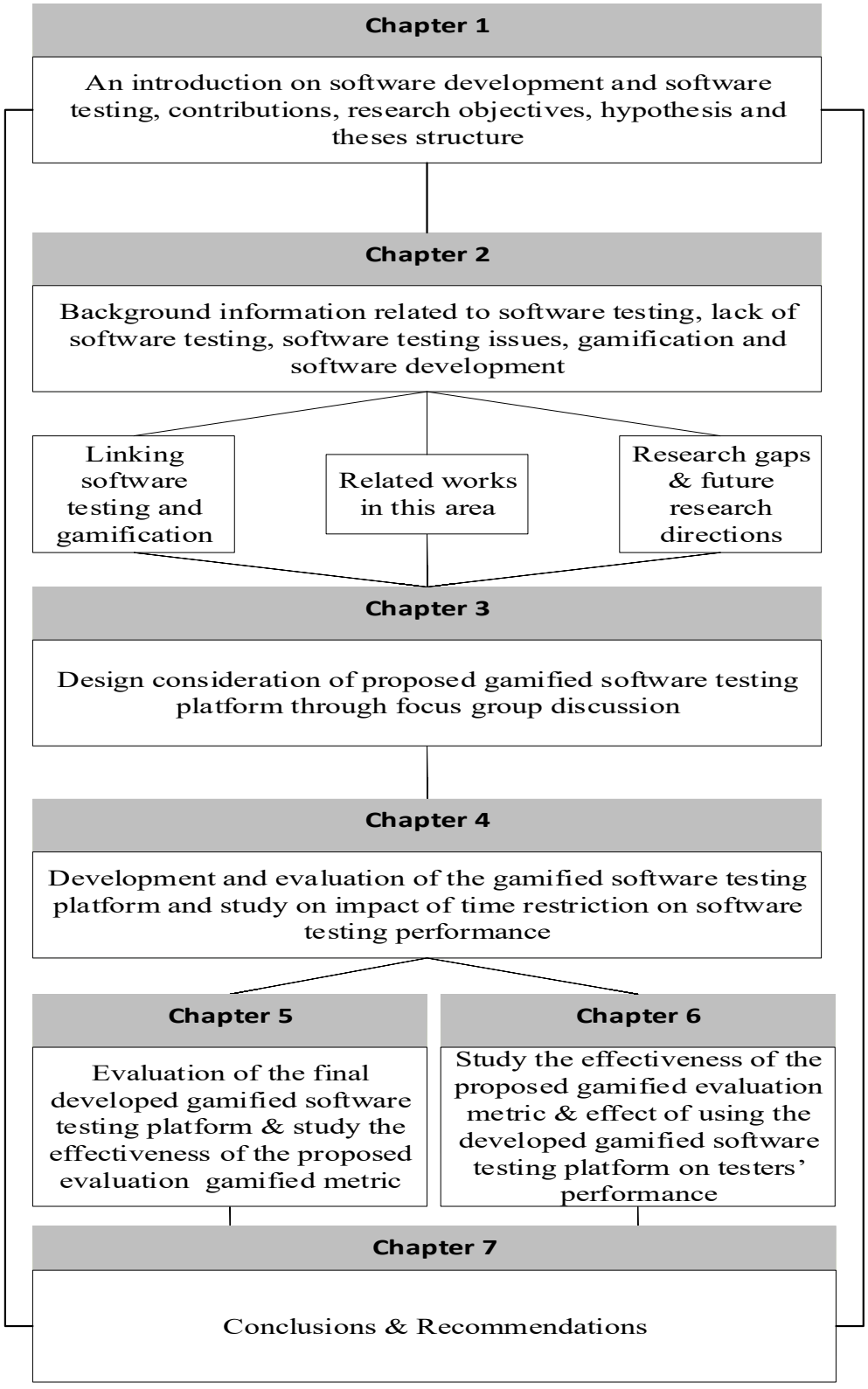

Figure 1.1: Schematic diagram of thesis structure.

Chapter 2

Literature survey

2.1 Introduction

In this chapter, the relevant literature and work related to this thesis will be presented. The work presented in this thesis falls into the domain of Software Testing and Gamification. This chapter builds the foundation for this research study. The outcome of the survey reveals the gaps in the current literature, which will be the focus of this research and thesis.

The first two sections introduce software testing and its importance to the software industry. The third section examines current issues in the software testing domain. In the fourth section, an overview of gamification and game elements is presented and the advantages of this approach are discussed. The final section of the literature review examines the linkage between gamification and software testing.

2.2 Software testing

Software testing is an activity that aims to evaluate and improve software quality by revealing software defects and problems. Software testing is an important phase in the software development lifecycle. Although there are various tools to ensure the quality of software systems, many software systems still suffer due to poor testing. After decades of research on various software quality assurance techniques, testing activity remains one of the most studied and used methods for improving software quality (Orso & Rothermel, 2014). Software testing consists of a detailed verification of the program behavior on a set of selected test cases (Bourque & Fairley, 2014). It is a vital activity of both the development and maintenance phases (Adrion, Branstad, & Cherniavsky, 1982; Schach, 1996). The main objective of software testing is to discover the existence of faults in the software product and to verify whether the

software meets its requirements. Software is considered to be of high quality, if it is fault free, user friendly, and provides satisfaction to its client (Singh, Singh, & Singh, 2010).

Software testing helps in providing assurance that the software is working correctly and according to its specifications. In other words, software testing can be used to verify and validate the software quality. Software testing is an important part of software engineering. Software testing takes 40% to 50% of the development effort in most projects and this percentage could be significantly higher for more complex systems that require higher level of quality (Luo, 2001). Testing is involved in every stage of the software development lifecycle. However, testing in each part of software development is different (Luo, 2001). Software testing activity could be viewed from two perspectives: technical and socio-technical (Myers, Sandler, & Badgett, 2011). It should be noted that software testing fulfils a social-technical outcome even though most parts of software engineering is purely technical (Martin, Rooksby, Rouncefield, & Sommerville, 2007). Software testing is an expensive activity in the software development process, which can consume up to 50% of the total cost of software development projects (Boehm & Abts, 1999). The scientific literature confirms that continuous testing during development can reduce project costs. For example, Maximilien and Williams (Maximilien & Williams, 2003) stated that minimizing the number of defects is one of the most effective ways to reduce the development cost. They also reported that the detection of errors after the release of the final product is up to 30 times more expensive than detection during the design or architectural development phase. A research by National Institute of Standards and Technology indicated that insufficient software testing costs the U.S economy nearly $59.5 billion annually (Tassey, 2002).

Software testing can be performed either manually or automatically, and one way to reduce the number of software failures is through robust testing. Recently, there has been a trend in the automation of software testing activities. However, the importance of testing by human software engineers will never disappear (Fraser, 2017a, p. 2). For instance, Ciupa et al. (Ciupa, Meyer, Oriol, & Pretschner, 2008) conducted a research

and compared different testing strategies. They found that the automated tool helped to find a high number of errors in a shorter time but missed out on faults that humans can easily detect.

Black box, white box, and grey box testing methods are deployed to detect various faults such as deign errors, statement errors, input or test errors, and specification errors. Testing in each stage of software development is different (Luo, 2001).

Unit Testing: This kind of testing is usually conducted at the lowest level and is also known as module or component testing. Its purpose is to test the basic unit of software.

Integration Testing: Integration testing is conducted when two or more software units are combined to produce a larger arrangement.

System Testing: System testing ensures the quality of the entire software. In this stage, non-functional quality attributes such as security and maintainability are also checked.

Acceptance Testing: This stage of testing is done when the completed system is available to users. The aim of this testing is to build confidence that the completed system is ready to use rather than to find new errors.

Testing is the configuration of input, running the system over the input, and analyzing the output based on the input. Testing technique uses the strategy and technique to select input test cases and analyze the outcomes. The most recognized categories of testing techniques are functional and structural testing.

Functional Testing: This type of testing is based on the requirement specifications, and in this testing, it is not important to test the code. Functional testing is also known as black box testing. Functional testing focuses on the external behavior of the system (Luo, 2001).

Structural Testing: Structural testing focuses on the internal structure of the code. This type of testing is also known as white box testing. This technique makes it possible to test each branch and decision in the program (Bansal, 2014). The following different techniques are used in white box testing.

Control Flow Testing: Control flow testing practices the program control flow as model control flow testing, which could be useful in small programs or parts of a larger program. This technique has some limitations and disadvantages.

Basis Path Testing: This technique assures that every independent path of the code is tested. An independent path could be defined as a path through the code that introduces at least one new condition or processing statement (Gupta, 2014). It also allows the test case designer to produce a logical complexity measure of procedural design and then uses this measure as an approach for outlining a basic set of execution paths (Bansal, 2014).

Data Flow Testing: This testing uses a technique to state how data circulates around the program. It also declares how the value could be assigned to a variable.

Loop Testing: The aim of this testing is to focus on the validity of loop construct. Loops could be easy to test except in cases where dependencies exist.

There are also different forms of white box testing, such as desk checking, code walkthrough, and formal inspections, which could be listed in static white box testing techniques.

2.3 Lack of software testing

Lack of testing may result in problems, and many software problems are due to bad testing. For example, the New Derivation Trading System of the Tokyo Stock Exchange's derivative products failed in February 8, 2008. As a result, the trading system was unavailable for several months. Investigations showed that the failure was due to an initialization error with the memory within the server (Kurokawa & Shinagawa, 2008). There can be higher risks for not identifying hazardous software failures and allocating suitable safety requirements (Hawkins, Habli, & Kelly, 2013). Recent studies have demonstrated that software bugs may cause major financial losses, which would be critical for organizations (Goyal & Sardana, 2017). In early 2016, HSBC stated that a complex technical issue caused a major IT outage in their

organization (Goyal & Sardana, 2017). Moreover, in August 2015, HSBC failed to process 275,000 individual payments due to a bug in their electronic payment system (Goyal & Sardana, 2017).

Another example of catastrophic software failure involved the NASA Mars Polar Lander in January 1999. In this case, the cause of the accident was identified as software error that caused premature shutdown of decent engines in the lander (Hawkins et al., 2013). The loss of the Mars Polar Lander (MPL) is an example of bad software verification and testing, which were the main factors causing this incident. On 7th October, 2008, an in-flight issue occurred on an Airbus A330-303 en route to Perth from Singapore at a cruising altitude of 37,000 ft. Several passengers and crew on board were injured. Investigations revealed that one of the Inertial Reference Units (ADIRUs) started to output intermittently incorrect values of vital flight parameters to the rest of aircraft subsystems (Favarò, Jackson, Saleh, & Mavris, 2013).

The demand for software testing is clearly on the increase, which mirrors the rapid growth in the use of software for industrial and commercial purposes.

2.4 Issues with software testing

Software testing is usually tedious, monotonous and boring (Kanij, Merkel, & Grundy, 2013; Shah & Harrold, 2010) as well as time-consuming, difficult, and often inadequate (Alrmuny, 2014). With systems becoming more complex and the increased usage of IT and software critical systems, software testing challenges are expected to increase in the future together along with a corresponding negative impact on software quality (Bertolino, 2007). By taking the socio-technical aspect into consideration, the skill and motivation of software testers are vital to increase software quality and success in software testing (Bertolino, 2007).

Many research papers and reports have focused on software testing (Binder, 1996; Torkar & Mankefors, 2003; Whittaker, 2002). For instance, Torkar et al. (Torkar & Mankefors, 2003) stated the importance of having a more integrated testing methodology combined with the old configuration management method to increase the quality and efficiency of software testing. Software engineering requires sophisticated

methods and tools to facilitate developers in building error-free software. In addition, graduates and candidates looking for positions in the technology sector are more inclined to lean towards software development and do not show much interest towards software testing (Deak, Stålhane, & Cruzes, 2013). With the increasing number of IT businesses and the high complexity of delivered software, organizations require more professional software testers (Deak et al., 2013).

Although testing is an important part of software development, this activity cannot guarantee the absence of errors even with well-designed tests (Myers et al., 2011).

2.5 Gamification

Gamification is the use of game-based elements, such as mechanics, aesthetics, and game thinking, in non-game contexts with the aim of providing a learning environment, solving problems, and engaging and motivating users (de Sousa Borges, Durelli, Reis, & Isotani, 2014; Seaborn & Fels, 2015). Gamification applies the "potentially magical power of games" (Llagostera, 2012) to given problems. Deterding et al. (Deterding, Dixon, Khaled, & Nacke, 2011) suggested that the term itself was invented within the digital media industry in 2008, but the use of the actual term itself did not become popular until around 2010. It is important to note that serious games are actual games while gamified platforms are not (Felicia, 2011).

Gamification consists of game elements, including rules, dynamics, principles, features, and control mechanisms of games (Dorling & McCaffery, 2012; Muntean, 2011b, pp. 324-328).Dorling and McCaffery (Dorling & McCaffery, 2012) indicated that game elements such as points, levels, badges, achievements, progress bar, competition, feedback, virtual gifts, and leaderboards "govern a behavior through a system of incentives, feedback and rewards with a reasonable predictable outcome".

Furthermore, a single game element may not be considered "gameful", but a combination of game elements could lead to behavior within the full range of human emotions that sparks user motivation (Dorling & McCaffery, 2012). In contrast, a serious game is considered as a category of games that have a specific intention rather than only entertainment (Djaouti, Alvarez, & Jessel, 2011). Serious games could be

considered as simulation games where the aim is to “train, investigate and advertise” (Muntean, 2011b, p.324). For example, Pex4Fun (Tillmann, De Halleux, Xie, & Bishop, 2013) is a serious game introduced by Microsoft Research, which enables an environment for training computer science development skills to attendees. This web-based environment allows students to edit code in different browsers, which can be executed and analyzed in the cloud. It helps students to identify the program process and procedures. It also detects the differences between the specification and student’s program (Tillmann, Bishop, Horspool, Perelman, & Xie, 2014).

The intention of gamification for designers is to provide a game that engages with the user experience by using the game design elements (Hamari, Koivisto, & Sarsa, 2014; Huotari & Hamari, 2012). For example, Arnarsson and Jóhannesson (Arnarsson & Jóhannesson, 2015) used the gamification method to increase the motivation of developers to generate effective and efficient unit tests. The developers were rewarded after their performance was reviewed. The developers suggested that the gamified tool encouraged them to create better unit tests.

Video games have recently become popular among people, and this has been drawing the interest of researchers and practitioners in a variety of fields (Hamari et al., 2014; Sailer, Hense, Mayr, & Mandl, 2017). Video games naturally supply a great level of motivation potential (Garris, Ahlers, & Driskell, 2002; Hense & Mandl, 2014; Ryan, Rigby, & Przybylski, 2006). Any process that impacts employees could be gamified to improve their engagement or experience, from selection and recruitment to training and performance (Callan, Bauer, & Landers, 2015).

Many businesses have applied gamification in their workplaces and in the industry. For example, gamification plays an important role as a creative professional solution to a problem. Moreover, some may think that gamification involves playing and could potentially lead to distraction of employees, which may lead to poor productivity, but experts believe otherwise (B. Henessy, 2012). A wide range of desired benefits could be achieved when using gamification properly (a list of possible outcomes are summarized in table 2.1). Improving engagement and fostering motivation are some of these possible outcomes. Gamification could potentially turn

boring and mundane tasks into more attractive activities for the intended users (Muntean, 2011b, p.323). Gamification could result in increasing loyalty through engagement, participation, and motivation (Deterding et al., 2011). Moreover, gamification could be an important factor in user retention level (Deterding et al., 2011). Furthermore, gamification could be an element in developing behavioral changes in users (Llagostera, 2012). Finally, gamification could potentially improve the productivity and efficiency of users (Llagostera, 2012). This could be either in an individual's personal life (e.g. completing course work) or in a business (e.g. enhancing the quality of assigned task).

In contrast, positive outcomes of gamification in organizations could be classified into external, internal, and behavioral changes. Businesses could achieve improvement in marketing, sales, customer involvement, and employee behavior. This could also help in improving employee engagement in resolving organizational issues.

Table 2.1: Gamification Effects

Main Generic Gamification Effects	**Main Business Gamification Effects**
• Increase engagement • Increase efficiency • Behavioural change • Increase participation rate • Loyalty	• Better efficiency • Higher business profits • Higher employee retention rate • Better employee productivity • Increase employee morale

Gamification consists of applying elements of gamefulness, gameful interaction, and gameful design with a specific intention in mind (Deterding et al., 2011). In recent years, gamification has been an active research topic and a subject of much interest as a means of supporting user engagement and enhancing positive patterns in service use, such as increasing user activity, social interaction, or quality and productivity of actions (Hamari et al., 2014). As a result of gamification, people are better engaged in tasks for solving specific problems (Kim, 2015). Gamification has recently been

successfully implemented in website marketing to create loyalty, brand awareness, and effective marketing engagement (Muntean, 2011a, p.326).

Many digital games use different techniques and resources, and these techniques have elements that motivate users and challenge them to solve problems and complete different stages and tasks. In gamification approaches, these elements are not the center of the system, but have the purpose of motivating users to use the system better (Raymer, 2011). Gamification of testing activities, based upon video game principles, has become a testing strategy in both academia and the industry (Hamari et al., 2014). The aim of using gamification is mainly to increase users motivation towards tasks or to improve the quantity and quality of the output of given tasks (Hamari & Koivisto, 2015). Studies suggest that organizations are increasingly interested in using this method as part of their training and operating processes (Morschheuser, Hamari, Werder, & Abe, 2017).

Gamification could have a positive impact on motivation, learning, and enjoyment of the gamified tasks (Hamari et al., 2014). Gamification is becoming a growing field of interest in many domains, such as testing (Chen & Kim, 2012), character recognition (von Ahn, 2013), education and academia (Kapp, 2012; Roth, 2017), language translation (von Ahn, 2013), and version control (Singer & Schneider, 2012). Gamification could also be a technique to influence user behavior such as directing user behavior in a certain direction through the use of various gamification elements. For instance, Grant and Betts (Grant & Betts, 2013) studied public data from the Q&A website StackOverflow that aims at influencing users through points and badges. The system provides users the Copy Editor badge after they complete editing over 500 posts on the site and thereby aims to increase the quality of posts on the website. Their study suggests that participants that are close to 500 posts increase their editing activity to reach the 500 edit mark. Moreover, their behavior is changed after achieving 500 edits. On achieving 500 edits and acquiring the Copy Editor badge, most users stop editing posts as they do not feel the need to edit more posts.

2.5.1 Education

Gamification in education refers to the use of game elements and game thinking for educational purposes and academic development in both formal and informal aspects. Denny (Denny, 2013) examined the usefulness of badges to increase participant motivation in a study of an online multiple-choice questionnaire (MCQ). Results showed a high positive impact on the number of student contributions, without impacting their corresponding quality. Danny (Denny, 2013) stated that students showed high interest in earning badges available in their user interface.

Dominquez (Domínguez et al., 2013) applied a gamification plugin for a well-known e-learning platform (Blackboard). Results of the experiment showed both positive and negative points of using gamification. Their findings revealed students achieved better scores in practical assignments and in the overall score due to the gamified experience. However, their study also showed that those same students did not perform well in their written assignments and had a lower participation rate in class activities, even though their motivation was high (Domínguez et al., 2013).

Goehle (Goehle, 2013) experimented using video game mechanics with an online homework program called WeBWorK to increase student engagement with online mathematics homework. Goehle used few gamification elements: experience points based on student progress through individual tasks, levels for reaching specific milestones, and a progress bar, achievements, and rewards for extra points. Although student progress was satisfactory, the authors were unable to conclusively decide whether gamification had any positive effect on student performance in the course (Goehle, 2013; Seaborn & Fels, 2015).

Li (Li, Grossman, & Fitzmaurice, 2012) implemented a gamified tutorial system (GamiCAD) to allow new AutoCAD users to study AutoCAD through a gamified process. Gamified elements used in this system were missions, scoring, levels, time limit, mini games, and rewards. Results indicated better engagement and performance in the users due to gamification (Li et al., 2012; Seaborn & Fels, 2015).

2.5.2 Crowdsourcing

Liu (Liu, Alexandrova, & Nakajima, 2011) developed a mobile crowdsourcing application called UbiAsk that allows users to create, deliver, run micro task activities, and share the results on a few social media platforms. Gamification elements were used in the application to encourage participants to use the application. The researchers discovered good participation and response by the users. However, it was unclear how the results compared to the non-gamified version of the application. In addition, results indicated gamification use did not reach statistical significance (Liu et al., 2011).

Witt (Witt, Scheiner, & Robra-Bissantz, 2011) examined what motivates users who participate in an online idea competition and what negative results the game mechanism could have on motivation. Game points and social points were used in the systems. Game points were awarded for completion of actions and social points for engagement in social behaviors (rating, commenting on user ideas, etc.). The questionnaire's results regarding the game mechanics fell into the "Neither Agree nor Disagree" category. The authors speculated that the system design, together with the leaderboard presentation, provided these mixed results (Seaborn & Fels, 2015).

2.5.3 Research study

Rapp, Marcengo (Rapp, Marcengo, Console, & Simeoni, 2012) implemented a gamified application to help raise the number of valid responses from users in a field study. WantEat was the application's name, which was selected for a study conducted around a cheese festival. Users received points based on actions such as providing feedback and tasting cheese with other contributors. Their progress was tracked on a leaderboard, achievements were unlocked, and additional rewards were earned by users. Results showed users rated the ease of use, usefulness, and high engagement of the application. Nevertheless, participants were not motivated to communicate more with new people or leave comments on other people's reviews. In addition, users continuously used the application even when they reached the maximum point level (Seaborn & Fels, 2015). Their conclusion was that it is unclear whether users were motivated to use the application due to the game element components (Seaborn & Fels, 2015).

2.5.4 Software Development

It is important to keep developers and testers motivated. In the last decade, gamification has been adopted by researchers and practitioners as a means to enable active participation and engagement during formal tasks, including the software development community (Huotari & Hamari, 2012). Recently, there has been a high level of interest in the use of game elements in non-gaming software applications with the aim of improving engagement and motivation (da Rocha Seixas, Gomes, & de Melo Filho, 2016; Fitz-Walter, Tjondronegoro, & Wyeth, 2012).

Various studies have indicated positive results of using gamification in software development both in the educational and industry contexts. For instance Passos et al. (Passos, Medeiros, Neto, & Clua, 2011) presented a case study with an agile software team, in which participants were awarded medals upon completion of assigned tasks, levels of test code coverage, and number of iterations accomplished within a specific time period. Results suggested that the game element significantly helped in improving developer motivation levels, and also helped the company in monitoring and controlling the development process.

2.6 Linking software testing and gamification

Gamification could be a solution to address current software testing issues and has been mostly used to improve motivation, engagement, and performance rate by applying game elements in a non-game context. To identify the current gaps and the required research initiatives, a detailed literature review was conducted and is presented here.

Pedreira et al. (Pedreira, García, Brisaboa, & Piattini, 2015) performed a systematic mapping to apply gamification in the software development process. Authors performed an analysis of 29 related research articles published until 2014. Their goal was to find answers to questions related to gamified software engineering processes and related research methods. Since software testing is part of the software development lifecycle, testing was also included in this search domain.

Mäntylä and Smolander (Mäntylä & Smolander, 2016) also performed a literature review on the gamification of software testing. The authors used previous review study by Pedreira et al. (Pedreira et al., 2015) to perform forward and backward snowballing. They verified 20 items and categorized results based on the testing types, individual roles, applied game elements, empirical evidence reports, related studies that presented support constructs to gamification, and challenges related to gamifying software testing context. Moreover, Fraser (Fraser, 2017b, pp. 2-7) reported issues in software testing and proposed gamification as a solution to address these issues.

Finally, (de Jesus, Ferrari, de Paula Porto, & Fabbri, 2018) conducted a review on gamification and software testing using the systematic mapping (SM) (Petersen, Feldt, Mujtaba, & Mattsson, 2008) method. They reviewed studies that compose the baseline for analysis and discussion relevant to this research area. Through these studies, they identified application context, testing levels and processes, testing techniques, gamification elements, and goals. Systematic mapping helps in providing an overview of the selected investigated area and assists in identifying research gaps and opportunities. For this purpose, the authors identified the following points through the literature review:

1. Areas that gamification has been applied.
2. Reasons for gamifying the context.
3. List of game elements used in the selected context.
4. Identifying testing techniques, processes and levels.

In contrast, a scoping review helped us in mapping the key concepts in the research area and to underpin these concepts to main sources and types of evidence accessible. This is a useful method specially in complex areas of study or those that have not been reviewed comprehensively (Mays, Roberts, & Popay, 2001). This method assists in providing in-depth analysis of accessible literatures depending on the aim of the review. Arksey and O'Malley (Arksey & O'Malley, 2005) stated following reasons for using scoping study method when undertaking literature review in the domain of study:

1. To study the extent, nature, and range of accessible research study on the area of interest. Although this may not result in describing research findings in detail, it assists in mapping fields of study.
2. To determine the need for undertaking a complete systematic review in which mapping the current literature could help in determining whether a full systematic review is feasible or if the literature undertaken is relevant. To identify the potential costs of undertaking such a full systematic review.
3. To help with summarizing and disseminating research findings.
4. To determine the research gaps in the current literature to aid planning and commissioning of future studies.

It is therefore important to determine the set of steps for conducting a scoping study. The first step is to identify the relevant research question. For instance, in our case, it helps in identifying from the existing literature the current position of the effectiveness of gamification in software testing. The second step would be to identify relevant studies and to filter unnecessary studies. This assists in identifying primary studies in the interest domain. For this purpose, different strategies such as electronic databases, reference lists, existing conferences, and journals were used. In total, 85 studies were retrieved. However, after filtering these studies by applying search string to assist in identifying relevant studies, the following papers were selected from the database (Anderson, Nash, & McCauley, 2015; Bell, Sheth, & Kaiser, 2011; Clegg, Rojas, & Fraser, 2017; Dubois & Tamburrelli, 2013; Fu & Clarke, 2016; García, Pedreira, Piattini, Cerdeira-Pena, & Penabad, 2017; Laurent et al., 2017; Reza Meimandi Parizi, 2016; Passos et al., 2011; Sheth, Bell, & Kaiser, 2012). Finally, charting the data and summarizing the results helped to complete this review process. To obtain the required information from each study, the researcher elaborated the summary of each identified article by considering the role of gamification as a supportive method in the context.

The following are some of key game elements that were reported from the relevant studies on gamification and software testing:

A. Achievement (Bell et al., 2011; Dubois & Tamburrelli, 2013; Passos et al., 2011): Achievement aims at defining objectives to be achieved. This could be used to stimulate more complex work.
B. Points(Anderson et al., 2015; Bell et al., 2011; Clegg et al., 2017; Dal Sasso, Mocci, Lanza, & Mastrodicasa, 2017; García et al., 2017; Liechti, Pasquier, & Reis, 2017; Reza Meimandi Parizi, 2016; Passos et al., 2011; Rojas & Fraser, 2016; Sheth et al., 2012): Points aim at quantifying the progress. This could be used to raise encouragement level.
C. Badge(Anderson et al., 2015; Dal Sasso et al., 2017; Fu & Clarke, 2016; García et al., 2017; Liechti et al., 2017; Passos et al., 2011): The goal of using badges is to represent achievements. This could be used to increase encouragement and motivation levels.
D. Leader board (Bell et al., 2011; Dal Sasso et al., 2017; Dubois & Tamburrelli, 2013; Fu & Clarke, 2016; García et al., 2017; Reza Meimandi Parizi, 2016): The goal of using a leaderboard is to provide a public display of performance. This could be an element to increase competition among players.
E. Virtual good(Dal Sasso et al., 2017): This aims at proving assets with either virtual or real goods. This could enhance the motivation and engagement level of players.
F. Level(Clegg et al., 2017; Dal Sasso et al., 2017; Fu & Clarke, 2016; García et al., 2017; Passos et al., 2011; Rojas & Fraser, 2016; Rojas, White, Clegg, & Fraser, 2017): The main goal of using level is to define steps to achieve. This helps in enhancing engagement, competition and motivation level.
G. Avatar (Dal Sasso et al., 2017; Reza Meimandi Parizi, 2016; Passos et al., 2011): The goal of using avatar is to provide a visual representation of characters. This assists in boosting engagement and motivation level.
H. Social graph(García et al., 2017): This aims at representing a social network within the game context. This could help in stimulating collaboration, engagement, and motivation.

Some gamifications goals and their aims are listed below:

A. Increase adoption(Rojas & Fraser, 2016): Aims at increasing the adoption of software testing.
B. Increase awareness(Rojas & Fraser, 2016): Its goal is to boost users' awareness related to their performance.
C. Increase engagement(Anderson et al., 2015; Bell et al., 2011; Clegg et al., 2017; Dubois & Tamburrelli, 2013; Fu & Clarke, 2016; García et al., 2017; Laurent et al., 2017; R. M. Parizi, 2016; Passos et al., 2011; Rojas & Fraser, 2016): Aims at increasing the engagement level of software testers in testing activities.
D. Improved skills(Clegg et al., 2017; Dal Sasso et al., 2017; García et al., 2017; R. M. Parizi, 2016; Rojas & Fraser, 2016; Rojas et al., 2017; Sheth et al., 2012): Aims at improving participants' level of knowledge, performance, and efficiency together with other related skills.
E. Increase enjoyment(Anderson et al., 2015; Bell et al., 2011; Clegg et al., 2017; Liechti et al., 2017; Rojas & Fraser, 2016; Rojas et al., 2017; Sheth et al., 2012): Aims at increasing enjoyment level of participants while performing testing activities.
F. Encourage testing habits(Bell et al., 2011; Sheth et al., 2012): Aims at encouraging participants to participate in testing activities until this becomes a habit.
G. Better fixing process(Dal Sasso et al., 2017): Aims at reducing the participants' effort in the fixing process.
H. Increase motivation(Anderson et al., 2015; Dal Sasso et al., 2017; Dubois & Tamburrelli, 2013; Fu & Clarke, 2016; Liechti et al., 2017; R. M. Parizi, 2016; Rojas & Fraser, 2016; Sheth et al., 2012): Aims at increasing the motivation level among testers to perform/learn software testing.

It is worth knowing that unit testing was the main testing level used in the selected studies (Clegg et al., 2017; Laurent et al., 2017; R. M. Parizi, 2016; Rojas & Fraser, 2016; Rojas et al., 2017). For instance, Rojas et al. (Rojas et al., 2017) used EvoSuite,

Randoop and Major automated tools with CODE DEFENDERS. EvoSuite and Randoop generate unit tests using JUnit tool for Java classes. In contrast, Major is a mutation testing tool used in their study. The listed tools were used to assist the single-player mode in the game context and also to run comparison between mutants and the created tests.

2.6.1 Extracted relevant studies

The following is an overview of some of the selected studies in which gamification was applied.

Anderson et al. (Anderson et al., 2015) proposed Learn2Mine, which is a cloud-based learning environment. This learning environment has been designed for instructors to be able to teach programming in data science courses and aims to provide students with a useful and more enjoyable environment that assists them navigate through the learning course. The system has been designed such that lessons are introduced as evolving sub problems, which students should constantly solve to achieve the lesson's goal. Although software testing is not the primary focus of the course, the sub problems introduced mirror unit testing concepts. This helps students to learn and use unit testing techniques in their code to accomplish the goals and earn rewards after concluding the lesson. The system allows participants to receive immediate feedback by grading student submissions, and the students also have the option to re-submit their answers. The main game elements used in the system are points, badges, and leaderboard, which helps in increasing the enjoyment, engagement, and motivation level of participants. The authors used a survey to capture student feedback about the designed system, and all questions received positive responses with a multi-submission system with the highest positive result. The authors also performed other experiments to compare the results from two groups, with one group having access to the gamified learning system and the other to the non-gamified version. Results indicated that the performance of the gamified group was significantly higher in the overall task completion. This supported the authors' hypotheses that gamification could be an element to help increase the completion rate. With positive feedback from the students who used the gamified version, Learn2Mine was

considered a successful learning environment for teaching data science course that could somehow be extended to software testing.

Parizi (R. M. Parizi, 2016, pp.1-8) raised the importance of methods of traceability between code artifacts and tests. The researcher proposed the conversion of "existing post-mortem recovery of trace links to proactive construction of traceable software systems with highly engaged human factors". For this purpose, they created a gamified conceptual framework called GamiTracify to increase the level of engagement and motivation among developers in traceability tasks aiming to improve the quality of tracing links. In this environment, developers must maintain or perform traceability of information while developing the test. This helps to reduce the post-verification of recovered links at later stages. The authors compared the developed tool with SCOTH, a peer approach to assess how the developed tool could impact the ability to obtain more accurate traceability information. For this purpose, precision and recall metrics were used to quantify the accuracy level. Results indicated that using GamiTracify resulted in higher recall and precision metrics. Results also supported the fact that gamification could have positive impact on motivation level to achieve improved accuracy. In this study, structural testing technique and unit testing level were covered. The authors used game elements such as points, leaderboard, and avatar to increase the engagement and motivation level of test developers.

Laurent et al. (Laurent et al., 2017) proposed a gamified crowdsourcing platform to help collect information related to the detection process. The aim of this system is to label equivalent mutants and to evaluate multiple parameters for the detection process. Gamification was applied to increase the engagement level of users. Moreover, game elements selected for this platform are points and leaderboard. The focus of the gamified prototype is in software testing activity, explicitly fault-based testing in the context of unit testing. However, the authors did not present any evaluation results for the proposed gamified platform.

Shet et al. (Sheth et al., 2012) is a poster that discusses HALO environment, which was used to evaluate student performance. The main aim of the HALO system was to transform software testing to a more fun and engaging experience and to provide an

engaging development environment. The pilot study discussed by the authors was conducted with undergraduate students. Students had the option to use a HALO prototype for their programming assignments. Results indicated that students who used this prototype performed better compared to those who did not.

Bell et al. (Bell et al., 2011) used "secret ninja" as a methodology and proposed to combine it with HALO (Highly Addictive, sociaLly Optimized) software engineering, which uses MMORPG (Massively Multiplayer Online Role Playing Game) design. This has the potential to be integrated to an IDE as a plugin. The main goal of combining HALO and secret ninja was to increase the motivation level of students to perform testing on their own code. The other main purpose of using HALO is to improve the engagement level throughout the software development process and to increase the student learning outcomes. With increasing the engagement and enjoyment level, the authors hope to achieve higher productivity and satisfaction by users. Points, levels, achievements, and leaderboard are the main game elements used in this gamified environment. The authors aim to discover the impact of a gamified environment on student performance.

CODE DEFENDERS is another gamified system that was used to teach software testing based on mutation testing principles. Furthermore, this system can also be applied for a crowdsourcing approach and education evaluation framework for educators. CODE DEFENDERS game was presented in the following four studies (Clegg et al., 2017; Rojas & Fraser, 2016; Rojas et al., 2017). The main purpose of using gamification in all four studies was to increase engagement, motivation, and enjoyment levels. Points, level, and leaderboard are some of the game elements used in these studies.

2.6.2 Findings overview

For the literature review purpose, the researcher considered both educational and industrial contexts in which gamification was applied related to software testing domain. The literature review indicates that most of the approaches can be used in both educational and industrial contexts. Studies indicate that points and levels were the main used game elements. The literature review helped in better understanding of

commonly used game elements and the way they could be applied in the gamified context. Moreover, these studies helped the researcher to identify the main gamification goals. The commonly used goals were to improve skills, motivation, and engagement levels.

Gamification is still a new trend in software engineering, and it is important to monitor its trend in software testing domain. The related literature review studies also helped the research to identify the key elements requiring investigation to achieve a better gamified platform for software testing purpose.

2.7 Chapter summary

The notion of gamification can be injected into software testing as testing is often tedious, monotonous, and boring (Shah & Harrold, 2010) and is also considered time-consuming and difficult activity in terms of focus (Alrmuny, 2014). Insufficient testing may be harmful, and many issues with recent software failures have been due to the lack of effective testing strategies being employed in practice.

Gamification can be implemented to transform difficult testing tasks into an engaging and motivating experience that taps into the gamer's creativity with the possibility of bringing about the detection of defects that may not have been detected in the traditional manner (Rojas & Fraser, 2016). Thus, gamification of software testing tends to resolution of issues ranging from tester engagement, industry issues, and testing limitations.

Gamification is still a new trend in software engineering, and it is important to monitor this trend in the software testing domain. Related literature review studies also helped to identify the key elements to be investigated to achieve a better gamified platform for software testing purpose. Furthermore, the main aims of using gamification have been discussed, which help the researcher to map each game element to a specific gamification goal. The researcher identified that although there has been a number of related works, most related research studies did not focus on the design aspect of the gamified platform. Thus, the researcher further investigated and identified the essential game elements that are closely applicable in the software testing

field. This mapping will help to choose proper game elements when designing the gamified software testing platform. Furthermore, in this thesis, we first focus on designing a gamified software testing platform that uses the relevant game elements, and in further iterations, through feedback obtained from focus group sessions with software developers and software testers, the final gamified software testing platform will be presented. In contrast, most papers related to gamification and software testing purely concentrated on the motivation and engagement levels. However, the performance of software testers performing testing tasks needs to be evaluated in more detail.

Chapter 3

Design consideration through focus group discussion

3.1 Introduction

In chapter one, the importance of software in our daily life, together with the motivation and objectives of this research, were highlighted. It identified the significance of this research with existing issues, and a brief overview of the various approaches to solve the stated problems was presented.

The previous chapter presented a comprehensive review of software engineering and development, software testing, gamification, and the reasons behind gamifying software testing. Furthermore, it reviewed the use of the gamification technique in software testing. This chapter presents the design consideration of gamifying software testing system through results obtained from multiple focus group sessions with software developers and testers.

This chapter covers the study of identifying the game-based design considerations that will assist software testers and businesses in achieving more efficient software testing, while providing an engaging and rewarding environment for software testers.

Some of the material in this chapter has previously appeared in the following publication:

Navid Memar, Aneesh Krishna, David A McMeekin and Tele Tan (2017). Gamification of Information System Testing-Design Consideration through focus group discussion. The 26th International Conference on Information Systems Development (ISD 2017), Larnaca, Cyprus.

For this purpose, an early stage web-based game engine prototype will be introduced. The developed prototype will be used to evaluate the significance of the game elements introduced into the software testing process. These evaluations are conducted using a focus group comprising software developers and testers.

3.2 Proposed methodology

This section explains the prototype development methods and the way game elements will be used in the chosen software testing application. However, it is first essential to choose a relative methodology to obtain optimal results. Therefore, the design science method based on the regulative cycle framework proposed by Wieringa (R. Wieringa, 2009) has been selected. This methodology places emphasis on the linkage of knowledge and practice to design for utility whilst acquiring better scientific knowledge and results. Design science works toward creating and improving things that serve human purposes (R. Wieringa, 2009). The regulative cycle could be defined as a general structure for the process of rational problem solving (March, 1994; Simon, 1955). This could be done by analyzing the current situation and current change goals, proposing possible changes to achieve identified goals, evaluating proposed changes and selecting one, implementing the selected change, and then starting all the phases over again (R. Wieringa, 2009). Figure 3.1 represents the regulative cycle suggested by Wieringa (R. Wieringa, 2009) and Figure 3.2 contains an explanation on how practical problems will be addressed in different stages. In Figure 3.2, we have structured the design science project as a series of nested problems in which the main problem is the practical problem. For this purpose, regulative cycle assists in solving practical problems. The regulative cycle (Van Strien, 1997; R. J. Wieringa, 1996) begins with identifying a practical problem, continues with solution design specifications and then validating the solution designs, and finally implements the chosen design. Moreover, the outcome of this design may then be evaluated, which could be the start of the new process in the regulative cycle (R. Wieringa, 2009).

In this research, the concept of the regulative cycle has been adopted as the conceptual framework for the logic of practical problem solving. In the next section, we will discuss the steps of the regulative cycle in relation to our research topic.

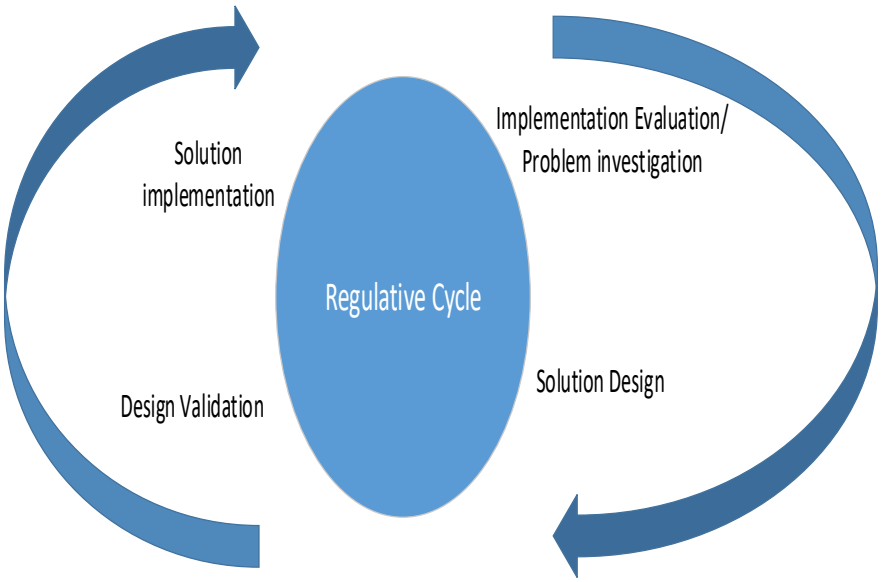

Figure 3.1: The regulative cycle (R. Wieringa, 2009)

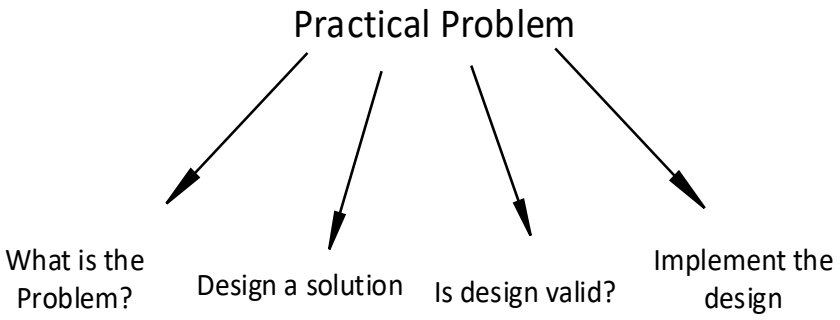

Figure 3.2: Breaking down practical problems into sub-problems

The first step is to investigate the problems that involve knowledge and an understanding of existing problems. A detailed explanation of the existing problems was given in the literature review section. Gamification has been used as the main method to be studied and analyzed as the potentially beneficial element to solve some current issues of software testing. By applying game elements to the proposed application, the possible advantages and disadvantages of using this method in the software testing area may be gauged. Gamification may be a solution to increase tester engagement in software testing, as it can potentially remedy the high degree of repetition and the ensuing boredom of a tester in the testing phase. Thus, to make this important part of software engineering more attractive, gamification may be a method to improve the quality of testing by making testing a more stimulating experience for the human software tester by reducing boredom.

3.2.1 Mapping game elements with the design elements

The key element to increase the motivation of players in serious game design is the storyline (Whyte, Smyth, & Scherf, 2015). The storyline is a helpful element to increase motivation for learning when learning opportunities are combined directly with the story content (Baranowski, Buday, Thompson, & Baranowski, 2008; Garris et al., 2002). Therefore, it was decided to apply this element to increase the level of motivation, engagement, and quality.

The second game element is goal-directed learning around targeted skills (Whyte et al., 2015). This practice involves the combination of primary end goals and intermediate incremental goals that provide challenges and progress (Whyte et al., 2015). Both medium and long-term goals have been applied to the current system to provide motivation and goals for the players.

Thirdly, feedback and awards play a very crucial role in shaping behavior in serious games as players are often motivated to work continuously to achieve certain goals (Whyte et al., 2015). Multiple feedback engines were injected into the current system to provide real-time as well as post-exercise feedback to players. Increasing levels of difficulty and individuation also play an important

role in serious games. Providing challenging but achievable goals in a supportive environment helps to increase competence for certain skills (Przybylski, Rigby, & Ryan, 2010). Serious games should endeavor to attain a challenging but achievable level of difficulty for players. Thus, it follows that these games should not be so difficult that players are discouraged from attempting to complete the game, but also not so easy that players are demotivated from learning new skills (Whyte et al., 2015). To achieve this, different badges and levels were implemented that players could unlock after obtaining points upon successful task completions.

Provision of choice is another element that increases motivation and enjoyment in serious games (Ryan et al., 2006). Allowing individuals to have choice over some aspects of the game environment helps players to maintain a sense of autonomy and control over their learning experience (Przybylski et al., 2010). In the current system, players have the ability to accept or reject a given task after it is assigned. Upon rejecting the task, it is automatically sent to the next available player for review.

3.2.2 Design consideration for the gamified software testing prototype

The aim is to study and understand if these game core elements and design elements may help to significantly enhance the performance of the software testers and the quality of the software defect detection and reporting. Figure 3.3 shows the class diagram for the initial gamified platform. The class diagram consists of three significant components of class, attributes, and methods. In Figure 3.3, the tester and administrator (subclasses) inherit all the attributes and methods of the super class (user). Moreover, multiplicities are clearly stated with numbers such as 0..1 or a specific number.

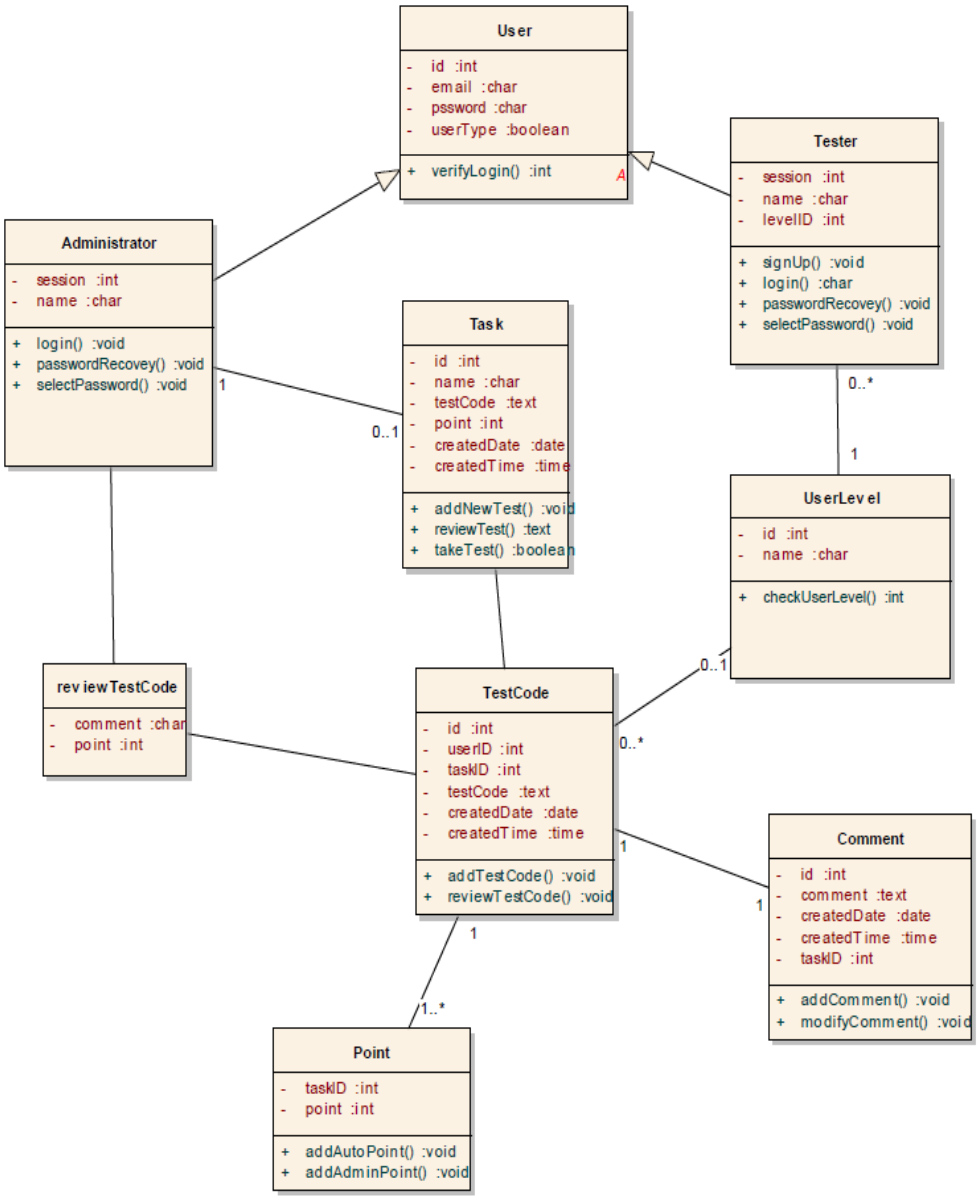

Figure 3.3: Class diagram of the pilot gamified system

Figures 3.4 and 3.5 show the interactions among the elements of the system. This is used to identify, clarify, and organize the system requirements. Use cases are scenarios showing how a user can use the system. A system can have multiple use cases. Use cases are textual artifacts, and a use case diagram can be drawn using three measure components. The first component is an actor who interacts with the system. Anything outside of the system boundaries are referred to as

actors. The second component, i.e., measurement, represents the functionality or services provided by the system. The last component of use case diagram is called relation, which represents the relationship between the actor and the system.

A system can have two types of actors, namely primary and secondary actors. Primary actors are those who can initiate a use case, and secondary actors are those who can have involvement in the use case but they cannot initiate the use case. For instance, in Figure 3.4, Admin is able to perform a set of actions such as register a tester, activate testers, manage the page, assign tasks, manage testers, and access query task results and query tasks and updates. In addition, in Figure 3.5, the tester can perform a list of actions such as sign up, sign in, sign out, view unlock tasks, update visibility status, or check his/her points in the system. In Figure 3.5, a direct connection is drawn from the tester to sign up, sign in, sign out, review task, view unlocked task, update visibility status, and check points since the tester is initiating these seven use cases. In Figure 3.5, the system acts as a secondary actor since it cannot initiate any use cases. By going one level deeper into the components of the use case diagram, one use case can be connected with another use case by an extend or an include relation. An extend use case shows an optional behavior, meaning that the based use case is complete by itself and the extended use case is an option. In contrast, include relation indicates the mandatory behavior, meaning that the based use case is incomplete by itself and it needs the included use case to fulfil a particular requirement.

For instance, in Figure 3.5, consider that an unlocked task use case is part of the view unlocked task use case, meaning that the view unlocked task is not complete by itself and it has to include the other use case. However, for the extend relation, we can refer to the update visibility status use case, which is a complete use case by itself. It is not mandatory for everyone to communicate to other online users or to request group testing. As can be seen in Figure 3.5, the request group testing and communicate to online user are the extends of the update visibility status use case.

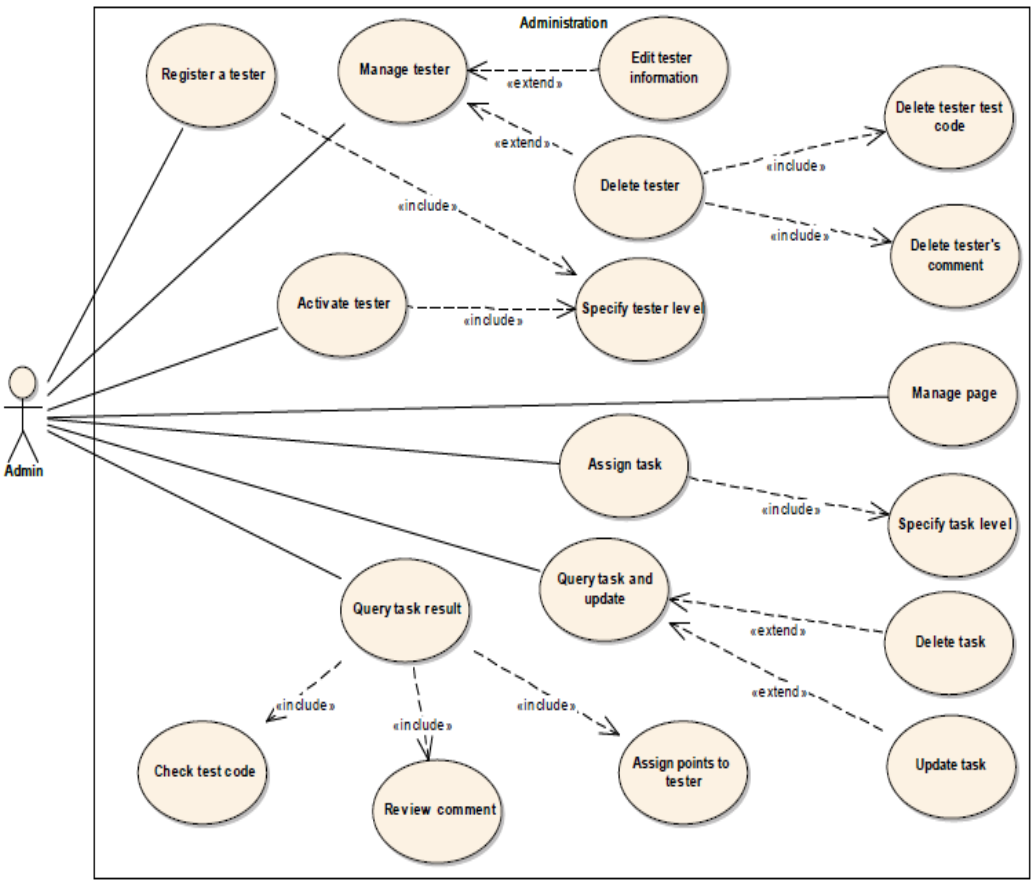

Figure 3.4: Admin use case of the pilot gamified system

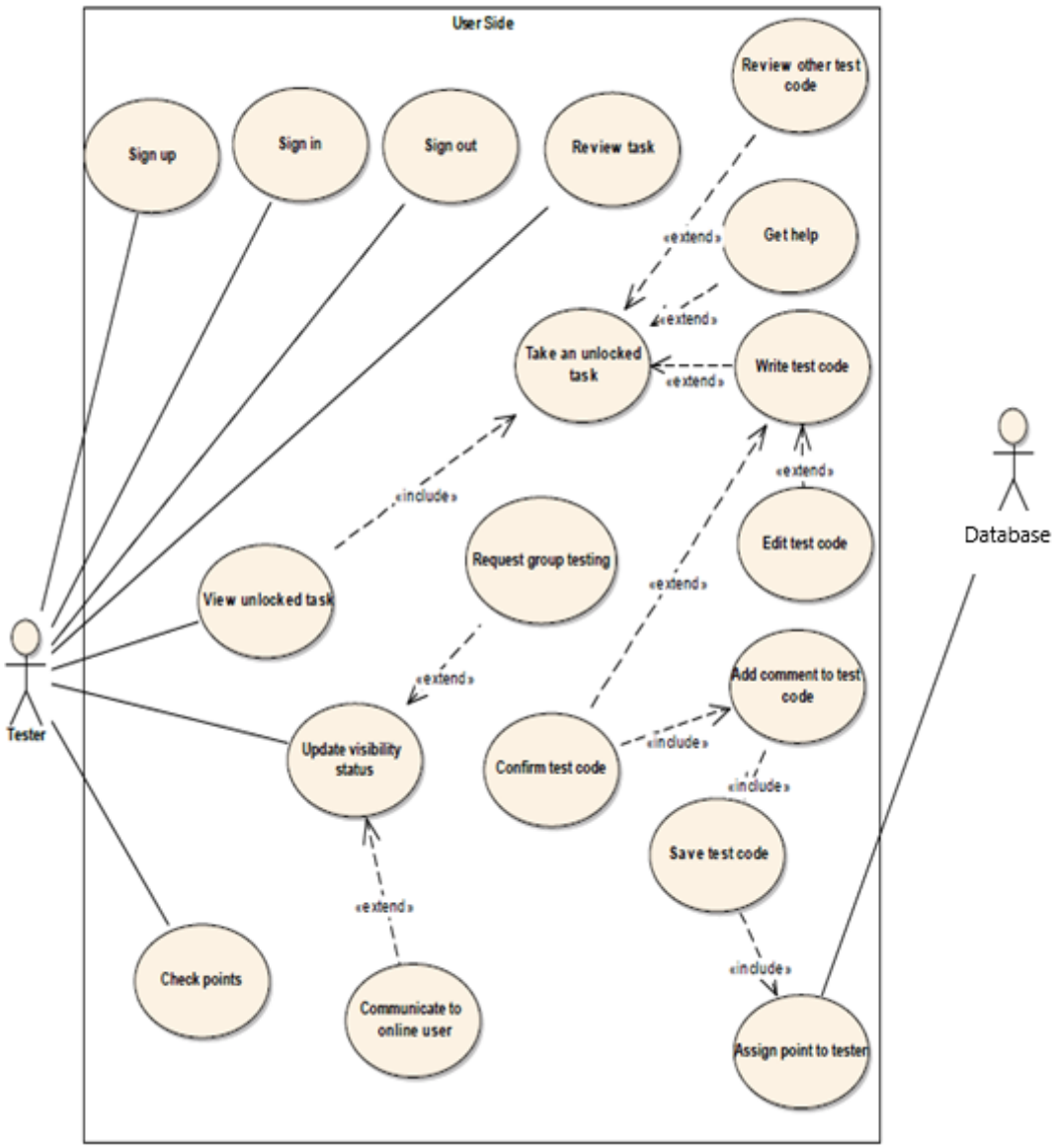

Figure 3.5: Tester use case of the pilot gamified system

Figures 3.6 and 3.7 describe the dynamic aspects of the system through activity diagrams. This helps to demonstrate the flow from one activity to another. In activity diagrams, a dot represents the starting point of the diagram, activities are denoted by round cornered rectangles, and conditions are denoted by diamond shapes. A diamond indicates a decision and determines which one of the two paths the program will take. For example, in Figure 3.6, admin enters login details,

which are then validated. Upon successful validation, the admin will navigate to the home page and will be able to review test codes and connect to the web server. Furthermore, upon successful connection, the admin will be able to check the code in the local software and approve the test code. In case the approval of the test code fails, the admin can provide the tester with additional comments to describe the reasons for failure. In contrast, if the test code is approved, the admin may assign the required points to the tester.

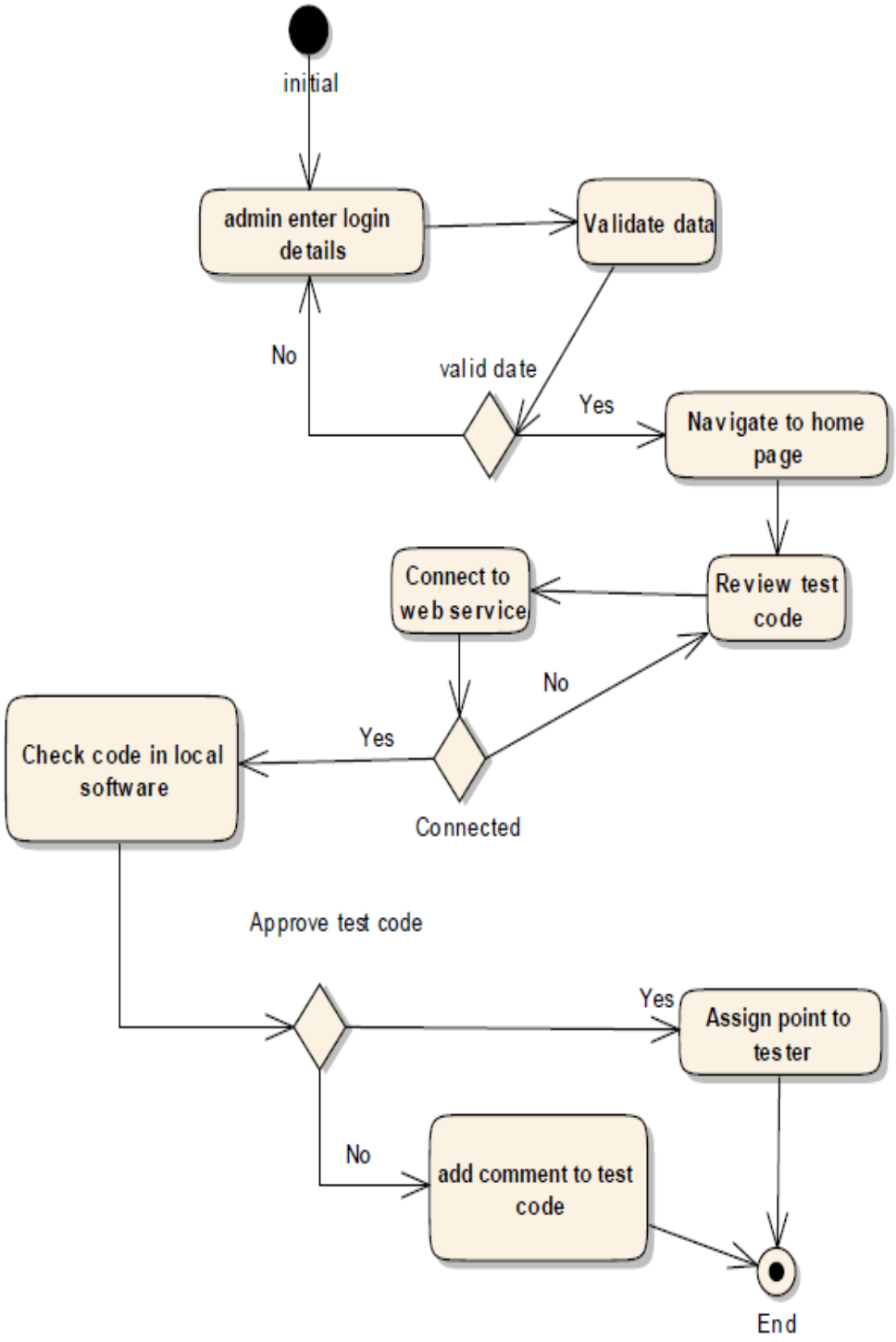

Figure 3.6: Admin activity diagram of the pilot gamified system

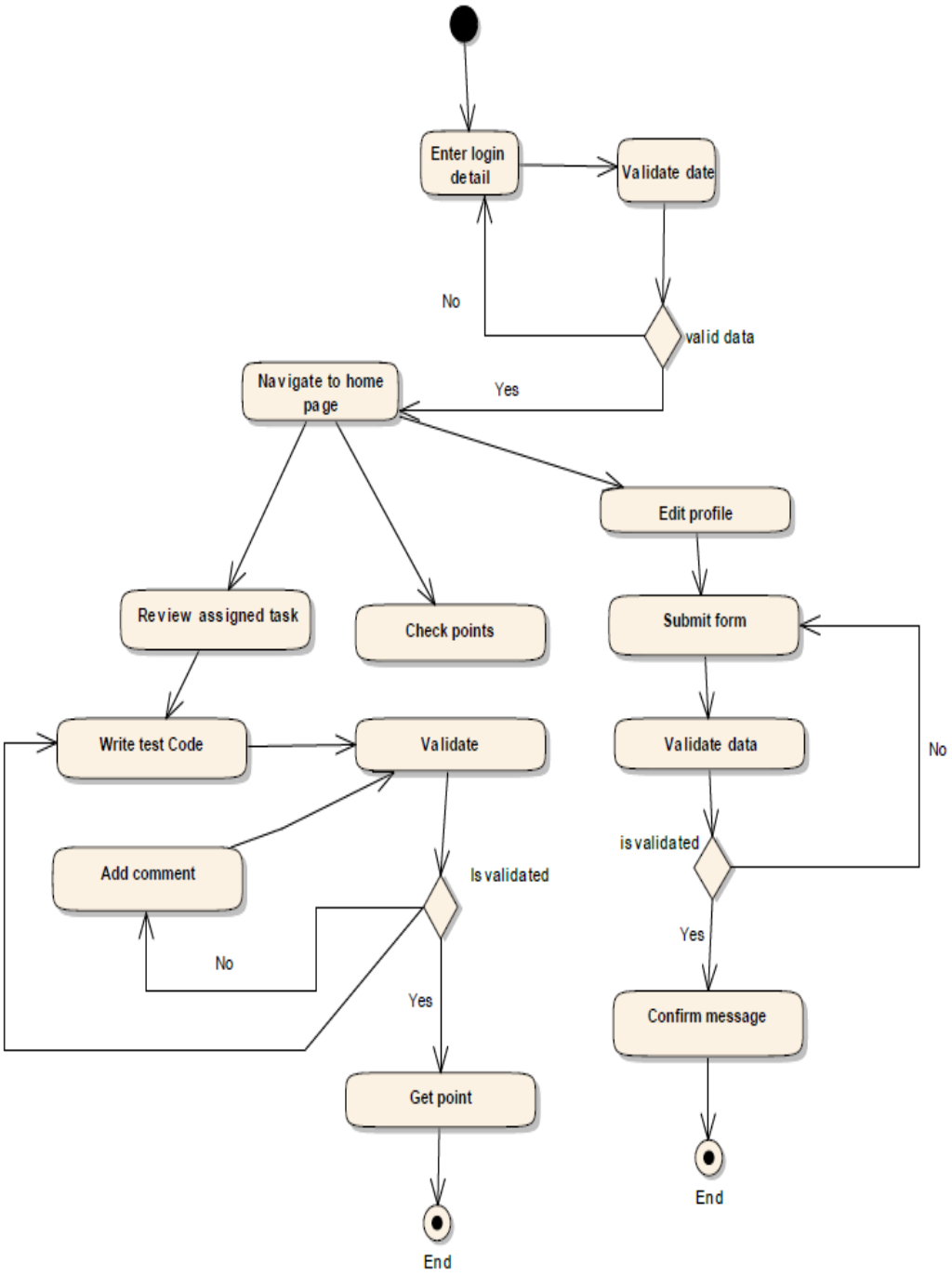

Figure 3.7: User activity diagram of the pilot gamified system

3.2.3 User interfaces

The literature review assisted the researcher to study and identify the core game elements that suit the needs of this study. In this section, the initial prototype that was used to gather useful information from the participants will be discussed. Based on the literature review, the importance of using game elements in designing a gamification user interface was discovered. The main purpose was to increase the engagement and satisfaction of software testers. The literature review helped to adopt core principles of serious game design suggested by Whyte et al. (Whyte et al., 2015), as listed below, to identify game elements required to design the gamified platform.

A. A story line that helps increase the level of motivation, engagement, and quality.

B. Goal-directed learning around targeted skills, which provides challenges and progress.

C. Feedback and awards, which play an important role in shaping behavior in serious games and motivate the players to work continuously to achieve certain goals.

D. Badges and levels, which help in motivating players to attain a challenging while maintaining an achievable level of difficulty for testers.

E. Provision of choice, allowing individuals to have choice over some aspects of the game environment.

The next step of this study is to design the gamified information system testing prototype based on the verified game elements in the literature review. Figure 3.8 shows the home page of the gamified software testing platform. In this page brief information about gamification is provided to the users. In addition, a user can sign in or sign up as a tester into this platform.

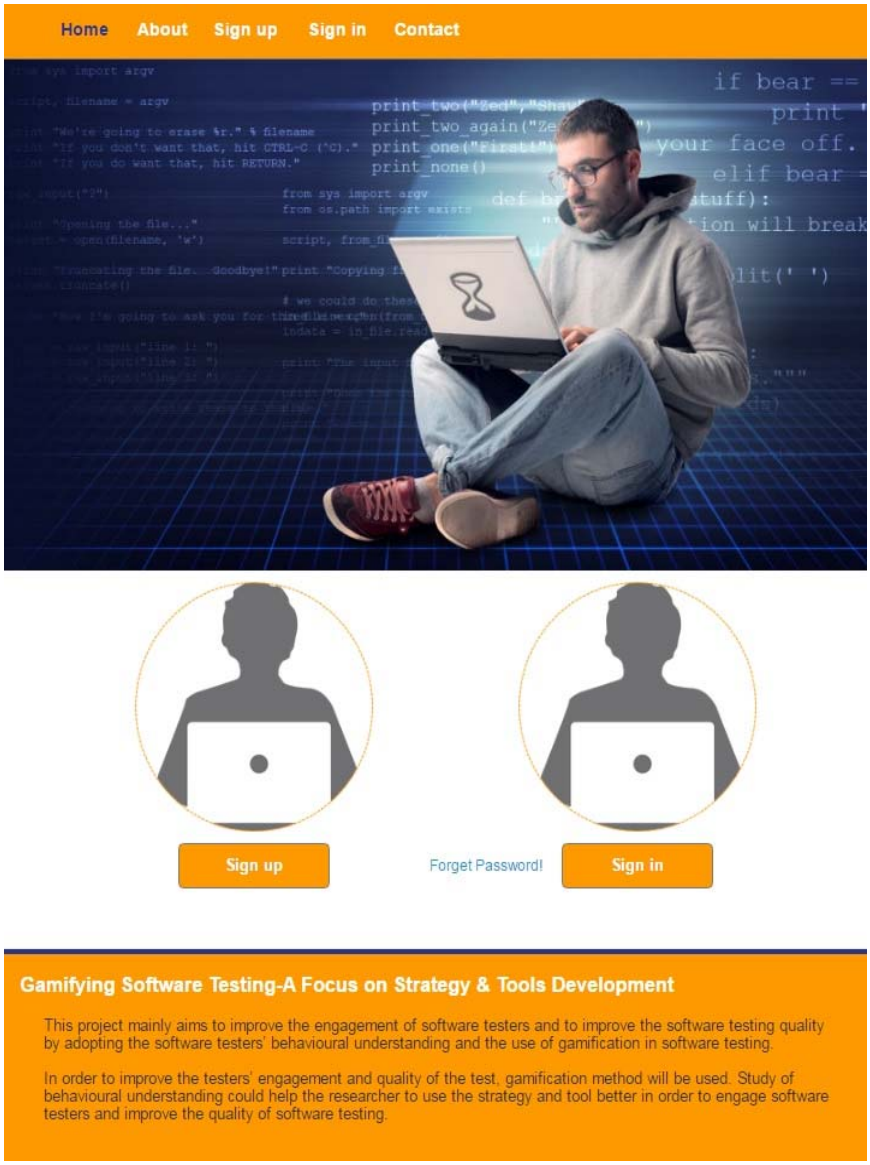

Figure 3.8: Gamified software testing platform home page

Figure 3.9 shows the software tester's home page. In this page, tester can perform testing activity individually or can invite other testers to participate in a common testing activity challenge. Testers can also communicate with each other using group testing chat option that enables them to share knowledge if required. Further, testers can check the performances rate of other testers, which helps them compare their performance with other testers and increase their performance level.

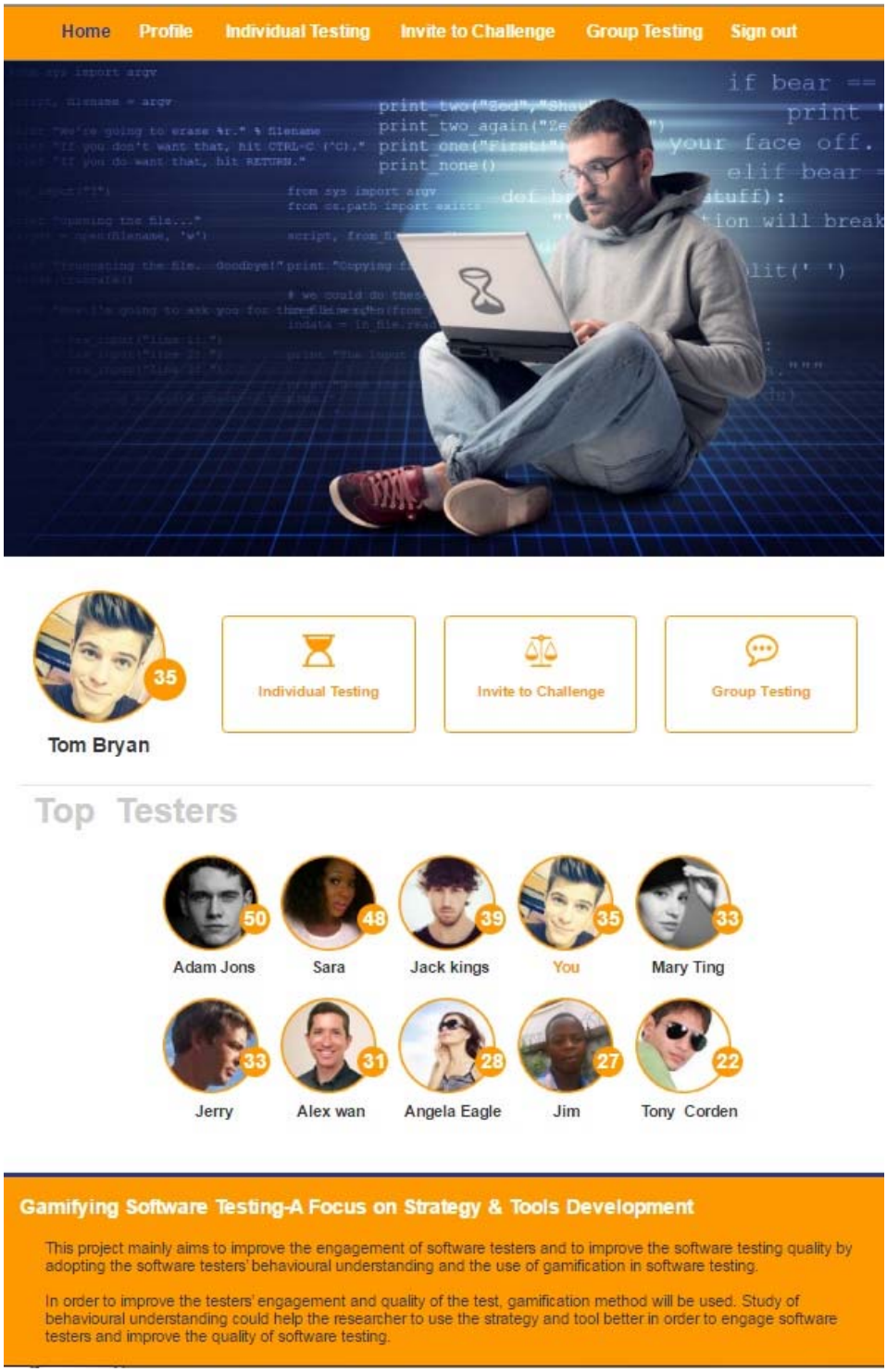

Figure 3.9: Software tester's home page

Figure 3.10 presents the list of tasks assigned to an individual tester. This table indicates the assigned task code, points assigned to each task, date that each task was assigned to the tester, and the option to initiate the testing activity.

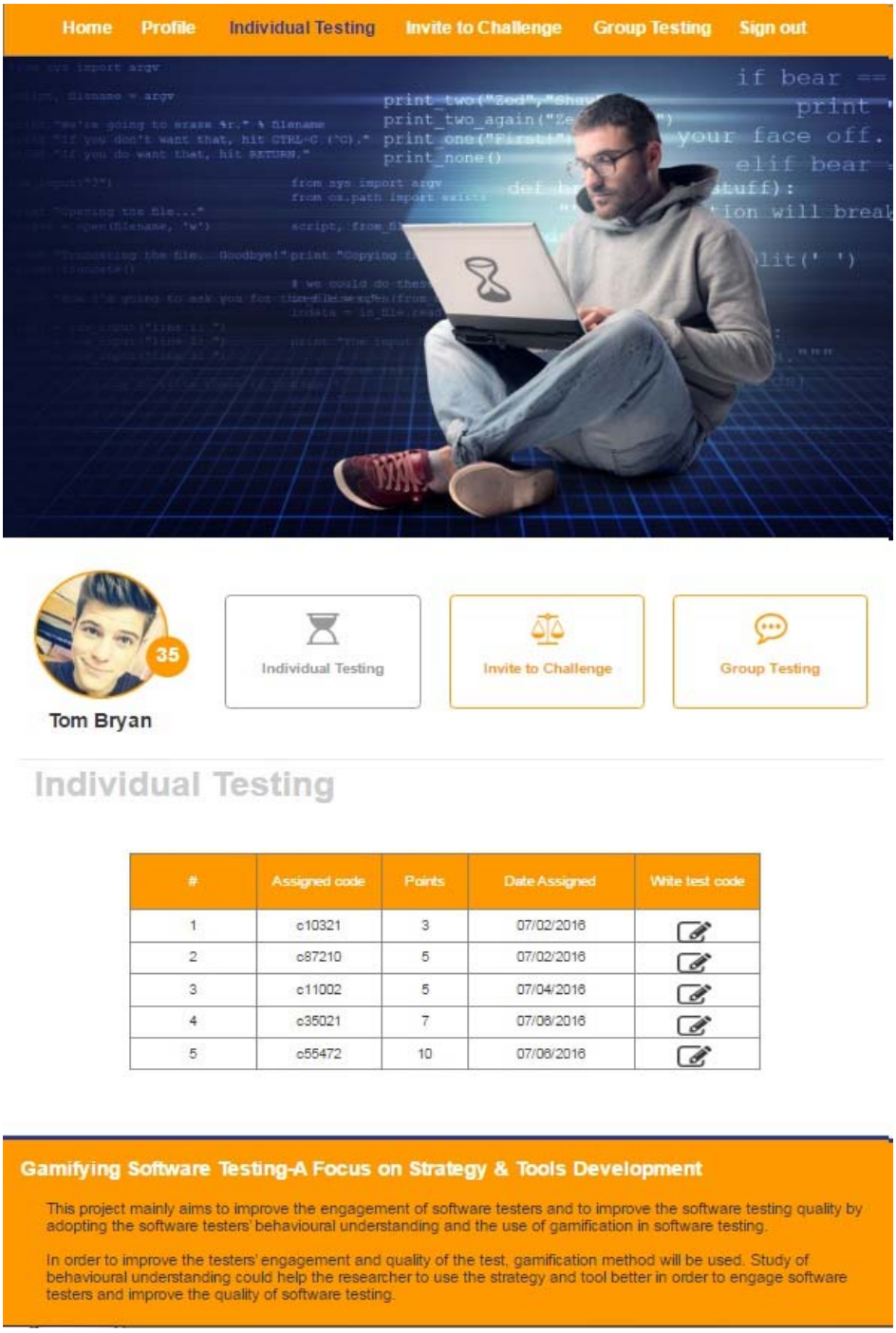

Figure 3.10: Software tester's testing tasks

Figure 3.11 shows the page in which the tester should write the test code for the given task. For this purpose, the tester can check the production code on the left hand side and can write the test code for the given production code in the right-side box. Moreover, a timer is provided to capture the testing period, which will be stored together with the written test codes when completed by the testers.

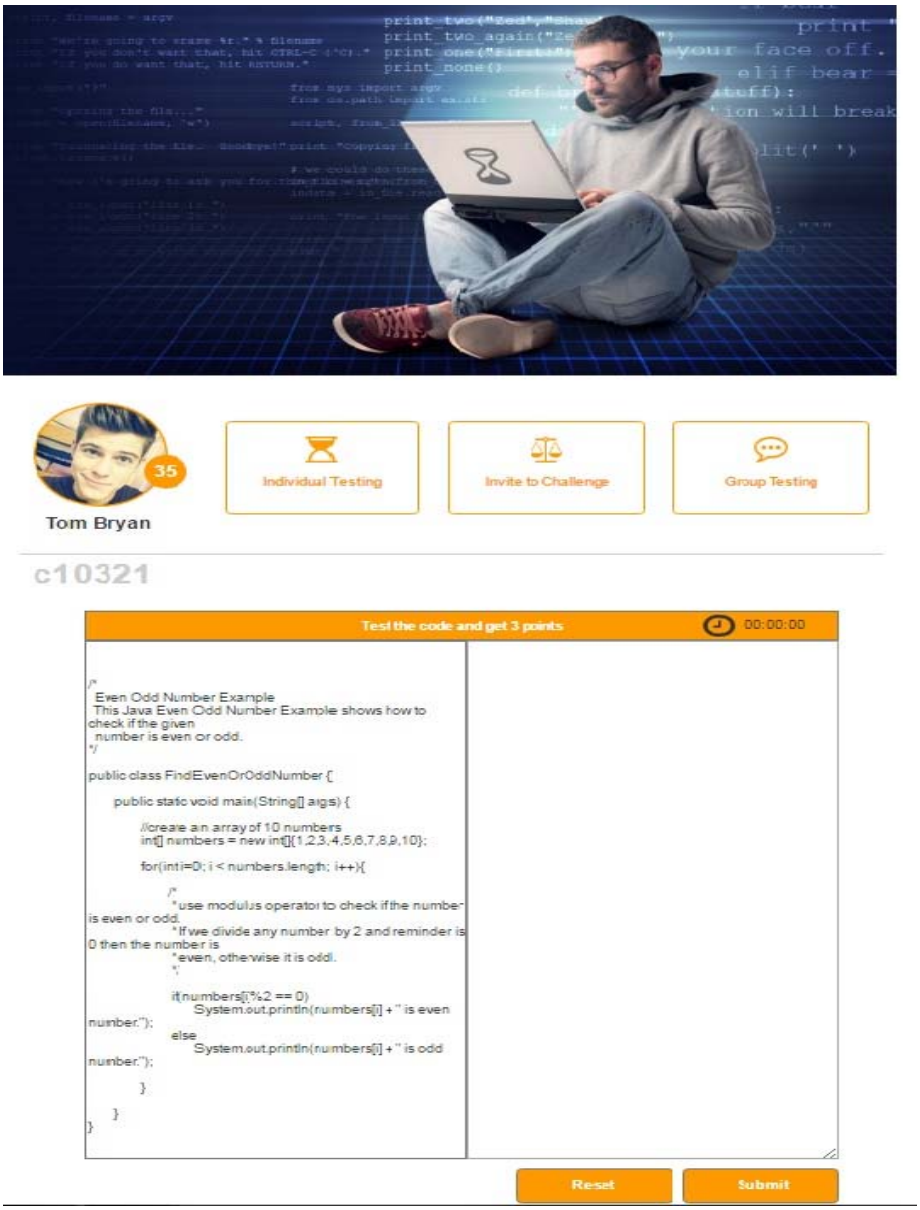

Figure 3.11: Software tester's testing page

Figure 3.12 presents the steps for the tester to invite one or more testers to participate in a testing challenge for the common testing task.

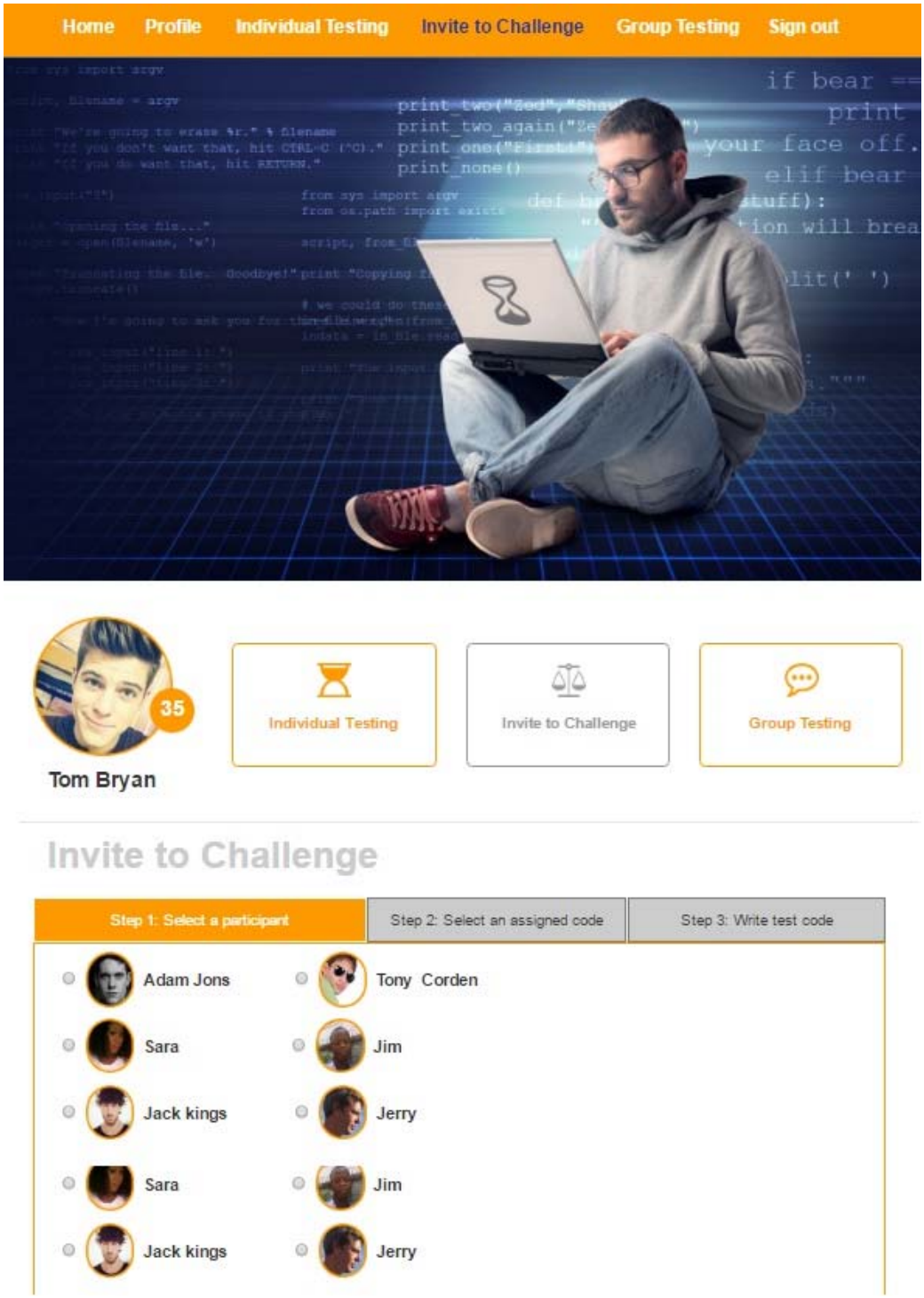

Figure 3.12: Steps to invite other testers to challenge

3.2.4 Participants recruitment process

The initial aim is to study the essential game-based design elements that could help software testers and businesses to achieve the outcome. For this purpose, various focus group sessions with software testers and developers were conducted to understand and identify the key gamification elements in an effort to give software testers a more interactive tool and environment to test the production system. During these sessions, an early stage web-based game engine prototype was used to demonstrate the chosen game elements identified through detailed literature reviews conducted by the researcher. In total, 20 students attended the sessions, and out of them, 70% had both a Software Development and Software Testing background, 25% of them had a Software Development background, and 5% had only a Software Testing background. All participants were briefed on the aim of this research study and on the web-based gamified prototype. The participants were also given the chance to explore the designed gamified platform.

This study has the approval of Curtin University's Human Research Ethics Committee (approval number RDSE-76-15).

3.3 Results and analysis

Various focus group sessions with the developers and testers helped the researcher ascertain key factors required to optimize a system that gamifies the software testing process.

In the following sections, the main themes and findings are described that emerged from the focus group sessions. Written questionnaires and recorded focus group discussions were used in gathering this information. The process of analysis began by reviewing the list of findings in all focus group sessions and the recorded focus group discussions. Key factors were identified and were coded with the number of similar responses associated with each factor to provide a precise result. Those numbers were then converted into graphs to represent the findings in a simplified method.

These findings can be summarized under the following headings:

3.3.1 Gamification encourages software testers

In the questionnaire, there was a question asking whether gamification could be an element to rectify the issues of disengagement for software testers. To support their answers further, participants also explained their point of view by a detailed explanation.

Most participants stated clearly that gamification may be an element to improve the testing efficiency and to rectify the issues in software testing. Two responses are listed below:

- "Absolutely! However, the environment should be designed in such a way that is very intuitive to use, provides results, and gives testers something to work towards."
- "Yes, gamification makes people get more interested and competitive."

In contrast, another participant argued that although gamification could avoid the repetitive nature of software testing, other issues might arise. The participant supported his statement with a comment stating that real gifts or learning new skills may be beneficial for the testers, but testers would not test the code simply for recreation in terms of the gamification aspect.

Another question asked participants for advice on steps and methods required to increase the level of tester encouragement. Some of the responses are listed in below:

- "There should be clear steps of getting awards."
- "Having game levels and points for good job is rewarding. Failure to do good things should be penalized as well, i.e., a concept of available lives. The points should be redeemable for more lives or extension of time. If the platform has some funding, it will be great to receive some real gifts."
- "Having a page much like StackOverFlow where users can post questions and seek advice without being judged on their code/question."

- "A point-based system is a good approach to encourage users. Users are intently attracted to virtual reputation. For instance, StackOverFlow utilizes this methodology. However, for the point-based system to actually work in motivating users to engage with the system, the system needs to be well-known or have a high reputation amongst testers."
- "Competition is a good way to motivate people. Players like to see improvement and comparison makes it more interesting."

3.3.2 Requirement and design documentation is a vital information required for testers

To identify the important information required for software testers to perform a testing task efficiently, participants were given related questions, and their responses were classified into four main factors (requirement & design documentation, tools, time, and knowledge).

Figure 3.13 represents the findings obtained from the various focus group sessions. From the figure, it is clear that requirements and design documentation, and knowledge are the key factors. The results from the focus group sessions suggest that system and task knowledge are essential to achieve efficient testing performance. The responses were coded and categorized into four sections, as shown in Figure 3.13.

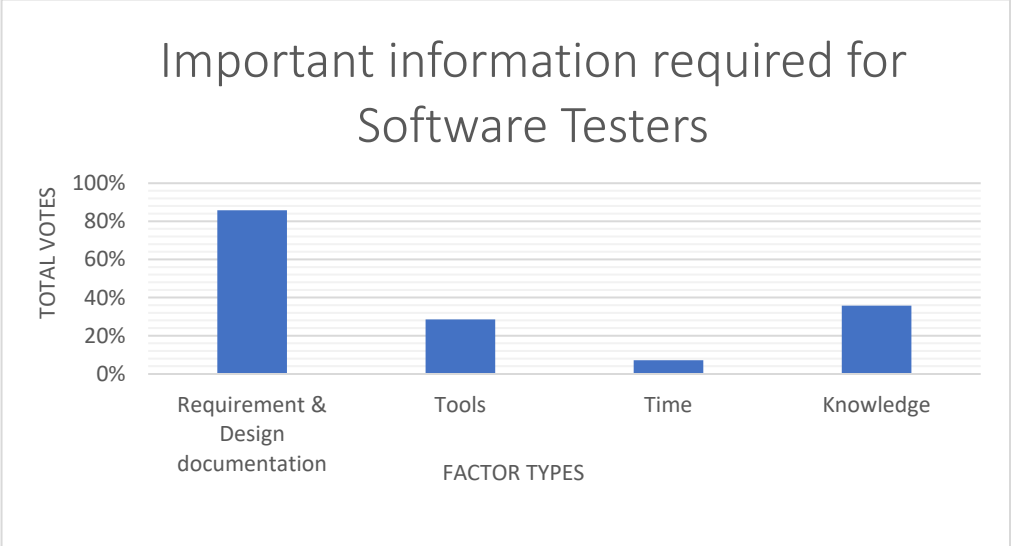

Figure 3.13: Factors required for software testers

3.3.3 Number of testers relate to the quality of testing

The results suggest that participants agreed on the fact that having more testers can increase the quality of testing. Some of the responses were as follows:

- "Theoretically, the more testers there are, the more bugs are going to be identified. Two brains are better than one."
- "Different individuals will approach problems in different ways and are likely to find different bugs. No matter how good a single tester may be, having multiple testers is always better."
- "A higher number of testers means a higher number of results. You get more results from different types of users as well as different levels of expertise."
- "The more the testers, the better the quality."
- "The higher the number of testers, the lesser the likelihood that there will be missing or incorrect results. Ideally, each tester should ensure 100% coverage of the code, but it is easy to overlook edge cases. If there are multiple

testers, this will confirm that the results are more likely correct if the results are consistent between testers."

- "It is relevant to have more than one tester because several points of view can help to improve the quality of the code."

Some also argued that although more testers could improve the testing performance, it could also complicate the testing process. The following is an example to support this idea:

- "In some cases, the number of testers positively influences the result of the testing. For instance, a greater number of testers can lead to a greater ability to recognize defects in the system. However, in other cases, a greater number of testers can complicate the testing process."

3.3.4 Essential game elements

Figure 3.14 presents motivational factors stated by participants. Under such circumstances, it is clear that all five categories are essential for the proposed system. Game elements, such as points, levels, real gifts, progressions, character development, provision of choice, and feedback, were the main factors stated by participants, which after precise consideration, have been coded into the main five design elements suggested by Whyte (Whyte et al., 2015). Among these 5 categories, participants voted the most for feedback and rewards-based learning element. Kay Berkling (Berkling & Thomas, 2013) also stated that students mostly look for elements such as attractive design, intuitive control, task overview, feedback, communication section, comparison tool, leaderboard, points, badges, and levels in a gamified platform.

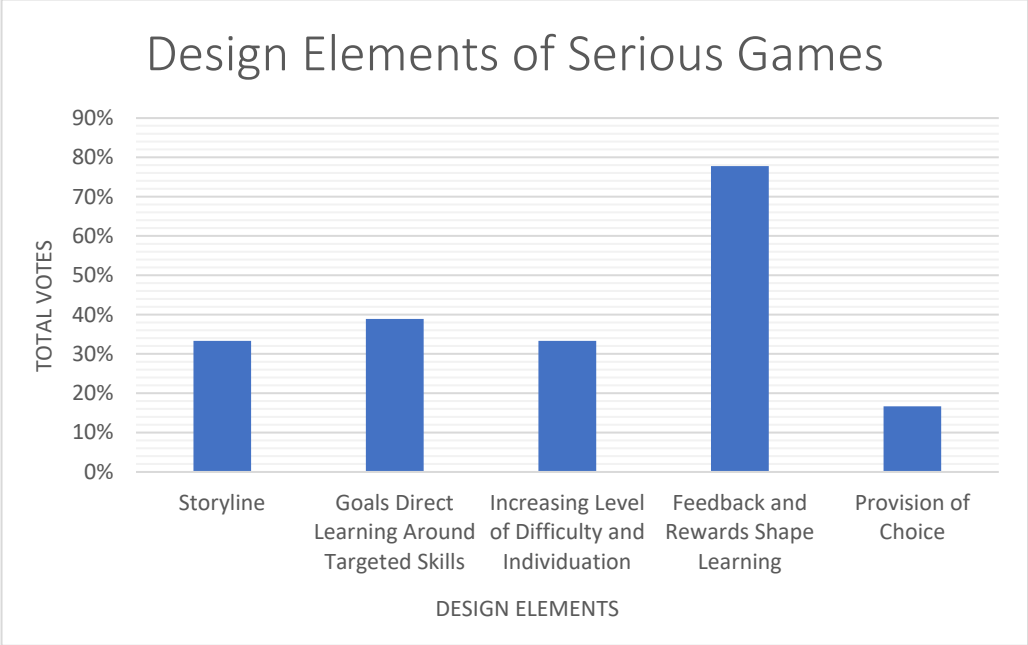

Figure 3.14: Design elements and motivation factors

We further asked participants the following question: "Do you play games? Please explain more about what you like about the game (for example, stories, character environment, levels, feedback, rewards, difficulty, and choices)". The responses indicated that levels, rewards, storyline, graphics, and challenges are the main factors that influence them when playing games. Some of the responses are listed below for quick reference:

- "I do not usually play games. I quire enjoy the gamification aspect of the app 'Swarm' where you get points for visiting places. It is fun to compete and to beat friends at the game. It means you have bragging rights over them for a while."
- "Yes. I enjoy the competitiveness and being the best in the game."
- "Yes. I like to be challenged with difficulty."
- "Progression in game is important."

Furthermore, participants were asked the following question: “Based on the current system, what would you like to include in the system so that it would be interesting and fun?”. Participants pointed that factors such as milestones, ranks, levels, and categories of assigned test codes are essential for the gamified software testing platform. Some of the responses are listed below for quick reference:

- “Certain amount of code segments tested, certain amount of critical/serious etc bugs found, certain amount of points collected, certain streak of time achieved ie at least one segment tested every day for 30 days etc, certain amounts of suggestions/help/messages communicated to other testers.”
- “Passing levels through points should be great. To start is easy to gain point but when the person make more progress should be more difficult to have the points and to pass to another level.”
- “There needs to be a way to normalize the effort done by each tester. If a code owner supplies simple and crap code, arguably the tester shouldn’t be more awarded than one who found a single problem in a very complex and well written system. Perhaps like ho GitHub used to do, the best way is to award based on the number of days they have visited (or in a row even) as ‘streaks’. 10 days in a row = level up or something. Using time would just encourage shoddy and incomplete work.”
- “Milestones could just be titles given: e.g. Beta tester, master tester, tester king, etc.”
- “Categorizing users based on their acquired points may act as a motivator for some users. For instance, a user with 1000+ points may be considered an expert while a user with <100 points may be considered a novice. Also, ranking users to a specific percentile would encourage competition and thus can act as a motivator.”

They further stated that the current platform requires further improvement by considering elements such as vouchers, badges, reward points, scoreboard, titles,

and physical rewards when designing the finalized gamified software testing platform.

To identify the importance of another key element for use in the final gamified platform, the participants were asked the following question: “Do you think that it is necessary to have a communication section for testers to share their results? How would you suggest that be added to the current system?”. Results suggest that majority of participants agreed that a communication section for testers could be beneficial. However, one participant stated that this may result in increasing the collusion level, which may affect the outcome. Some of the responses are listed below for quick reference:

- “Absolutely! A communication section opens a healthy environment for testers to discuss and promote knowledge. In addition, it can also enable healthy competition between testers to debate certain techniques and process.”
- “Yes. Helps to have more knowledge by exchanging their experience.”
- “Yes, good communication between the code owner and the tester is crucial to make sure that both parties understand what is expected of them.”

3.4 Recommendations

Results obtained from various focus group sessions with developers and software testers assisted the researcher to identify the main elements to be used in a gamified software testing platform. In the results section, we described the main themes and findings obtained from the conducted focus group sessions.

The following findings emerged from these focus group sessions:

- Gamification encourages software testers
- Number of testers is related to the testing quality
- Essential game elements for gamifying software testing platform
- Requirements and design documentation is vital information required for software testers

Furthermore, we have listed below the important information required for software testers based on the priority of these factors:

1. Requirements and design documentation
2. Knowledge
3. Tools
4. Time

Results suggested that majority of participants agreed that requirement and design documentation is the main information required for software testers. System and task knowledge help software testers to exhibit better testing performance.

We also identified the essential design elements to design the gamified software testing platform. These main 5 design elements suggested by Whyte (Whyte et al., 2015) were used and were coded with the number of similar responses associated with each factor. These factors are listed below based on their priority for designing a successful gamified testing platform based on the results obtained from the various focus group sessions.

1. Feedback and rewards-based learning
2. Goals that direct learning around targeted skills
3. Storyline
4. Increasing level of difficulty and individuation
5. Provision of choice
6. This study helped to identify the importance of gamification as a tool to encourage software testers. This also helped the researcher to identify and validate the essential game elements to optimize a system that gamifies the software testing process. These findings also helped the researcher to identify the required information for software testers to achieve a more efficient testing performance.

3.5 Chapter summary

In this chapter, a brief discussion about the core design of serious game elements was provided. Thus, the initial proposed gamified software testing platform was developed. The pilot platform was then evaluated by software developers and software testers through multiple focus group sessions. The proposed game elements were identified through the results obtained from focus groups session with the participants. The results of this study help in understanding the vital information and elements required for software testers for designing a successful gamified software testing platform. The next step is to design a gamified platform for testers based on the identified game elements. An evaluation of the final platform may help determine whether gamification can be a factor that increases the quality of testing and the engagement of testers. In the next chapter, the final gamified software testing platform will be presented. Moreover, a detailed discussion on the results obtained from the evaluation of the final platform will be provided.

Chapter 4

Qualitative validation through focus group discussion

4.1 Introduction

In the previous chapter, we first examined the core design surrounding the serious game elements, and a web-based game engine prototype was implemented with a focus group session to identify the core elements that facilitate the further enhancement of the gamified software testing platform.

In this chapter, the development and evaluation of the final gamified software testing platform will be presented. The aim of this study is to understand if the developed features within the platform can help fulfill the objectives of the gamified software testing platform. Moreover, this study helps to investigate the impact of gamification on software testing by applying the chosen and verified game elements proposed in the previous chapter.

Some of the material in this chapter has previously appeared in the following publication:

Navid Memar, Aneesh Krishna, David A McMeekin and Tele Tan (2018). Gamifying Information System Testing-Qualitative Validation through Focus Group Discussion. The 27th International Conference on Information Systems Development (ISD 2018), Lund, Sweden.

4.2 Methodology

This section describes the chosen method for the evaluation of a gamified software testing platform. To realize this purpose, a number of activities were performed, which can be categorized into the following stages.

Stage 1 - Conducting a literature review to ascertain the broad elements and categories used in serious games, followed by applying the findings into the initial prototype (presented in Chapter 3).
Stage 2 - Conducting focus group sessions with software testers and developers to validate both the initial prototype and chosen game elements (Presented in Chapter 3).
Stage 3 - Implementing the final product with previous findings and validating the final product by organizing focus group sessions with software developers and testers.

In this chapter, we will focus on the development and evaluation of the final gamified platform by conducting focus group sessions with software developers and testers. This will help to ascertain the usefulness of this tool in increasing tester motivation, and provide them with an engaging and rewarding environment.

4.2.1 Summary of proposed changes to the initial prototype

In the previous chapter, we discussed why and how initial game elements were selected in detail. The main reason for choosing the right game elements is to increase the engagement and satisfaction of the software testers, which can lead to improved experience with the testing activities. In the following section, feedback received from the previous focus group sessions will be discussed. The feedback helped the researcher to build new sets of prototypes that meet software tester and software developer expectations. The feedback is summarized below:

1. To increase the level of tester encouragement, it is essential to set clear steps for participants to achieve awards.

2. The system should be designed to increase the competition level among testers. Each player should be able to check their progress level and compare their progress with other testers.
3. It is essential to provide a complete requirement and design document to the software testers as part of the testing task.
4. Although the number of testers could improve the testing performance, in some cases, this could complicate the testing process.
5. Essential game elements such as feedback and rewards system, goal-directed learning around targeted skills, storyline, increasing level of difficulty and individuation, and provision of choice should be included in the gamified software testing platform.
6. Factors such as milestones, ranks, levels, and categories of assigned production codes are essential for the gamified software testing platform.

This feedback helped the researcher make significant changes to the initial prototype. Along with the detailed literature review conducted on a gamified platform, the feedback helped to bring about the following enhancements to the initial prototype.

1. A detailed information package about the production code (a sample of information package is available in appendix ….) was made available to the software testers to help them obtain a better understanding of the objectives of the testing activity.
2. By enhancing the interface design with categorizing and labeling every element, testers could have a better knowledge for identifying the steps required to accomplish the given task.
3. Different levels of difficulty were introduced in the gamified platform. Each tester should achieve a certain point to be able to unlock different levels (junior, intermediate, senior, and voucher/physical reward). Moreover, testers could compare their performance with other testers from their home page).

4. It was decided to remove group testing and instead allow the code owner to assign testing tasks to multiple available testers. This allows capturing of wider results, which could potentially lead to more accurate bug detection reports.
5. Essential game elements such as feedback and rewards system, goal-directed learning around targeted skills, storyline, increasing level of difficulty and individuation, and provision of choice were included to improve the design in the updated prototype.
6. In the updated design, code owners could choose to select the category of testing tasks (e.g. black box testing, white box testing). This also allows testers to have a better understanding of the given testing task. Moreover, milestones, ranks, and levels were categorized separately to allow testers to obtain better feedback on their software testing performance.

4.2.2 Final product implementation and validation

In the previous chapter, the proposed idea of how gamification could encourage software testers and motivational factors were also discussed and identified. The next step is to design the finalized gamified information system testing platform based on the verified game elements for software testers. Figure 4.1 shows the tester's home page. To increase the motivation and engagement levels of software testers, identified game elements such as points, badges (difficulty in levels), real gifts, feedback, comparison, and provision of choice were added into the system. To unlock each badge, testers need to obtain certain points by the review team. Furthermore, comparison is the next element included in the platform for testers to boost their performance by reviewing the performance of other testers. Moreover, provision of choice helps the testers to control over requests to accept or decline any assigned testing tasks after reviewing each request. Real gifts is another motivational factor identified in the previous chapter, and has been included for those testers who receive enough points to unlock the final badge.

Following the implementation of the finalized gamified software testing platform, several focus group sessions were conducted to evaluate the developed gamified platform. The focus group discussions were conducted to ascertain the usefulness of this method to increase tester motivation, and provide them with an engaging and rewarding environment.

Figure 4.2 shows the code owner's home page. In this page, the code owner can assign white box or black box testing tasks to any number of available testers. The code owner can also review the results of completed assigned testing tasks submitted by testers and provide feedback and required points to each software tester. Figure 4.3 shows the required steps and information to assign a testing task to each software tester. In order for the code owner to assign a testing task to software testers, he/she must provide information about the task, specify the time required for completing the testing task, and provide information about the points given for each identified level of bugs. Furthermore, the code owner should attach the software code and submit the task to software testers. Once software testers complete the testing tasks, the code owner can review the results and provide feedback and points based on the performance of software testers. Figure 4.4 shows how the code owner can review the results and check the status of each assigned testing task.

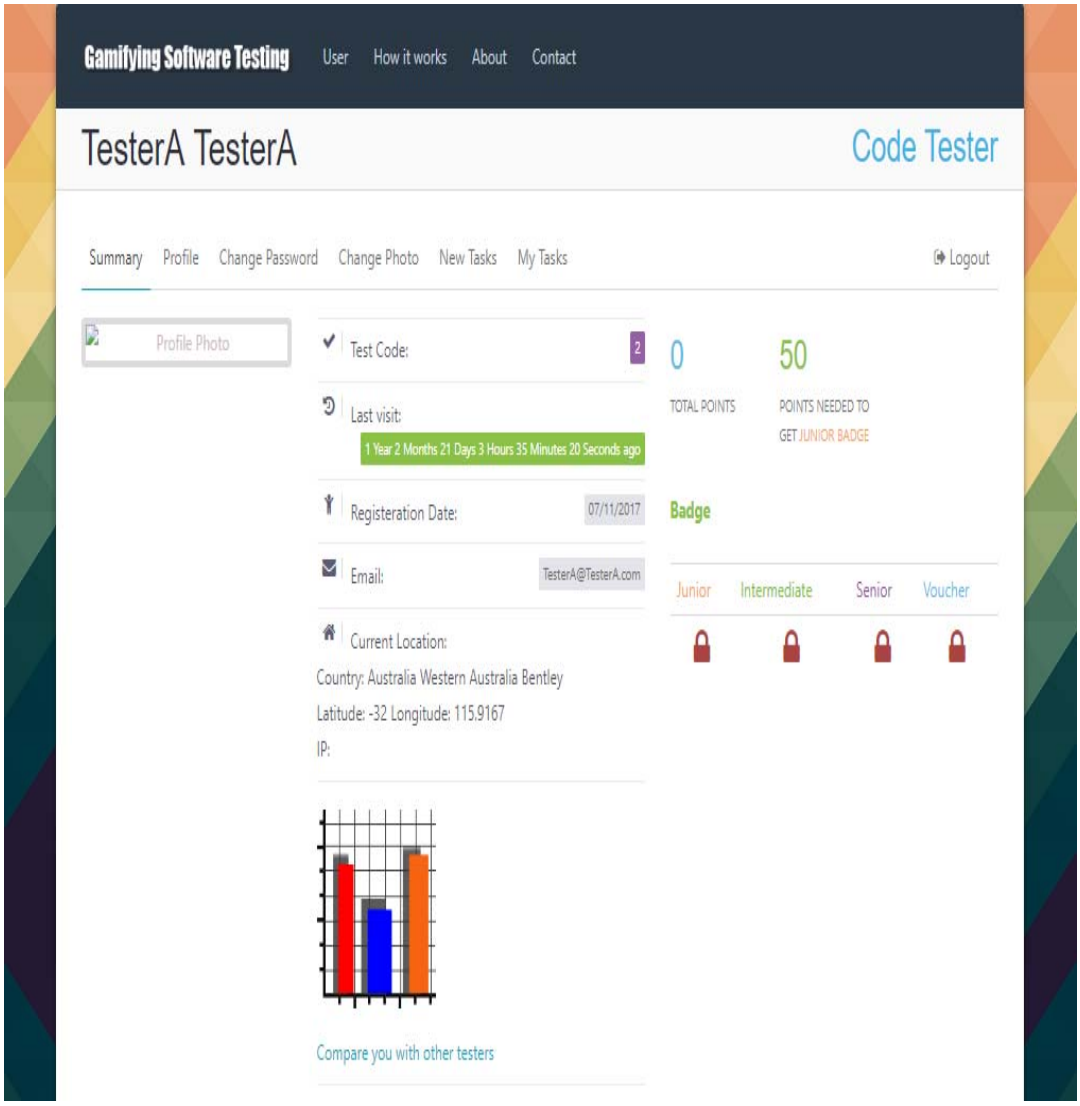

Figure 4.1: Game elements used in tester's home page

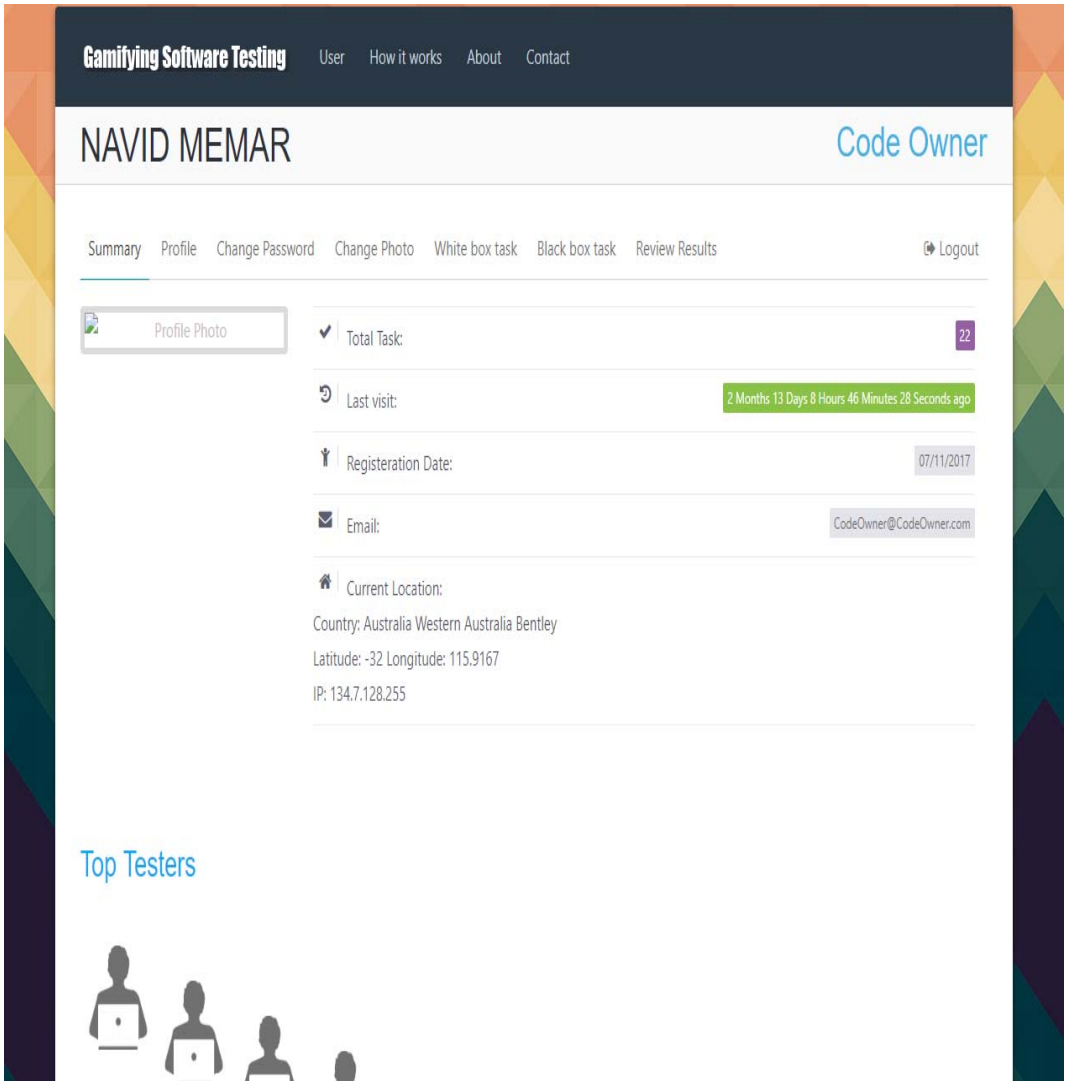

Figure 4.2: Code owner's home page

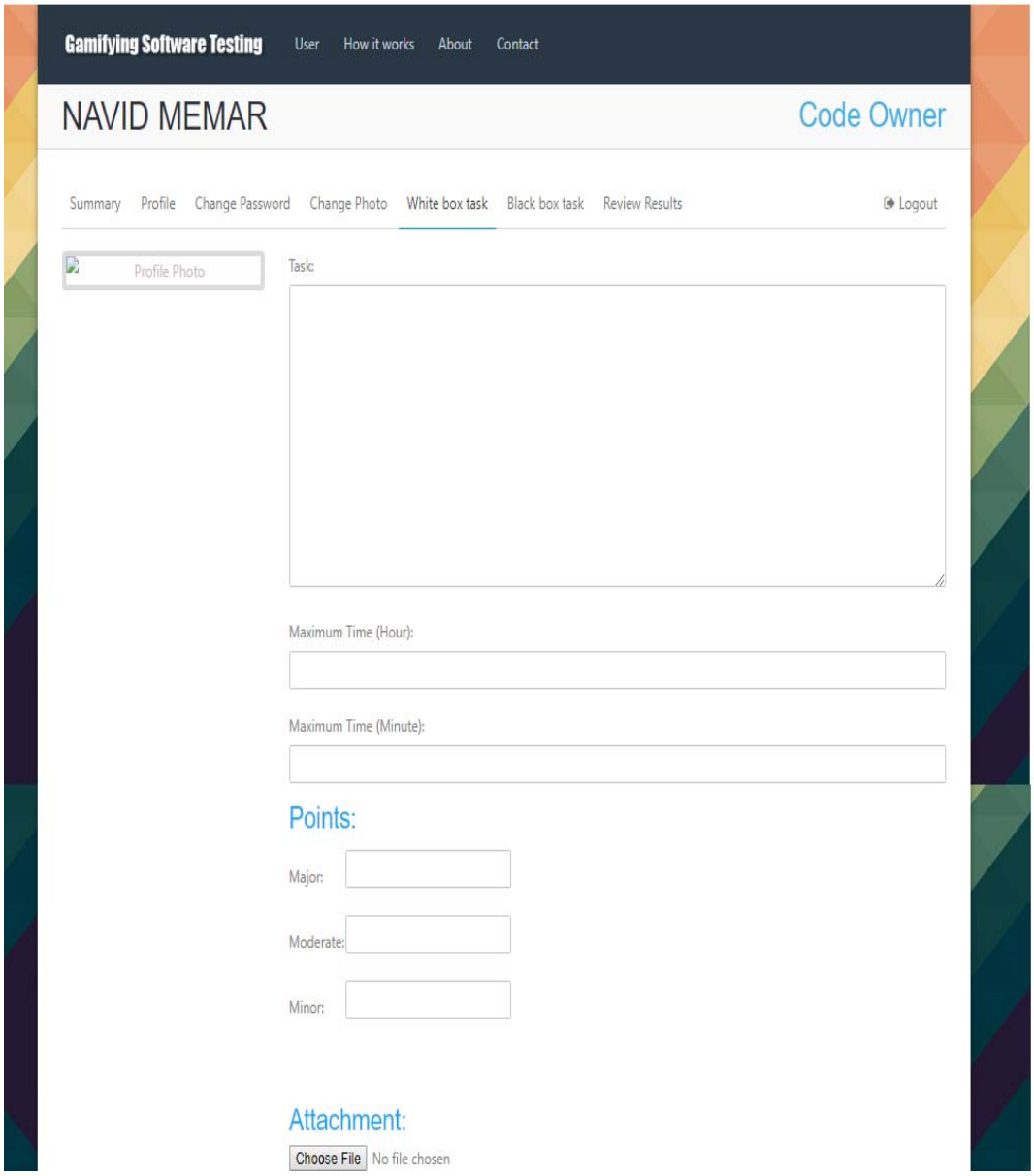

Figure 4.3: Steps to assign a testing task to testers by the code owner

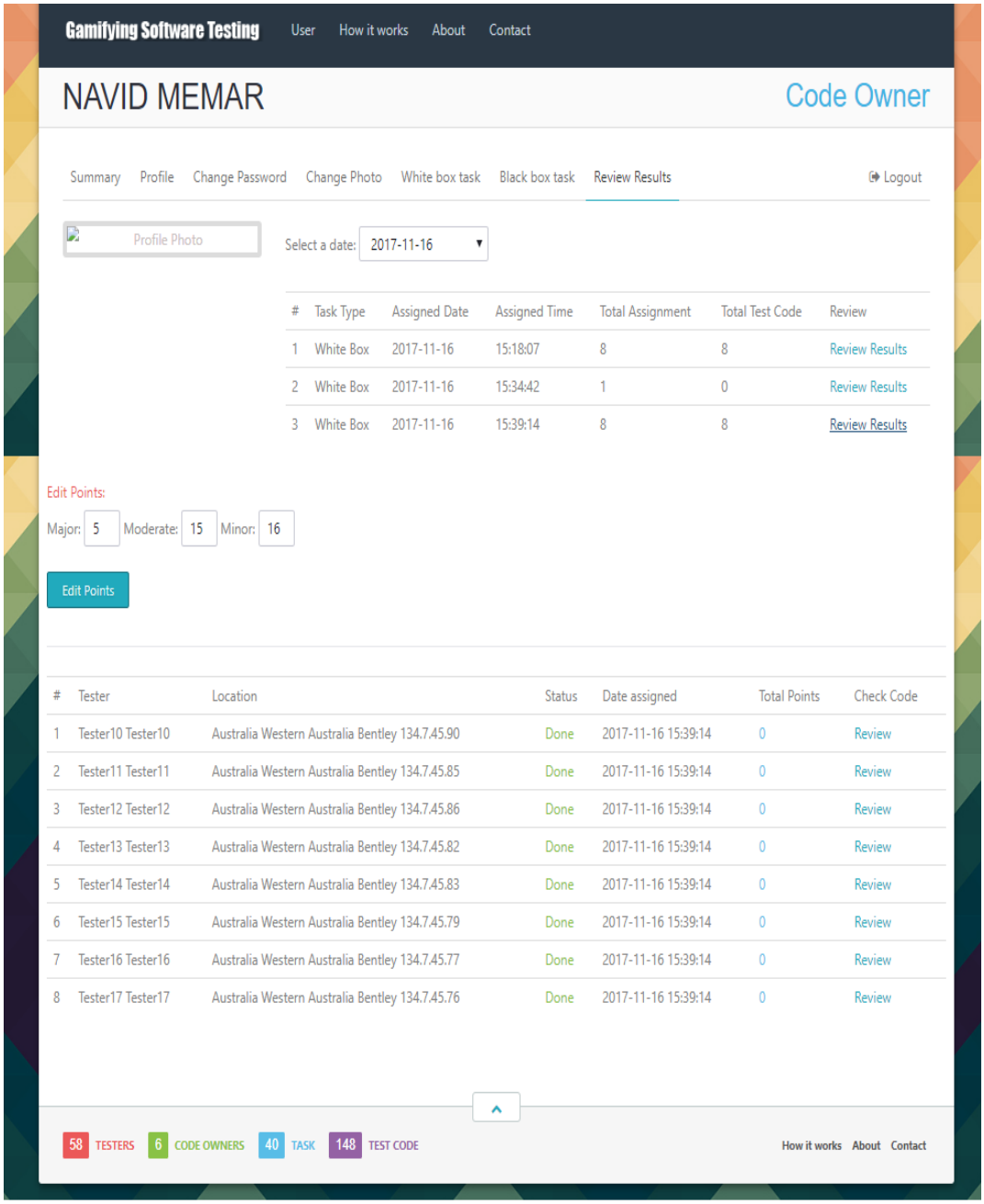

Figure 4.4: Testers' tasks revision by code owner

Following the implementation of the finalized gamified software testing platform, several focus group sessions were conducted to evaluate the developed gamified platform. The focus group discussions were conducted to ascertain the usefulness of this method to increase tester motivation, and provide them with an engaging and rewarding environment. Recorded focus group discussions and

written questionnaires were used for gathering this information. This information and a number of similar responses were then matched with identified key factors and converted into graphs to represent the results. In the next section, these findings will be presented and explained in detail.

4.3 Results and analysis

In this evaluation, a total number of 20 third year computing students (17 male and 3 female), who had experience in working on simulated industry projects and practices as part of their learning process, attended the sessions. Out of them, 80% had both Software Development and Software Testing background, 10% had Software Development background, and the remaining 10% had Software Testing background. All 20 students had learned about and were familiar with unit testing techniques (mainly expert users of Junit framework). During the focus group sessions, we allowed the participants to use the platform and experience the gamified environment. Recorded focus group discussions and written questionnaires were used for gathering this information. This information and a number of similar responses were then matched with identified key factors and converted into graphs to represent the results. In the next section, these findings will be presented and explained in detail. In the questionnaire, the participants were given the following question to answer: “How likely is it that you would recommend other testers to use this software testing platform for their testing task?”. Results suggested that majority of participants were satisfied with the platform and would like to recommend it to other testers. Figure 4.5 shows the level of satisfaction after evaluating the final platform.

In addition, results suggest that the participants felt that this application is interesting and motivating. Arnarsson and Jóhannesson (Arnarsson & Jóhannesson, 2015) also applied the gamification method to increase the motivation of developers to generate high quality unit tests. Their findings also

suggest that developers were satisfied and agreed that the gamified platform encouraged them to participate actively and provide useful tests.

Another question was related to the likelihood that participants would recommend other testers to use the current software testing platform, and results suggest that majority of participants (15 out of 20) would recommend this platform to other software testers for their testing activities.

Some of the responses are listed below for quick reference:

- "I enjoyed the fact that there is a reward system to complete testing and that there is a certificate for completing a certain amount of testing. I do feel like once a certificate is achieved, there is less of a motivation to continue with the testing so perhaps further rewards are needed. I do like the idea of the progression system, however."
- "The competitive aspect of it could be quite entertaining in a workplace environment, especially with highly competitive/social co-workers."
- "This will create a competition environment for testers."
- "Testing through JUnit, blackbox, or whitebox seemed simple."
- "The time and point system makes it more rewarding to complete."
- "Experience is rewarding in the way it gives positive reinforcement upon submission of code, and also provides an incentive to perform better with the rewards of points and badges."

Some also argued that gamification may not be a method to motivate all software testers. The following statement is an example to support this idea:

- "Gamification could appeal to certain people as a motivation for software testing, but it may do little to nothing for some people. Some people may even find it patronizing, although I imagine that this is a small minority of people."

Figure 4.5: Game elements used in the gamified software testing platform

Participants were also given the following question to answer: "How did the gamification experience (i.e., points, storyline, feedback, levels, badges, progression, comparison, certificate, and provision of choice) encourage you to participate and complete the provided software testing task?". The responses indicate that majority of participants agreed that the final gamified platform was encouraging. Some of the responses are listed below for quick reference:

- "The levels, almost a ladder-like structure, appeal to a competitive drive within me, and with the addition of physical rewards, it creates a desire to find something important (such as a design breaking bug) to reach the top of the standings."
- "Knowing that there was a reward for testing the software helped motivate me. In addition, the competitive aspect also motivated me to try to perform better than my peers."
- "I felt slightly more compelled to do well due to the competitive nature of seeing other people's points. It reminds me of when I did language perfect where you could see the people that were ahead of you on points. It kind of pushed you to beat and 'one-up' them and get ahead."

- “The strongest points in my experience are the comparison, certificate, and feedback. Receiving physical and real world rewards both as a certificate and by comparing myself to others is a strong motivator in doing more, better tests.”
- “I really love the fireworks! It made me feel accomplished and having a point reward system would help drive me as the tester to finish testing the software.”

Arnarsson and Jóhannesson (Arnarsson & Jóhannesson, 2015) (Arnarsson & Jóhannesson, 2015) also reported that the leaderboard and point system were the most effective elements identified in their study to influence developers to generate more effective unit tests. This provides a competitive environment that increases the motivation and engagement levels of developers to perform testing tasks.

Furthermore, findings suggest that participants agreed that gamification can be used as a tool to increase the quality of software testing activities. From the responses, 85% of the participants supported this fact. Figure 4.6 shows the participant responses to this question. Some of the responses are listed below:

- “As listed above, I think it would exponentially increase the amount of work people would put in, knowing they would be compensated for it either via digital standings or actually.”
- “I believe that if the tester feels some sort of accomplishment in their testing, they'd certainly feel more motivated, thus improving the quality.”
- “Physical or virtual rewards are very important for everyone. To achieve better points compared to others will motivate the software tester to focus more on the quality because points are assigned to their quality of testing.”

Some also had different responses, as listed below:

- “Yes and no, because if the software tester is looking more at the code and less at the actual gamified software testing application, then the link between the reward and tested software is weaker. However, if the depth of the software testing is short and it allows users to test more with more rewards given in a shorter time, then I would assume people would have a higher level of quality for the testing.”

- “I don't think it would necessarily improve the quality of the software testing, as I didn't feel that it helped my skills or process of testing while I was doing it.”

The following question was also asked to the participants: “From your experience, do you think that gamification of software testing would increase the engagement of software testers?”, Results suggest that 85% of the participants agreed that gamification is a method used to increase the engagement of software testers. Figure 4.7 shows the responses obtained from the participants. Some participants stated that testers would like to gain incentives when they are given tasks to work on.

Some of the responses are listed below:

- “It is likely to achieve achievements (and no punishment) when we are in gaming environment so it would encourage more to testers to test the program.”
- “Being able to see how you are doing compared to other and having the competition part of the system does add a competitive edge to the testing and that can increase the engagement.”
- “It helps as people do love incentives when given a specific task. I guess doing a redundant task and not being rewarded would put off people for some of the testing software especially if it is a tedious and straining one.”
- “While the quality may not increase, I do believe the engagement would. The way the website handles the jobs, for me at least, makes it more enjoyable than thinking "I really need to test this code" and then having to make yourself do it. I know for me, I generally procrastinate writing the test code and stuff because its really tedious and I feel like I have to start by my own volition. With the jobs and points system, its easier to start because it almost starts the task for you. By that I mean when you assign points to it and make it out to be a goal or objective, it makes it seem to be more of an actual task rather than a necessary evil.”
- “Yes, appealing to the competitive nature of people can be a big motivator. Giving points also allows testers to feel like there's a quantifiable measurement of their progress.”

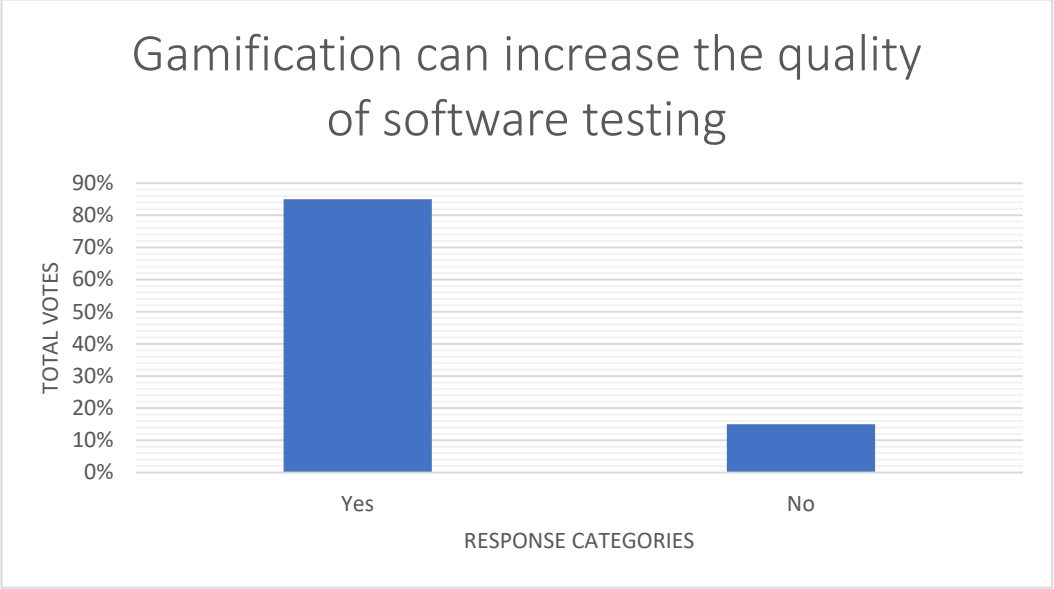

Figure 4.6: Gamification may increases the quality of testing activities

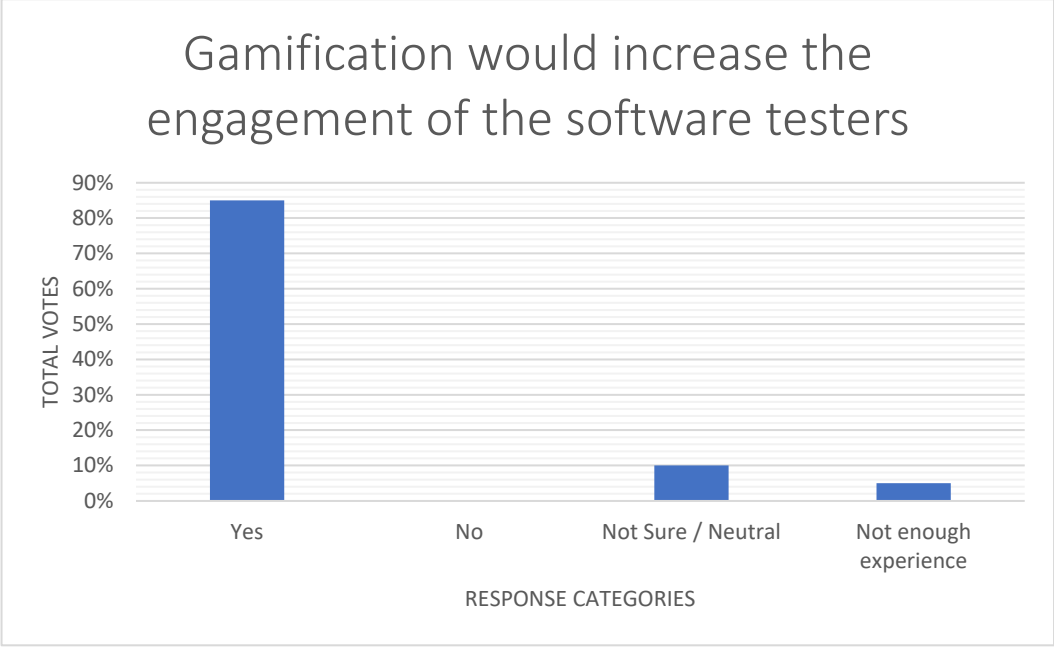

Figure 4.7: Gamification increases the engagement level of software testers

To identify the importance of another key element used in the software testing platform for software testers, participants were asked the following question: "How do you rate the importance of the feedback element?". Responses suggest that feedback is a vital element in designing serious games in software testing environment. Figure 4.8 shows the participant responses to this question. Following this question, participants were asked: "Do you feel encouraged to participate more after reviewing your performance against the performance of others in the software testing platform?". Their responses suggest that majority agreed on the fact that comparison is very important to encourage testers to perform better. Some of the responses are shown below for quick reference:

- "I am competitive, I don't like not winning, and I'd participate more in order to reach the top."
- "The competitive aspect gives a novel gimmick to encourage engagement."
- "Having people to compare gives context to my skill level, which helps motivate me to improve."

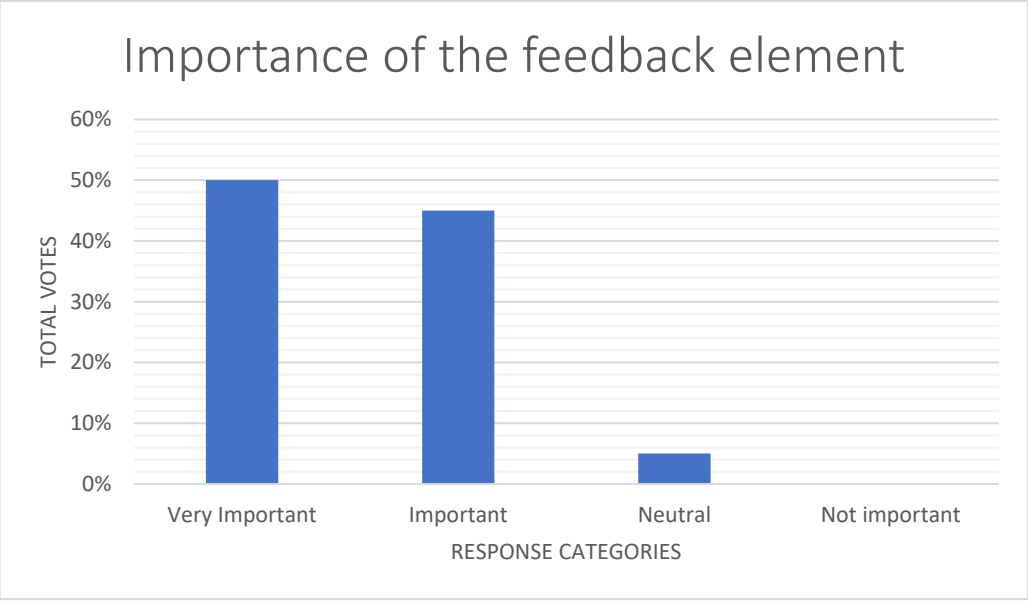

Figure 4.8: Importance of the feedback element

To capture the importance of gamification and game elements used in the gamified software testing platform, we asked the following question: “How did the gamification experience (i.e. points, storyline, feedback, levels, badges, progression, comparison, certificate, and provision of choice) encourage you to participate and complete the provided software testing task? Responses indicate that majority of participants felt encouraged by working with the gamified platform and experiencing the game elements used in the system. Some of the responses are listed below:

- “The levels, almost ladder-like structure appeals to a competitive drive within me, on top of the addition of physical rewards, it creates a want to find something important such as design breaking bug in order to reach the top of the standings.”
- “I really love the fireworks! It made me feel accomplished and having a point reward system would help drive me as the tester to push to finish testing the software.”
- “I felt slightly more compelled to do well due to the competitive nature of seeing other people's points. It reminds me of when I did language perfect where you could see the people that were ahead of you on points. It kind of pushed you to beat and 'one-up' them and get ahead.”
- “The strongest points in my experience are the comparison, certificate and feedback. Receiving physical and real world rewards both as a certificate and as comparing myself to others is a strong motivator in doing more, better tests.”

Results also indicate that a majority of the participants agreed that reviewing their performance against the performance of other testers helped to encourage them. Some of the related responses are listed below:

- “The competitive aspect gives a novel gimmick to encourage engagement.”
- “Having people to compare gives context to my skill level, which helps motivate me to improve.”
- “Being able to see how I went compared to others adds a nice competitive edge to the testing.”

- “It gives some indication of where I stand in comparison with others. It gives me a clear path to improve and feedback on how I am going.”
- “I am competitive and if I am losing to another person then i will be encouraged to participate more.”

4.4 Recommendation

This study helped the researcher obtain information to validate the developed gamified software testing platform. These results were obtained after conducting various focus group sessions with developers and software testers and gathering their information after experiencing the developed tool. Results suggested that participants agreed that the developed tool helps to motivate software testers. Majority of participants indicated that they would recommend this platform to other software testers for their testing activities. Participants also suggested that feedback plays an important role in software testing and is a vital element in designing serious games in software testing environment. Participants invited to participate in the focus group sessions in this study were senior level computing students with experience of working on real-time industry projects as part of their learning process. The results obtained in the study might impact on the outcome of the study as there might be higher expectations or different views in relation to design validation by a larger group of professional software testers. In this study, we tried to choose participants with both software development and software testing background and experience to obtain better results on the final gamified platform validation. Undoubtedly, it would be highly beneficial to obtain the expert opinions of industry professionals for further evaluation of the platform. This will help to identify whether the developed platform also meets the expectations of professional testers who have deeper level of knowledge about industry needs.

4.5 Chapter summary

In this chapter, findings were presented after conducting focus group sessions for the evaluation of the final gamified platform. Results suggested that the developed gamified platform may be a solution to increase the level of satisfaction and engagement of software testers in practice. In addition, majority of the participants agreed that game elements such as feedback and comparison may help to increase testing motivation, engagement, and experience of software testers. Furthermore, they rated the feedback element as a vital element in gamified software testing. Majority of the participants also agreed that gamification can be used as a method to increase the quality of software testing activities. Next, we will examine whether time restriction can be a factor that motivates testers to increase their productivity. Moreover, we will examine if gamification can be a method to increase the quality of the written test codes.

Chapter 5

Software testing gamification with time restrictions on testers' performance

5.1 Introduction

In the previous chapter, the results obtained from various focus group sessions with software developers and testers after the evaluation of the developed gamified software testing platform were presented. The results helped to evaluate whether the developed tool could help increase the motivation and engagement of software testers in a rewarding testing environment.

This chapter focuses on the evaluation of the developed gamified software testing tool to make the software testing experience more engaging and rewarding for software testers. Furthermore, this chapter presents a new gamified metric to evaluate the effectiveness of software tester performance fairly and more accurately.

Finally, this chapter explores the importance of time pressure on software testing practice and evaluates whether the proposed tool can improve software tester performance.

Some of the material in this chapter has been submitted to the Australasian Journal of Information Systems (under review):

Navid Memar, Aneesh Krishna, David A McMeekin and Tele Tan (2019). Investigating Information System Testing Gamification with Time Restrictions on Testers' Performance.

5.2 Earlier work

This section briefly discusses the related works carried out previously. This helps lay the context within which the gamified software testing platform reported in this chapter has been developed and evaluated.

In the background chapter, issues such as lack of interest and motivation were identified as barriers for computing graduates (and students) to consider software testing. Gamification is the strategy suggested as a potential solution to address these issues (Memar, Krishna, McMeekin, & Tan, 2017, 2018). The gamification method has the potential to increase the engagement of software testers. It can also help remedy the high level of repetition and reduce the boredom level of testers while executing their testing activities.

The serious game principles suggested by Whyte et al. (Whyte et al., 2015) helped in identifying the game elements suitable for the gamified software testing platform. A series of focus group sessions involving software developers and testers helped in determining the main elements to be applied in the gamified software testing platform. The main purpose of the study (Memar et al., 2017) was to determine if the design elements and game core elements can help to increase the learning and motivation of software testers.

The core serious game principles are categorized as storyline, goal-directed learning around targeted skills, feedback and rewards, badges and levels, and provision of choice. Storyline helps in improving the level of motivation, quality, and engagement. Goal-directed learning helps to provide a challenging environment. Feedback and awards play a critical part in shaping behavior in players of serious games. Badges and levels help to provide a challenging yet achievable level of difficulty for software testers. Finally, provision of choice helps players have some level of control over certain aspects of the game.

Results discussed in the previous chapter indicated that the developed gamified software testing platform may be a solution to improve tester performance, engagement, and testing experiences. Participants of this study suggested that the developed platform is interesting and motivating for software testers. The results also suggested that gamification can be used as a tool to increase the quality of software testing activities (Memar et al., 2018).

5.3 Methodology

As discussed in the background section, there are few studies related to software testing and gamification. Critical gaps remain in the current framework that need to be addressed to ensure its wider adoption. The objectives of this study are as follows:

1. To identify the perceived effect of time restriction in a gamified software testing environment on individual software tester performance;
2. To introduce a new metric to evaluate the performance of software testers fairly and more accurately;
3. To identify the effect of gamification on detecting different levels of bugs within the software.

This section explains the chosen method for the evaluation of software tester performance in a gamified software testing platform. The activities involved are discussed here in detail.

5.3.1 Evaluation of final gamified testing platform

The next step of the research was first to evaluate the effect of time pressure on software tester performance, and to identify whether time restriction is a factor that can increase the productivity of software testers. For this purpose, 40 computing undergraduate students who had experience in software development and software testing as part of their undergraduate course were recruited. Some participants had previously participated in industry-based software development projects. The Human Research Ethics Committee of Curtin University (approval number HR28/2016)

approved the study. The participants were all introduced to the software testing technique using the JUnit framework. They were briefed on the gamified platform and were given enough time to become familiar with the environment. In total, the participants were given two tasks. Task 1 consisted of 6 easy, 2 medium, and one hard bug, while Task 2 consisted of 2 easy, 4 medium, and 2 hard bugs. Participants were briefed about each task and had the opportunity to ask questions during their reading time. Participants were given 15 and 20 minutes to work on Tasks 1 and 2, respectively. Moreover, an additional 5 minutes of reading time was provided at the beginning of each task to help participants get familiar with the tasks to be performed. To ensure that the participants were aware of the code contents, comments about the code were added alongside the code. Table 5.1 presents the number of bugs introduced for each task. Bugs were grouped into three categories: easy, medium, and hard.

Table 5.1: Number of bugs for each task

Tasks	Easy	Medium	Hard
1	6	2	1
2	2	4	2

The following are the description and examples for each bug type:

A) Easy: A bug or mistake in the software that can be fixed by modifying one line of source code, or a bug whose invalid logic resides in one line of source code.

Example 1: var = “temp”;//instead of var = imported_variable

Example 2: setting the value of a field to the imported value; without validation (i.e. Checking for NULL).

B) Medium: Logical errors that will not function correctly on all occasions.

Example 1: fuel += amount; // This refuels the fuel tank by the given amount, but it does not check for invalid fuel amounts (i.e. negative).

Example 2: Not checking array bounds. The logic will work fine until the array is full, and then the program will crash.

C) Hard: Bugs are classified hard if they are both hard to detect, and only trigger on rare cases.

Example 1: In a class named 'Book', writing a page works fine unless the page is full and the number of pages in the book is full.

Example 2: Driving the car works fine in most cases, except that the car should not drive if there is no fuel.

Written questionnaires and recorded discussions were used to establish the evaluation results. In the following section, the findings from this study are explained in detail.

5.4 Results and analysis

This section presents the results and findings obtained from the evaluation of the developed gamified software testing platform. In the first section, the focus is on the qualitative results obtained from the questionnaires during the evaluation session. The given questions are designed to identify the effect of time restriction on software tester performance. In the second part, the available metrics are defined and new metrics to measure the performance of software testers are introduced. Furthermore, the proposed metrics are validated through a comparison of the existing and proposed metrics. The last section presents the performance evaluation results of software testers for each level of task difficulty. This comparison helps to identify the effectiveness of gamification based on the level of task difficulty.

5.4.1 Qualitative results

In this evaluation, 40 participants attended the evaluation session, and out of them 72% have both software development and software testing background, 20% have software development background, and the rest have software testing knowledge. All participants had experience in unit testing technique using the JUnit framework.

During the evaluation, students were given an introduction about the gamified platform, and they had the chance to test the platform and experience the gamified environment. Participants were given the chance to write test codes for two different tasks. As discussed earlier, they were given 5 mins of reading time for each task. The testers were also given 15 and 20 mins to work on Tasks 1 and 2, respectively. The tasks were designed such that the testers could understand the code easily and write test codes for the given tasks in the given time frame. At the end of the evaluation session, the participants were given the questionnaire to provide feedback on the effect of time restriction on their testing experience. In the questionnaire, participants were asked the following question: "What is the effect of time restriction on the software testing performance in the current gamified software testing platform?". Results suggested that majority of participants agreed to the fact that time pressure may compromise the performance of software testers. Some of the responses are listed below for quick reference:

- "Having a time restriction may place pressure on the tester, causing the tester to rush."
- "The time constraints make me feel under pressure and I do not think logically due to the stress."
- "The time restriction puts a level of pressure, making you feel rushed, leading to the lack of time to properly read the code and understand what is going on and leading to rushed testing."
- "As a tester you feel that you won't have the required time to test everything within the classes, and therefore can only find some potential errors."
- "I felt under pressure when being timed, meaning I made quick, poor decisions I would not normally make when not being timed."
- "Testing should be very broad and cover as much of the code as possible, putting time pressure on it, it makes you have to compromise in terms of which methods to test and how in depth you test the methods."
- "Time restriction in my opinion is a limiting factor when it comes to software testing. A good software tester is able to get work done in a good amount of time

without having an artificial restriction imposed on them. Longer timeframes may improve performance because the tester will be able to write all test cases anyways, but overall I feel it is diminishing when the timeframe is too low."

Another question was asked regarding the effect of time restriction on the software testing quality in the current gamified software testing platform, and the results suggest that everyone agreed that time restriction can compromise the software testing quality.

Some of the responses are listed below:

- "Depending on how long the restriction is, just like writing code, with writing test harnesses mistakes can be made if in a rush. And without the proper time to go through it all, the mistakes can get through and create incorrect results."
- "Under time restriction, a subject would be more focused on finding more solutions to a problem rather than elaborating on them. Limited time may leave a subject spending less time on an individual task to ensure they get more tasks done."
- "When having a time restriction it led to feeling rushed, meaning I didn't have time to properly test to a high enough quality as I had to keep in mind of the time. Meaning I could not cover all major test cases, meaning more bugs could slip through."
- "When there is a time restriction, there is pressure to rush and compromise quality for quantity. This leaves room to miss several edge cases and therefore not catch as many bugs."

The participants were also asked the following question: "How likely is it that time pressure makes the testers more productive?". The responses suggest that 41% agreed of the fact that time pressure may be a method to increase the productivity of software testers, while 35% did not agree to the fact and the rest of participants were not sure whether this would make the testers more productive. At the end of the questionnaire, participants were given a chance to provide additional comments or suggestions, some of which are listed below for quick reference:

- "If time restriction is ever to have any positive impact, a large amount of research should be dedicated to finding how much time testers should be given for a certain task. Giving too much time or too little time would decrease the performance of the tester, but if a certain limit is reached, it may improve performance. But this limit would be different per tester and per task, so it seems unlikely to be feasible."
- "It really all depends on who the testers are, as some people become more productive with time restraints and others will get stressed out and end up doing less work. Personally having a flexible time pressure helps me ensure that I try to finish it for a dead line, however if needed can extend the time to get more work done."

5.4.2 Quantitative results

5.4.2.1 Metrics definition

In the software testing context, one of the important factors for software tester performance is to be evaluated fairly and accurately, and this may result in higher job satisfaction rate in organizations (a detailed discussion has been provided in the background section). In this section, new metrics will be introduced for this purpose. By using the proposed metrics, software tester performance could be evaluated based on the importance of identified bugs. In addition, a comparison of new metrics with existing metrics will be provided to validate the proposed metrics. A detailed discussion is presented to explain why these metrics should be adopted for software testing performance evaluation to provide more accurate evaluation results. Finally, the accuracy and fairness of the proposed metrics are also discussed in detail.

1. Effectiveness metric can be calculated with the following formula: E = vdf / vdt while the proposed gamified metric evaluates the performance of testers with the use of following formula: Gamified E= (((0.2*e)/E) + ((0.3*m)/M) + ((0.5*h)/H)). In the effectiveness metrics, E is effectiveness, vdf is the number of unique valid defects found, and vdt represents the total number of unique valid defects (Mäntylä & Itkonen, 2013). In contrast, in the gamified effectiveness metric, Gamified E is the

effectiveness, e represents the number of unique valid easy defects found, E is the total number of Easy defects, m is the number of unique valid medium defects found, M is the total number of Medium defects, h in the number of unique valid hard defects identified by software testers and H is the total number of Hard bugs existing in the software. This new metric acknowledges the performance of software testers' based on the level of detected bugs. For instance, each identified easy bug consists of 0.2 points, while each valid identified medium and hard bug consist of 0.3 and 0.5 points respectively. The new metric provides weight for each particular bug found by the tester depending on the importance of each bug detected. Figure 5.1 presents the comparison of effectiveness, known as Recall in (Baeza-Yates & Ribeiro, 2011)) and gamified metrics. Results suggest that the correlation coefficient between the two metrics is 0.96656 which shows a strong relation between the two metrics.

2. Although each tester may identify a certain share of unique defects, they additionally may produce a set of invalid bugs (also referred as false positive (Dunsmore, Roper, & Wood, 2003)). The share of valid unique findings among all findings is called validity (Hartson, Andre, & Williges, 2001) and (in the domain of information retrieval, this is commonly referred to as precision (Baeza-Yates & Ribeiro, 2011) and this number is often not reported in in empirical software engineering (Mäntylä & Itkonen, 2013). Validity or precision can be calculated with the following formula: V= (tp / (tp + fp)) where TP is the true positive, (when tester detects a bug and bug exists) and fp represents the false positive (invalid bug reports). In contrast, the proposed gamified validity metric evaluates the performance of testers with the use of following formula: Gamified V = (((0.2*e)/(E + fp))+((0.3*m)/(M + fp)) + ((0.5*h)/(H + fp))). Figure 5.2 presents the comparison of validity and gamified validity (known as Precision (Baeza-Yates & Ribeiro, 2011)) metrics. Results indicate that the two metrics have a high correlation coefficient with a value of 0.973281. This number supports the fact that there is a strong relation between the two metrics.

3. Lastly, for decision making purposes, the combination measures of effectiveness and validity will be beneficial to evaluate the overall performance of testers more accurately. The importance of this combination measure is in determining a better

performance results. In the information retrieval domain, both effectiveness and validity are combined in a measure called F-score (Van Rijsbergen, 1986). F score can be calculated with the following formula: Fs = 2 * ((Validity * Effectiveness) / (Validity + Effectiveness)) (Mäntylä & Itkonen, 2013) while, the proposed gamified f score can be calculated as following: Gamified Fs= 2 * ((Gamified Validity * Gamified Effectiveness) / (Gamified Validity + Gamified Effectiveness)). Figure 5.3 presents the comparison of f score and gamified f score metrics. Results indicate that the two metrics have a strong correlation coefficient with a value of 0.974372.

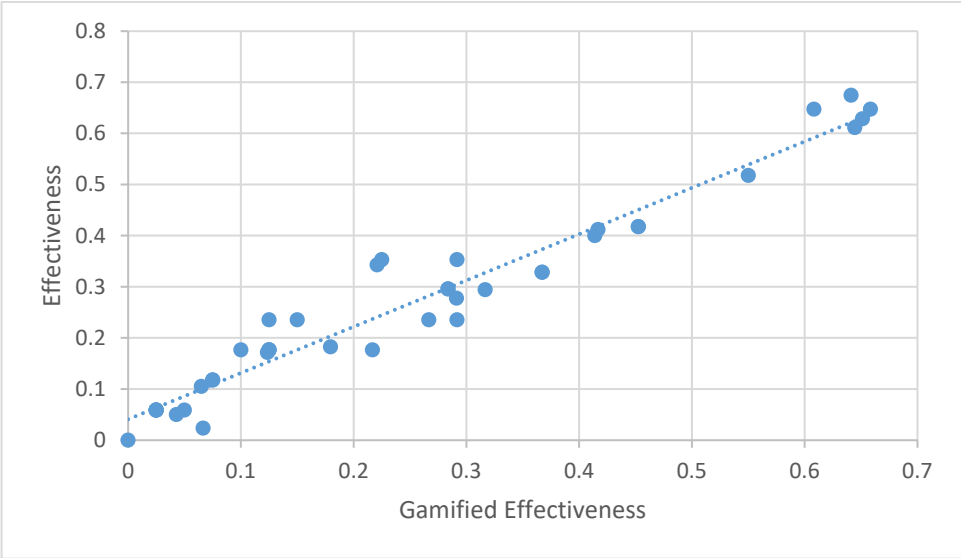

Figure 5.1: Effectiveness Metric vs. Gamified Effectiveness

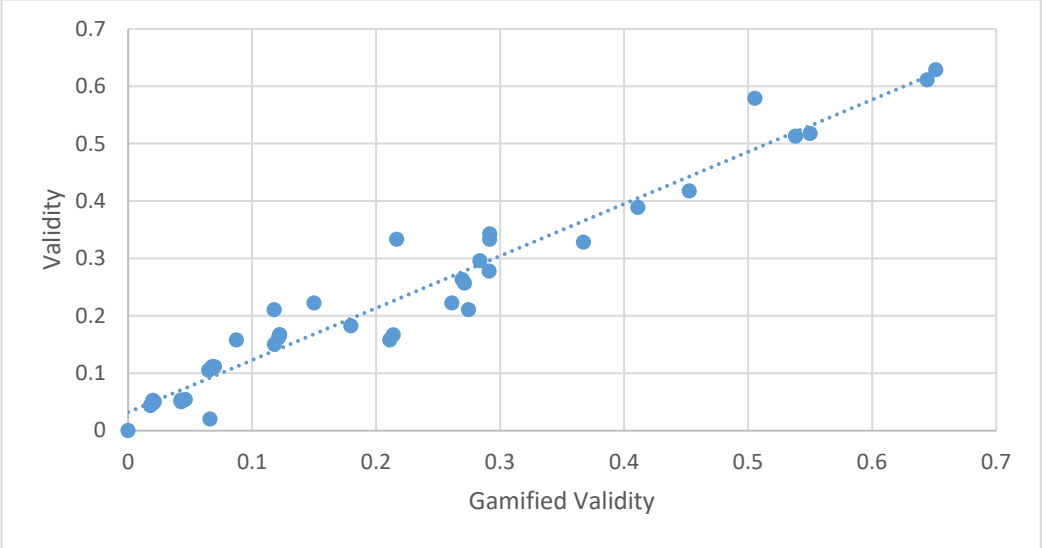

Figure 5.2: Validity metric vs. Gamified-Validity metric

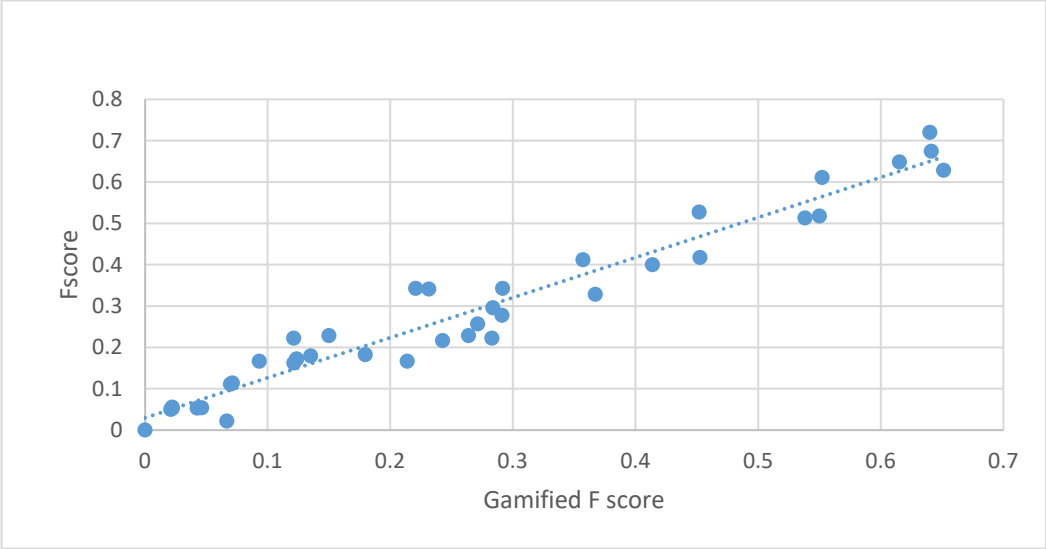

Figure 5.3: F Score metric vs. Gamified F-Score metric

To categorize the performance of software testers, the gamified effectiveness metric has been selected. The performances of the participants were grouped into three categories of low (performance rate less than 0.2), medium (performance score between 0.2 and 0.5), and high (performance rate greater than 0.5).

Table 5.2 presents the performance evaluation of participants using effectiveness metric, while Table 5.3 presents the performance of testers using the gamified

effectiveness metric. For this purpose, the performance of all 40 participants has been evaluated. Findings suggest that the effectiveness metric provides similar performance results for participants that identified the same number of bugs by disregarding the importance of bugs identified by the participants. In contrast, the proposed gamified effectiveness metric provides more accurate performance results by considering the type of bug detected by participants. For instance, participants who identified more important bugs achieved better performance results compared to those who identified non-critical bugs. For this comparison, we calculated the average performance of participants using both metrics. Furthermore, the average performance of participants in identifying the total number of bugs helped to find the relationship between the scores obtained from the effectiveness metric and number of bugs detected by the participants. In contrast, the average performance of participants in detecting different categories of bugs helped the researcher to identify the relationship between the performance of participants and their performance in identifying different levels of defects using the gamified effectiveness metric.

Moreover, gamified validity and gamified F-score metrics provide more accurate evaluation results considering different level of bugs while the existing validity and f score metrics do not consider the importance of bugs identified by testers. Table 5.4, presents the performance evaluation of participants using validity metric while Table 5.5, shows the performance evaluating using gamified validity metric. Furthermore, as discussed earlier, for decision making purposes, the combination measures of effectiveness and validity will help to evaluate the performance of participants more accurately. Table 5.6, represents the participants' performance evaluation using the f score metric. However, in order to provide more accurate and fairer results, gamified f score metric was used. Table 5.7, represent the results of participants' performance using gamified f score metric.

Table 5.2: Performance evaluation using effectiveness metric

	Performance	Score (number)	Total Bugs (%)
Effectiveness	**Low**	0.1039	10.39
	Medium	0.2974	29.19
	High	0.6164	61.64

Table 5.3: Performance evaluation using gamified effectiveness metric

	Performance	Score (number)	Easy (%)	Medium (%)	Hard (%)
Gamified Effectiveness	**Low**	0.06842	16.13	14.27	0.72
	Medium	0.2893	28.74	29.62	29.23
	High	0.6466	66.66	50	68.24

Table 5.4: Performance evaluation using validity metric

	Performance	Score (number)	Total Bugs (%)	Invalid bugs (number)
Validity	**Low**	0.9208	10.39	3
	Medium	0.2849	29.19	1
	High	0.5784	61.64	1

Table 5.5: Performance evaluation using gamified validity metric

Gamified Validity	Performance	Score (number)	Easy (%)	Medium (%)	Hard (%)	Invalid bugs
	Low	0.0834	16.13	14.27	0.72	3
	Medium	0.2942	28.74	29.62	29.23	1
	High	0.5333	58.76	54	68.24	1

Table 5.6: Performance evaluation using F-score metric

F-score	Performance	Score (number)	Total Bugs (%)	Invalid bug(number)
	Low	0.9866	10.39	3
	Medium	0.2802	29.19	1
	High	0.5943	61.64	1

Table 5.7: Performance evaluation using gamified F-score metric

Gamified F-score	Performance	Score (number)	Easy (%)	Medium (%)	Hard (%)	Invalid bug(number)
	Low	0.0648	16.13	14.27	0.72	3
	Medium	0.2904	28.74	29.62	29.23	1
	High	0.6045	58.76	54	68.24	1

5.4.2.2 Effectiveness of software testers' performance based on the level of difficulty

In this section, we study the effect of gamification on the performance of the participants based on the level of task difficulty. Table 5.8 presents the performance of all participants in the given tasks. To calculate the performance of the participants, we use the effectiveness, validity, and F-score metrics. It is important to note that the proposed gamified metrics could evaluate the overall performance of participants when considering all levels of bugs. However, in this case, we are evaluating the performance of testers focusing on a specific category of bugs. Thus, the existing metrics will be used to evaluate participant performance. To evaluate the performance of the participants, we use the following metrics:

1. Effectiveness = vdf / vdt
2. Validity= (tp / (tp + fp))
3. F score = 2 * ((Validity * Effectiveness) / (Validity + Effectiveness))

Results suggest that the performance of participants in detecting easy bugs in the gamified software testing platform was higher compared to other levels categories. The average performance of all 40 participants in performing testing activity to detect bugs categorised as easy using f score metric was 22.14% while their average performance for detecting medium and hard bugs were 20.4% and 15.52% respectively. In order to calculate these, measures that were presented in the information retrieval domain were used (Baeza-Yates & Ribeiro-Neto, 1999). However, these measures have been partially adopted by the usability community (Hartson et al., 2001) and software engineering community (Mäntylä & Itkonen, 2013).

Table5.8: Effectiveness of testers' performance based on the level of difficulty

Measures	Easy (Task 1) (%)	Easy (Task 2) (%)	Average Performance for Easy Task 1 & 2 (%)	Medium (Task 1) (%)	Medium (Task 2) (%)	Average Performance for Medium Task 1 & 2 (%)	Hard (Task 1) (%)	Hard (Task 2) (%)	Average Performance for Hard Task 1 & 2 (%)
Effectiveness	22.54	22	**22.27**	9	42	**25.5**	5	28.23	**16.61**
Validity	21.17	21.85	**21.51**	9	31.8	**20.4**	5	26.04	**15.52**
F score	21.44	22.84	**22.14**	9	31.8	**20.4**	5	27.04	**15.52**

5.5 Discussion and limitation

In this chapter, results were provided after evaluating the final gamified software testing platform through the evaluation session conducted with 40 undergraduate computing students. Due to higher expectation or different view in relation to evaluation of gamified software testing by a larger group of professional software testers, the obtained results may impact on the outcome of this study. In addition, in order for students to show interest in participating in the focus group session, the duration of the session had to be less than one hour. Thus, time pressure may be the main factor that affected participant performance in this study, and resulted in a lower performance rate by the participants. Moreover, in the given questionnaire, the students agreed that the time pressure could be a factor that affects the quality and performance of the software testers. In this study, the researchers tried to choose participants with knowledge in both software testing (mainly unit testing) and software development. Finally, the researcher discovered that the proposed metrics are

beneficial when evaluating the performance of software testers for different levels of bugs (by considering the importance of bugs identified by the software testers). However, to evaluate the performance of the software testers for a specific category of bugs, the proposed metric may not be beneficial to evaluate the performance of the software testers.

5.6 Recommendation

Evaluation of the gamified software testing platform resulted in several findings. From a broad perspective, we noticed that intensive time pressure could significantly reduce the efficiency of software testers and, in some cases, could be a limiting factor. Furthermore, some participants agreed that time pressure creates good urgency to complete the task. However, this may result in poor testing performance, which defeats the purpose of software testing.

In the software testing context, it is crucial to evaluate software tester performance fairly and accurately, which could result in a higher job satisfaction rate in organizations. Thus, in this study, new gamified metrics have been introduced, which help in evaluating software tester performance based on the importance of identified bugs. This metric has been validated in earlier sections and has been compared with current evaluation metrics. Moreover, in this study, the accuracy and fairness of the proposed metrics is discussed in detail. However, it is important to note that the proposed metric is useful for evaluating the performance of software testers in combination with different levels of bugs, and it may not be useful when evaluating the performance of software testers for a specific category of bugs.

5.7 Chapter summary

In this chapter, the topics of gamification, software testing, and effect of time restriction on performance were discussed. Results suggested that majority of participants agreed to the fact that time pressure may compromise the performance of software testers, negatively impacting on the software testing efficacy. The quantitative results suggested that the performance of software testers was affected by the time pressure introduced during the prototype evaluation. In addition, a set of new metrics was proposed to better capture the performance of the software testers. It has been demonstrated that these metrics can fairly distribute the scores to reflect the types of bugs being reported. Further work will include evaluation of the performance of software testers working individually by using the gamified platform to perform testing tasks versus individuals performing testing activities without using the developed tool. The researcher also plans to investigate the importance of introducing the proposed metrics for evaluating the performance of software testers in the gamified software-testing platform prior to conducting the testing activity. This will then help to identify whether the new metrics could be beneficial in helping the testers detect higher priority bugs within the software.

Chapter 6

Gamified software testing performance evaluation and the effect of proposed gamified metric on the performance level

6.1 Introduction

The previous chapter reported on the evaluation of the developed gamified software testing tool to enhance software-testing experience with the aim of creating a more engaging and rewarding environment for software testers. Moreover, new gamified metrics to evaluate the effectiveness of software tester performance fairly and more accurately were presented. We also investigated the importance of time pressure on the software testing performances.

In this chapter, we study the effect of using the gamified software testing-platform on the performance of software testers with regards to detecting different levels of code defects and the quality of the participants' written test codes. We also examine the significance of introducing the gamified evaluation metric, introduced in Chapter 5, to the participants prior to the testing activity. This will help to identify whether prior knowledge of the evaluation metric would lead to the software testers identifying more difficult bugs when performing the testing tasks.

6.2 Methodology

This section describes the method used to evaluate the gamified software testing platform. The objectives of this study are as follows:

1. To identify the perceived effect of the developed gamified software testing tool in the performance of software testers.

2. To identify the effect of the developed gamified software testing tool in the level of motivation and encouragement of software testers.
3. To understand the effect of gamification metrics on detecting different levels of bugs within the software.

6.2.1 Participant recruitment

The first task is to evaluate the effectiveness of the developed software testing gamified tool on software tester performance to identify whether gamification can motivate software testers when performing testing tasks. For this purpose, 26 participants (20 computing undergraduates, 6 computing PhD students, and 1 computing post-doctoral candidate), who have knowledge in software development and software testing as part of their respective courses, were recruited. 20 participants had previously participated in industry-based software development projects. The participants were all introduced to the information system testing technique using the JUnit framework as some point during their course. JUnit is a simple yet practical testing framework for Java classes that helps to make writing testing code easier and more maintainable (Cheon & Leavens, 2002) . This framework is useful for software unit testing that is usually laborious, tedious, cumbersome, and often difficult. Participants were briefed on the gamified platform and were given enough time (the required time was suggested by a senior tutor and a lecturer from the department of Computing in Curtin university, with deep knowledge in the software testing field) to become familiar with the environment.

This study has the approval of Curtin University's Human Research Ethics Committee (approval number RDSE-76-15).

6.2.2 Evaluation setup

Participants first took part in a 30-min focus group session to discuss software testing and the influence of introducing gamification methods to achieve software testing outcomes. Participants were asked to participate voluntarily in a pre-survey questionnaire with the aim of capturing their opinion about the developed gamified platform and its effect on testing performance. After the focus group sessions, participants were divided into two groups, one group to write the test codes for the given task without access to the developed software testing platform and the other group were asked write test codes for the same task using the developed gamified software testing platform. Both groups were asked to write the test code individually. In the evaluation sessions with both groups, participants were briefed about the task and had the opportunity to ask questions during their reading time. To study the perceived effect of the proposed gamification metric on detecting different levels of bugs, the second group that had access to the developed software testing platform was informed about the gamified evaluation metric. This would help to identify whether this prior knowledge could help the testers detect more important bugs. All participants were given 30 minutes to work on the given task. Moreover, an additional 10 minutes of reading time was provided at the beginning of the given task to help the participants get familiar with the task to be performed. Participants were also given an information package containing vital information about the task and objectives of the production code. To ensure that participants were aware of the code contents, comments about the code were added alongside the task. Finally, participants were asked to voluntarily participate in a short post-survey questionnaire. These questions were designed to obtain participant feedback to understand their perception and importance of the gamification elements used in the developed platform.

6.2.3 Task design process

In total, all participants were given a production code and the task was to identify the bugs in the code. This code consisted of two easy, two medium, and two hard bugs (see Table 6.1). In the process of designing the task, the quality of the production was assessed by two senior tutors and a software engineering lecturer from the school of Electrical Engineering, Computing and Mathematical Sciences, Curtin University. This was to ensure that the designed production task meets the criteria for testing purposes in a specific amount of time matching the participants' level of knowledge. These criteria are listed below:

- Designed production code is bug free (apart from the introduced bugs for the purpose of testing activity)
- Sufficient time frame given to participants to be able to cover introduced bugs
- Information package carries required information for this practice
- Level of bugs introduced in the production code

Table 6.1: Number of bugs in the given task

Easy	Medium	Hard
2	2	2

The description and examples for each bug type are as follows:

A) Easy: These bugs will usually occur in a single line of code. These are the most straightforward to identify and fix. Examples include:
    - Forgetting to replace temporary variables - e.g. var = "tempt" instead of var = imported_value.
    - Not validating imported values – e.g not checking for *Null* / invalid values in setters.
B) Medium: These are bugs where the code may look fine at first, but contain logic errors that only occur in certain cases. Examples include:

- Not properly checking array bounds - errors only occur when the array is full.
- Allowing objects to enter incorrect states - e.g. *weight += amount* may look fine, but adding negative amounts may lead to invalid (negative) weights.

C) Hard: These are the most difficult bugs to detect. The code and logic appears fine, but there are errors which occur only in very specific / rare cases. Example include:

- Adding a new Fruit to a Shipment fails only when the Shipment is full.

Table 6.2 lists the different levels of bugs and their description found in the production code used in this experiment (full production code is accessible in Appendix B).

Example of an easy bug is shown below:

```
public int calcTotalWeight() {
        int totalWeight = 0;
        int numFruits = 0;
   if (numFruits > 0) {
            Fruit f = getFruit(0);
            for (int ii = 0; ii < numFruits; ii++) {
                getFruit(ii);
                totalWeight += f.getWeight();
            }
        }
```

In this case, instead of getNumFruit(), a temporary variable has been used, which causes calcTotalWeight() to return the BASE_PRICE in this case.

An example of a medium level bug is shown below:

```
public void setOrigin(String origin) {
        if (origin.trim() == " ") {
```

```
            throw new IllegalArgumentException("Invalid origin: " +
origin);
        } else {
            this.origin = origin.trim();
        }
    }
```

It can be seen that in this example, instead of empty string, string with a space has been used that allows an empty string as origin in this case.

Lastly, an example of a hard bug is shown below:

```
public double calcPricePerGram() {
        double average;
        average = calcTotalPrice() / calcTotalWeight();
        return average;
    }
```

In this example, calcPricePerGram() returns infinity if there is no item presented. The solution to this bug is as follows:

```
public double calcPricePerGram() {
        double average;
        if (getNumFruits() == 0) {
            average = 0.0;
        } else {
            average = calcTotalPrice() / calcTotalWeight();
        }
        return average;
    }
```

Table 6.2: Bugs introduced in Shipment.java

Difficulty	Line#	Bug
Easy	37	Origin is not validated in Constructor
Easy	116	Using a temp var instead of getNumFruit() causes calcTotalWeight() to only return BASE_PRICE
Medium	86	Comparing with " "(space) instead of""(empty string) means it will accept an empty string as origin
Medium	77	Wrong comparison using < instead of <= means we cannot set id to '9999999'
Hard	131	calcPricePerGram() returns Infinity if no items are present (should be 0.0)
Hard	122	Not using the return value of getFruit() causes calcTotalWeight() to return an incorrect value unless all fruits are the same weight

In the experiment, participants were tasked with using JUnit to test part of an application for a local grocery store used to manage shipments of fruit. They were provided two classes *Fruit.java* and *Shipment.java* for this purpose (Appendix B). The fruit class has been fully tested, but the shipment class contained a number of bugs. Participants were asked to write test cases to identify as many of these bugs as possible in the allowed time.

6.2.4 Class descriptions

Fruit is a simple container class that describes fruits. This class consists of a price (in dollars) and weight (in grams). The shipment class describes a shipment of fruits. Shipments have an id number, origin, and a list of the fruits in the shipment. There are methods to add fruits and calculate the total price, the total weight, and the average price per gram of the shipment's contents.

Furthermore, we provided participants with the following tips to get them more familiar with the production code.

- "Use the tests in FruitTest.java and ShipmentTest.java to help you get started. They contain examples of the different Junit test case you may need."
- "In the interest of time, you do not need to test the getters (getNumFruits(),getId(),etc)."
- "The Fruit class is already tested! You only need to write unit tests for the Shipment class."

In the process of evaluation, written questionnaires and recorded discussions were used to establish the evaluation results. In the following section, the findings from this study are explained in detail.

6.3 Results and analysis

In the following sections, the main findings from the focus group sessions are described. In the first part, qualitative results obtained from the written questionnaires and recorded focus group discussions are described. For this purpose, the list of findings from all focus group sessions and recorded focus group discussions are reviewed. This helped in identifying the key factors that then were coded with the number of similar responses associated with each factor to present more accurate results. Furthermore, those numbers were converted into graphs to represent the findings. In the second part, the available metrics discussed in the previous chapter are used to measure the performance of the software testers. The results of this evaluation helps to identify the effectiveness of the developed gamified tool on software tester performance.

6.3.1 Qualitative results

In total, 26 participants attended the sessions, and out of them, 850% had both a software development and software testing background, 15% had only software

development background. All participants had experience of working on real-time industry projects as part of their learning process, and they had learned unit testing techniques as part of their undergraduate studies. In the following sections, the main themes and findings from the focus group sessions conducted are described. The qualitative findings can be summarized under the following headings:

6.3.1.1 Performing software testing individually is vital

In the questionnaire, participants were first asked the following question: "How do you rate the importance of software testing when performing testing task individually?". This helped in capturing the participant impressions regarding the importance of testing activity when performing individually. A majority of participants agreed on the importance of software testing when performing testing activity individually. Some of the responses are listed below for quick reference:

- "Testing is important to find critical bugs before they cause catastrophic failures."
- "It depends on level of testing. It is obvious that developers have a better understanding of their own code and it would be easier for them to perform unit testing on their production code."
- "Code needs to be tested. This gives assurance that the code is bug free and ready to be used."
- "Codes cannot be used before testing. There are usually faults within the code that need to be discovered before releasing."

6.3.1.2 Gamification improves individual software testing performance

The participants were also asked the following question: "In your opinion, how do you think gamification would affect individual software testing performances?". All 26 participants agreed that gamification could improve the performance of software testers when performing testing task individually.

Some of the responses are listed below for quick reference:

- “Increases the scope of testing which results in improved software quality.”
- “Competition always lead to improvement in performance. Being able to compare your performance with other testers provides encouragement in performing better. On the other hand, point system provides motivation to achieve higher points to unlock higher levels and possibly rewards.”
- “Helps in identifying more bugs. This gives motivation to testers to check all aspects of the code.”
- “Competition improves motivation.”
- “Gives a bit motivation to work on stuff rather than plain old boring way.”

6.3.1.3 Gamification of software testing increases the quality of software testing

Another question was regarding the effect of gamification on the quality of software testing. A majority of the participants agreed that gamification could be an element to improve the quality of software testing. Some of their responses are listed below:

- “Competition and reward system gives that courage to participants to perform better. This means better quality of written test code plus detecting more and higher important bugs within the production code.”
- “It is likely that the performance quality improve when motivation and competition factors are involved.”
- “Efficiency leads to higher quality.”

6.3.1.4 Proposed gamified evaluation metric encourages participants to identify higher priority bugs

Furthermore, to capture the importance of introducing the developed gamified performance evaluation metric prior to the testing activity, participants were asked the following question: “In your opinion, how do you think introducing the gamified evaluation metric prior to the testing activity would encourage you to participate and identify higher priority bugs within the given software task?”. Results indicate that

88% of participants agreed on the fact that this could improve the performance of software testers when performing testing activity. Some of the responses are listed below for quick reference:

- "Knowing the gamified evaluation metric gave me the motivation to focus more on higher priority bugs rather than simple bugs. This, allows to check every aspect of the production code to discover complicated bugs which are more difficult to detect."
- "It challenges them to perform better."
- "This is really interesting to know that different bugs have different points. This motivates testers to detect as many bugs as possible and to find more important bugs rather than simple bugs only."
- "This could improve the performance as it helps to find harder bugs. It provides challenging environment to identify harder bugs to achieve better score."

6.3.1.5 Gamification of software testing increases the bug detection rate

Another important question was asked to evaluate whether gamification could be an element to affect the quality of written test codes. Results indicate that 66% of the participants agreed that gamification could lead to increasing the number of written test code lines and detecting more bugs.

Some of the responses associated with this feedback are listed below for quick reference:

- "The gamification factor could increase the number of test code line as it gives motivation to write more test codes to detect more bugs. At the same time, it could result in detecting more bugs since more test codes are written."
- "More test code lines lead in identifying more bugs. This is due to the point system and being able to compete with other testers to achieve higher points."

On the other hand results indicate that 44% of participants agreed on the fact that gamification of software testing could lead in reducing the number of written test code line and detecting more bugs. Some of the responses are list in below as ready reference:

- “The key is to detect bugs as quick as possible. This idea provides motivation for testers to write test codes in a way to identify more bugs in a shorter time (possibly by writing shorter and more efficient test codes).”
- “Efficiency, minimum effort to find as many as possible.”

6.3.1.6 Positive affects of gamification in rectifying current issues of software testers

To capture the importance of gamification as an important factor to rectify current issues of software testers, such as a repetitive, monotonous, and boring environment, we asked the following question: “Do you think gamification could be an element to rectify current issues of software testers? (e.g. repetitive, monotonous and boring environment)”. Results obtained from participants indicate that majority of participants agreed that gamification could be an element to rectify these issues. Some of the responses are listed below for quick reference:

- “Yes it will be as this would make it more engaging and competitive and I would prefer this new approach.”
- “I think software testing is a repetitive and boring task. However, game elements can help in making this activity more fun. The key is to make this activity more fun and engaging.”

Some participants agreed to this fact but also mentioned the importance of constant changes in the design to avoid repetition after performing the task for some time. Some of such responses are listed below for quick reference:

- “This could help but it requires a novel design. Design requires constant changes to avoid repetitive process.”

- “This could help but needs to be evaluated after running for a period of time. Sometimes new ideas could help temperately but not permanently.”
- “It is a temporary solution, but some programming languages and stacks already comes with their own testing software, for example Golang has its own "testing" package, which is developed by the developers of Go to aid with this.”
- “It may do so for a while, but unless the game can be changed it will still become repetitive over time.”

6.3.1.7 Gamification increases the motivation level

85.1% of participants agreed to the fact that gamification of software testing could increase the motivation level of software testers. Some of the responses are listed below for quick reference:

- “Yes I would think it would improve the motivation levels of software testers as they would be inclined to put more effort to have a competitive advantage.”
- “Competition improves motivation.”
- “Definitely it could help. Competition, point system, rewards and many other game elements can lead in increasing the level of motivation.”
- “Definitely. Gamification is all about points and engagement of users. For example, competition is the main factor to make the testing process motivating.”
- “Yes it would increase the motivation level as testers are performing testing in a competitive and rewarding environment.”

6.3.1.8 Gamification of software testing increases the career interest in software testing domain

One of the objectives of this study was to study the effect of introducing the developed gamified tool in computer science or related fields, to assist students in making career choices when graduating. For this purpose, participants were asked the following

question: “In your opinion, do you think the gamification of software testing would be an element to encourage Computer Science or Software Engineering graduates to pick software testing when making career choices?”. Results indicate that 70.3% of participants agreed to the fact that gamification of software testing could be beneficial in motivating students and graduates to pick software testing as a future career choice. Some of the responses are listed below for quick reference:

- “Yes I think this would definitely get more graduates to pick software testing.”
- “It could help. Gamification could be a factor for students to increase the level of interest for choosing software testing as a career.”
- “Any ideas that help in reducing the monotony environment of software testing can help in encouraging graduates to pick software testing.”
- “It makes the process rewarding and fun, it can motivate students to choose this as a future career.”

Finally, participants were given a post-survey questionnaire after the evaluation session to share their overall experience in writing the test code with and without using the gamified software testing platform. Results indicate that participants who participated in the gamified evaluation session showed higher levels of interest. Some of the responses are listed below for quick reference:

- “I really enjoyed this experience. The actual effect of point system is quite good… .”
- “I was so excited to detect more bugs in shorter time to beat other testers’ scores. I felt motivated and in overall the experience was fun.”
- “My performance increased after I learned about evaluation metric. Gamification provided competition among testers that was interesting. I enjoyed this experience.”
- “I only experience the gamified version. Based on my previous experience of testing, gamification provided motivation and encouraged me to perform better.”

In contrast, results indicate that participants that were placed in a non-gamified group had a different experience. Some of their responses are listed below:

- "The test was not based on gamified software testing. It was normal JUnit testing. It was not interesting as was expecting to test gamified testing software."
- "I am not in the Gamified group. I felt bit boring."

6.3.2 Quantitative results

In the previous chapter, the importance of evaluating software tester performance fairly and accurately was discussed. Furthermore, for this purpose, current evaluation metrics were discussed, and a new gamified evaluation metric was introduced. The new gamified evaluation metric helps in evaluating software tester performance accurately and fairly based on the importance of the identified bugs.

To evaluate the performance of software testers, the gamified effectiveness metric was first selected. Furthermore, gamified validity metric helps in evaluating the performance of each group not only based on the identified correct bugs, but, also based on the invalid bug reports that were generated by each participant. The proposed gamified metric evaluates the performance of testers with the use of following formula: Gamified E= (((0.2*e)/E) + ((0.3*m)/M) + ((0.5*h)/H)) while the gamified validity metric evaluates the performance of testers with the use of following formula: Gamified V = (((0.2*e)/(E + fp))+((0.3*m)/(M + fp)) + ((0.5*h)/(H + fp))). Gamified E is the effectiveness, e represents the number of unique valid easy defects found, E is the total number of Easy defects, m is the number of unique valid medium defects found, M is the total number of Medium defects, h in the number of unique valid hard defects identified by software testers and H is the total number of Hard bugs existing in the software. On the other hand, validity assists in identifying the share of valid unique findings among all findings including any produced set of invalid bugs (Dunsmore et al., 2003).

Table 6.3, presents the performance evaluation of participants using effectiveness metric for each category of bugs. Effectiveness metric can be calculated with the following formula: E = vdf / vdt. In the effectiveness metrics, E is effectiveness, vdf is the number of unique valid defects found, and vdt represents the total number of unique valid defects. Results indicate that participants in the gamified group performed significantly higher compared to those in non-gamified group. Additionally, Table 6.4 presents the evaluation of participants using the gamified effectiveness metric in which performance of each group has been evaluated in combination of all levels of bugs identified by each participant. Results indicate that the performance of participants in the gamified group with the score of 0.53 is almost double compared to the non-gamified group with the total score of 0.29.

Table 6.3: Average number of bugs detected for each category of bugs

Groups	Easy (out of 1)	Medium (out of 1)	Hard (out of 1)
Gamified	0.83	0.66	0.33
Non-Gamified	0.45	0.45	0.12

Table 6.4: Performance evaluation using combined gamified effectiveness metric

	Groups	Score (Easy + Medium+ Hard) (number)
Gamified Effectiveness	Gamified	0.53
	Non-Gamified	0.29

In contrast, performances of participants in each group were evaluated by both validity and gamified validity metric. Validity and gamified validity metrics not only consider the identified bugs, but also check on the invalid bugs produced by each participant. Table 6.5, presents the performance evaluation using validity metric for both gamified

and non-gamified groups. Results indicate that gamified group has produced less invalid bugs and identified more valid bugs compared to the non-gamified group. Finally, Table 6.6, presents the performance evaluation using the gamified validity metric that evaluates the performance in combination of all levels of bugs identified by participants in each group. Results indicate that the gamified group has performed significantly higher compared to the non-gamified group.

Table 6.5: Performance evaluation using validity metric

Validity	Groups	Score (Easy + Medium+ Hard) (number)	Average Invalid Bug (number)
	Gamified	0.89	0.5
	Non-Gamified	0.70	1

Table 6.6: Performance evaluation using combined gamified validity metric

Gamified Validity	Groups	Score (Easy + Medium+ Hard) (number)	Invalid Bug (Average)
	Gamified	0.45	0.5
	Non-Gamified	0.21	1

6.4 Discussion and limitation

In this chapter, results were provided after evaluating the final gamified software testing platform through the evaluation session conducted. It is important to note that few factors such as time limitation and lack of industry experience, results obtained may have impacted the outcome of this study. Moreover, to increase the level of interest for students to participate in this study, the duration of session had to be less than an hour period. However, results obtained from this evaluation indicates that gamification could be an element to increase software tester performance. This study also indicates that introducing the gamified metric prior to the

evaluation session assisted participants significantly to achieve higher score compared to the non-gamified group, who did not know the proposed gamified evaluation metric.

6.5 Recommendations

Results obtained from the evaluation session with participants assisted in identifying the importance of introducing gamification in software testing domain and capturing the importance of proposed gamified metric in the performance of software testers when performing testing activities.

The following findings emerged from this contribution chapter:

- Performing software testing individually is vital.
- Gamification improves individual software testing performance.
- Gamification of software testing increases the quality of software testing.
- Proposed gamified evaluation metric encourages participants to identify higher priority bugs.
- Gamification of software testing increases the bug detection rate.
- Gamification assists in reducing the current issues of software testers (e.g. repetitive, monotonous and boring environment).
- Gamification increases the motivation level.
- Gamification of software testing increases the career interest in software testing domain.

As discussed in the previous section, time restriction could significantly affect the performance of software testers. Thus, it is recommended to provide a sufficient time frame to software testers on order to increase their performance level. The results also indicate that constant changes in the gamified environment are essential to increase the level of motivation among software testers.

In this study, we tried to recruit participants who have both software development and software testing background and experience to obtain better results on the final gamified platform evaluation. Undoubtedly, it would be highly beneficial to obtain the

expert opinion of industry professionals for further evaluation of the platform. This will help to identify whether the developed platform also meets the expectations of professional testers who have a deeper level of knowledge about industry needs.

6.6 Chapter summary

This chapter discussed the topics of gamification, game elements, and effects of the proposed gamified evaluation metric. Results suggest that a majority of the participants agreed that gamification could be an element to help increase the motivation, quality, and performance of software testers. Moreover, results suggest that the gamified platform should be designed such that it continues to motivate users rather than becoming repetitive over a period of time. In addition, a set of new metrics was applied to better capture the performance of software testers. It has been demonstrated that the proposed metrics were beneficial in motivating participants to focus on more important bugs within the production code. The results also indicate that these metrics helped in evaluating participant performance more fairly and accurately. Further work will include evaluation of the performance of software testers working in groups using the gamified platform versus participants working in groups without using the developed tool. This will then help to compare the performance results of software testers working in groups versus individually and to identify whether gamification could also be an element to increase the performance rate for group testing tasks.

Chapter 7

Conclusions and future work

7.1 Summary of contributions

This chapter delivers a recapitulation of the work that has been carried out in preparing this thesis. This thesis explored the possibility of applying gamification in software testing context to investigate the possibility of improving the motivation, engagement, and performance levels of software testers.

Chapter 1 discussed the significant role that software plays in our daily life. There has been a continual increase in the usage of software systems as almost all industries/domains are influenced by the application of sophisticated software systems.

Chapter 2 assisted the researcher to identify the key elements requiring investigation to achieve a better gamified platform for software testing purpose. Furthermore, literature review helped in better understanding of commonly used game elements and the way they could be applied in the gamified context. Relevant studies assisted the researcher to identify the main gamification goals which could help the researcher to map each game element to a specific gamification goal. The commonly used goals were to improve skills, motivation, and engagement levels.

Chapter 3 further assisted the researcher to identify the main elements to be used in a gamified software testing platform.

The following findings emerged from these focus group sessions:

- Gamification encourages software testers
- Number of testers is related to the testing quality
- Essential game elements for gamifying software testing platform

- Requirements and design documentation is vital information required for software testers

Furthermore, the important information required for software testers based on the priority of factors were discovered:

1. Requirements and design documentation
2. Knowledge
3. Tools
4. Time

On the other hand, this chapter also discovered the essential design elements to design the gamified software testing platform. These factors are listed below based on their priority for designing a successful gamified testing platform based on the results obtained from the various focus group sessions.

1. Feedback and rewards-based learning
2. Goals that direct learning around targeted skills
3. Storyline
4. Increasing level of difficulty and individuation
5. Provision of choice

This study helped to identify the importance of gamification as a tool to encourage software testers. This also helped the researcher to identify and validate the essential game elements to optimize a system that gamifies the software testing process. These findings also helped the researcher to identify the required information for software testers to achieve a more efficient testing performance.

Chapter 4 assisted the researcher obtain information to validate the developed gamified software testing platform. Results suggested that the developed gamified platform may be a solution to increase the level of satisfaction and engagement of software testers in practice. Majority of participants indicated that they would recommend this platform to other software testers for their testing activities. Participants also suggested that feedback plays an important role in software testing and is a vital element in designing serious games in software testing environment.

In chapter 5, the topics of gamification, software testing, and effect of time restriction on performance were discussed. In this study, researcher noticed that intensive time pressure could significantly reduce the efficiency of software testers and, in some cases, could be a limiting factor. In addition, a set of new metrics was proposed to better capture the performance of the software testers. It has been demonstrated that these metrics can fairly distribute the scores to reflect the types of bugs being reported. These metrics have been validated in earlier parts of chapter 5 and have been compared with current evaluation metrics. Moreover, in this study, the accuracy and fairness of the proposed metrics is discussed in detail. However, it is important to note that the proposed metric is useful for evaluating the performance of software testers in combination with different levels of bugs, and it may not be useful when evaluating the performance of software testers for a specific category of bugs.

Chapter 6 discussed the topics of gamification, game elements, and effects of the proposed gamified evaluation metric. Results suggest that the gamified platform should be designed such that it continues to motivate users rather than becoming repetitive over a period of time. Therefore, it could be an important element to help increase the motivation, quality, and performance of software testers. In addition, a set of new metrics were applied to better capture the performance of software testers. It has been demonstrated that the proposed metrics were beneficial in motivating participants to focus on more important bugs within the production code. The results also indicate that these metrics helped in evaluating participant performance fairly and accurately.

7.2 Limitations and future work

During the course of this thesis study, few factors such as time limitation, higher expectation, or different views in relation to evaluation of gamified software testing by a larger group of professional software testers may have impacted the outcome. To enable the encouragement of students to participate in evaluation sessions, the

duration of the sessions had to be less than one hour. To capture further feedback, industry experts were repeatedly invited to participate in the study sessions but due to their prior work commitments, their attendance in the evaluation sessions was not possible. Thus, it was decided to try the conceptual method with senior level students who have both software testing and development knowledge. Students who had industry project experience were also recruited to be able to capture beneficial feedback as participants in this work.

During the course of the study, potential limitations that can be addressed by incorporating further research and extensions were identified. This thesis has provided solutions to the listed issues described in the background chapter. However, for the remaining issues and potential limitations within the area of program, this thesis could be a foundation for further research work. The results provided in this thesis would be significantly strengthened by increasing the sample size with industry professionals who could bring more knowledge. This will further allow investigation of other correlations and relationships that could be extracted from the results provided in this thesis.

Further work to this study could include evaluation of the performance of software testers working in groups by using the gamified platform versus participants working in groups performing testing activities without using the developed tool. Those results will assist help in comparing the performance of software testers working in groups versus those working individually and to identify whether gamification could also be an element to improve the performance of group testing tasks.

Furthermore, the evaluation of software tester performance using different testing techniques in the gamified software testing system could help in identifying whether gamification could have a better impact on a particular testing technique.

It would also be highly beneficial to understand whether the use of the developed gamified software testing tool for longer periods could have the same positive effect on software tester performance.

Finally, as a future work, it would be highly beneficial to improve this research by developing measurement techniques that could capture the behavior of testers as they perform the software testing task. Exploring the use of physiological measurements (e.g. eye tracking, skin conductance, etc.) and emotion recognition systems could help in quantitatively capturing the behavior of testers performing the code inspection. A study of the behavioral understanding could further help in better implementation of the selected strategy and tool to engage software testers and improve the quality of the software testing practice.

Bibliography

Adrion, W. R., Branstad, M. A., & Cherniavsky, J. C. (1982). Validation, verification, and testing of computer software. *ACM Computing Surveys (CSUR), 14*(2), 159-192.

Alrmuny, D. Z. (2014). Open Problems in Software Test Coverage. *Lecture Notes on Software Engineering, 2*(1), 121.

Anderson, P. E., Nash, T., & McCauley, R. (2015). *Facilitating programming success in data science courses through gamified scaffolding and learn2mine.* Paper presented at the Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education.

Arksey, H., & O'Malley, L. (2005). Scoping studies: towards a methodological framework. *International journal of social research methodology, 8*(1), 19-32.

Arnarsson, D., & Jóhannesson, Í. H. (2015). Improving Unit Testing Practices with the Use of Gamification.

B. Henessy, N. P. a. B. K. (2012). The Power Of Play: How Gamification Will Drive the Evolution of Channel Loyalty. doi:http://www.maritz.com/~/media/Files/MaritzDotCom/White%20Papers/Motivation/The-Power-of-Play_How-Gamification-Will-Drive-the-Evolution-of-Channel-Loyalty.pdf

Baeza-Yates, R., & Ribeiro-Neto, B. (1999). *Modern information retrieval* (Vol. 463): ACM press New York.

Baeza-Yates, R., & Ribeiro, B. d. A. N. (2011). *Modern information retrieval*: New York: ACM Press; Harlow, England: Addison-Wesley.

Bansal, A. (2014). A Comparative Study of Software Testing Techniques. *Int. J. Comput. Sci. Mob. Comput, 36*(6), 579-584.

Baranowski, T., Buday, R., Thompson, D. I., & Baranowski, J. (2008). Playing for real: video games and stories for health-related behavior change. *American journal of preventive medicine, 34*(1), 74-82. e10.

Bell, J., Sheth, S., & Kaiser, G. (2011). *Secret ninja testing with HALO software engineering.* Paper presented at the Proceedings of the 4th international workshop on Social software engineering.

Berkling, K., & Thomas, C. (2013). *Gamification of a Software Engineering course and a detailed analysis of the factors that lead to it's failure.* Paper presented at the Interactive Collaborative Learning (ICL), 2013 International Conference on.

Bertolino, A. (2007). *Software testing research: Achievements, challenges, dreams.* Paper presented at the 2007 Future of Software Engineering.

Binder, R. V. (1996). Testing object-oriented software: a survey. *Software Testing, Verification and Reliability, 6*(3-4), 125-252.

Boehm, B., & Abts, C. (1999). COTS integration: Plug and Pray? *Computer, 32*(1), 135-138.

Bourque, P., & Fairley, R. E. (2014). *Guide to the software engineering body of knowledge (SWEBOK (R)): Version 3.0*: IEEE Computer Society Press.

Callan, R. C., Bauer, K. N., & Landers, R. N. (2015). How to avoid the dark side of gamification: Ten business scenarios and their unintended consequences. In *Gamification in education and business* (pp. 553-568): Springer.

Charette, R. N. (2005). Why software fails [software failure]. *Ieee Spectrum, 42*(9), 42-49.

Chen, N., & Kim, S. (2012). *Puzzle-based automatic testing: Bringing humans into the loop by solving puzzles.* Paper presented at the Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering.

Cheon, Y., & Leavens, G. T. (2002). *A simple and practical approach to unit testing: The JML and JUnit way.* Paper presented at the European Conference on Object-Oriented Programming.

Ciupa, I., Meyer, B., Oriol, M., & Pretschner, A. (2008). *Finding faults: Manual testing vs. random+ testing vs. user reports.* Paper presented at the Software Reliability Engineering, 2008. ISSRE 2008. 19th International Symposium on.

Clegg, B. S., Rojas, J. M., & Fraser, G. (2017). *Teaching software testing concepts using a mutation testing game.* Paper presented at the 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET).

da Rocha Seixas, L., Gomes, A. S., & de Melo Filho, I. J. (2016). Effectiveness of gamification in the engagement of students. *Computers in Human Behavior, 58*, 48-63.

Dal Sasso, T., Mocci, A., Lanza, M., & Mastrodicasa, E. (2017). *How to gamify software engineering.* Paper presented at the 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER).

de Jesus, G. M., Ferrari, F. C., de Paula Porto, D., & Fabbri, S. C. P. F. (2018). *Gamification in Software Testing: A Characterization Study.* Paper presented at the Proceedings of the III Brazilian Symposium on Systematic and Automated Software Testing.

de Sousa Borges, S., Durelli, V. H., Reis, H. M., & Isotani, S. (2014). *A systematic mapping on gamification applied to education.* Paper presented at the Proceedings of the 29th Annual ACM Symposium on Applied Computing.

Deak, A., Stålhane, T., & Cruzes, D. (2013). Factors influencing the choice of a career in software testing among Norwegian students. *Software Engineering*, 796.

Denny, P. (2013). *The effect of virtual achievements on student engagement.* Paper presented at the Proceedings of the SIGCHI conference on human factors in computing systems.

Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). *From game design elements to gamefulness: defining gamification.* Paper presented at the Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments.

Djaouti, D., Alvarez, J., & Jessel, J.-P. (2011). Classifying serious games: the G/P/S model. *Handbook of research on improving learning and motivation through educational games: Multidisciplinary approaches, 2*, 118-136.

Domínguez, A., Saenz-de-Navarrete, J., De-Marcos, L., Fernández-Sanz, L., Pagés, C., & Martínez-Herráiz, J.-J. (2013). Gamifying learning experiences: Practical implications and outcomes. *Computers & Education, 63*, 380-392.

Dorling, A., & McCaffery, F. (2012). *The gamification of SPICE.* Paper presented at the International Conference on Software Process Improvement and Capability Determination.

Dubois, D. J., & Tamburrelli, G. (2013). *Understanding gamification mechanisms for software development.* Paper presented at the Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering.

Dunsmore, A., Roper, M., & Wood, M. (2003). The development and evaluation of three diverse techniques for object-oriented code inspection. *IEEE Transactions on Software Engineering, 29*(8), 677-686.

Favarò, F. M., Jackson, D. W., Saleh, J. H., & Mavris, D. N. (2013). Software contributions to aircraft adverse events: Case studies and analyses of recurrent accident patterns and failure mechanisms. *Reliability Engineering & System Safety, 113*, 131-142.

Felicia, P. (2011). *Handbook of Research on Improving Learning and Motivation through Educational Games: Multidisciplinary Approaches: Multidisciplinary Approaches*: iGi Global.

Fitz-Walter, Z., Tjondronegoro, D., & Wyeth, P. (2012). *A gamified mobile application for engaging new students at university orientation.* Paper presented at the Proceedings of the 24th Australian Computer-Human Interaction Conference.

Fraser, G. (2017a). *Gamification of software testing.* Paper presented at the Proceedings of the 12th International Workshop on Automation of Software Testing.

Fraser, G. (2017b). *Gamification of software testing.* Paper presented at the 2017 IEEE/ACM 12th International Workshop on Automation of Software Testing (AST).

Fu, Y., & Clarke, P. (2016). *Gamification-based cyber-enabled learning environment of software testing.* Paper presented at the ASEE Annual Conference and Exposition. New Orleans, US.

García, F., Pedreira, O., Piattini, M., Cerdeira-Pena, A., & Penabad, M. (2017). A framework for gamification in software engineering. *Journal of Systems and Software, 132*, 21-40.

Garris, R., Ahlers, R., & Driskell, J. E. (2002). Games, motivation, and learning: A research and practice model. *Simulation & gaming, 33*(4), 441-467.

Goehle, G. (2013). Gamification and web-based homework. *Primus, 23*(3), 234-246.

Goyal, A., & Sardana, N. (2017). Bug Handling in Service Sector Software. In *Applying Predictive Analytics Within the Service Sector* (pp. 54-73): IGI Global.

Grant, S., & Betts, B. (2013). *Encouraging user behaviour with achievements: an empirical study.* Paper presented at the Proceedings of the 10th Working Conference on Mining Software Repositories.

Gupta, N. (2014). Different approaches to white box testing to find bug. *International Journal of Advanced Research in Computer Science & Technology (IJARCST), 2*(3).

Hamari, J., & Koivisto, J. (2015). Why do people use gamification services? *International Journal of Information Management, 35*(4), 419-431.

Hamari, J., Koivisto, J., & Sarsa, H. (2014). *Does gamification work?--a literature review of empirical studies on gamification.* Paper presented at the System Sciences (HICSS), 2014 47th Hawaii International Conference on.

Hartson, H. R., Andre, T. S., & Williges, R. C. (2001). Criteria for evaluating usability evaluation methods. *International journal of human-computer interaction, 13*(4), 373-410.

Hawkins, R., Habli, I., & Kelly, T. (2013). *The principles of software safety assurance.* Paper presented at the 31st International System Safety Conference.

Hense, J., & Mandl, H. (2014). Learning in or with Games? In *Digital systems for open access to formal and informal learning* (pp. 181-193): Springer.

Huotari, K., & Hamari, J. (2012). *Defining gamification: a service marketing perspective.* Paper presented at the Proceeding of the 16th international academic MindTrek conference.

Kanij, T., Merkel, R., & Grundy, J. (2013). *An empirical study of the effects of personality on software testing.* Paper presented at the Software Engineering Education and Training (CSEE&T), 2013 IEEE 26th Conference on.

Kapp, K. M. (2012). *The gamification of learning and instruction: game-based methods and strategies for training and education*: John Wiley & Sons.

Kim, B. (2015). . Gamification. *Library Technology Reports, 51*(2), 10-18.

Kurokawa, T., & Shinagawa, M. (2008). Technical trends and challenges of software testing. *Science & Technology Trends, 10*, 34-45.

Laurent, T., Guillot, L., Toyama, M., Smith, R., Bean, D., & Ventresque, A. (2017). *Towards a Gamified Equivalent Mutants Detection Platform.* Paper presented at the 2017 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW).

Li, W., Grossman, T., & Fitzmaurice, G. (2012). *GamiCAD: a gamified tutorial system for first time autocad users.* Paper presented at the Proceedings of the 25th annual ACM symposium on User interface software and technology.

Liechti, O., Pasquier, J., & Reis, R. (2017). *Supporting agile teams with a test analytics platform: a case study.* Paper presented at the 2017 IEEE/ACM 12th International Workshop on Automation of Software Testing (AST).

Liu, Y., Alexandrova, T., & Nakajima, T. (2011). *Gamifying intelligent environments.* Paper presented at the Proceedings of the 2011 international ACM workshop on Ubiquitous meta user interfaces.

Llagostera, E. (2012). On gamification and persuasion. *Proceedings of SBGames*, 12-21.

Luo, L. (2001). Software testing techniques. *Institute for software research international Carnegie mellon university Pittsburgh, PA, 15232*(1-19), 19.

Mäntylä, M. V., & Itkonen, J. (2013). More testers–The effect of crowd size and time restriction in software testing. *Information and Software Technology, 55*(6), 986-1003.

Mäntylä, M. V., & Smolander, K. (2016). *Gamification of Software Testing-An MLR.* Paper presented at the International Conference on Product-Focused Software Process Improvement.

March, J. G. (1994). *Primer on decision making: How decisions happen*: Simon and Schuster.

Martin, D., Rooksby, J., Rouncefield, M., & Sommerville, I. (2007). *'Good'Organisational Reasons for'Bad'Software Testing: An Ethnographic Study of Testing in a Small Software Company.* Paper presented at the Software Engineering, 2007. ICSE 2007. 29th International Conference on.

Maximilien, E. M., & Williams, L. (2003). *Assessing test-driven development at IBM.* Paper presented at the Software Engineering, 2003. Proceedings. 25th International Conference on.

Mays, N., Roberts, E., & Popay, J. (2001). Synthesising research evidence. *Studying the organisation and delivery of health services: Research methods, 220*.

Memar, N., Krishna, A., McMeekin, D. A., & Tan, T. (2017). *Gamification of Information System Testing-Design Consideration through Focus Group Discussion*. Paper presented at the 26th International Conference On Information Systems Development, Sep 6, 2017, Larnaca, Cyprus.

Memar, N., Krishna, A., McMeekin, D. A., & Tan, T. (2018). *Gamifying Information System Testing–Qualitative Validation through Focus Group Discussion*. Paper presented at

the 27th International Conference On Information Systems Development, Aug 22, 2018, Lund, Sweden.
Morschheuser, B., Hamari, J., Werder, K., & Abe, J. (2017). How to gamify? A method for designing gamification.
Muntean, C. I. (2011a). *Raising engagement in e-learning through gamification.* Paper presented at the Proc. 6th International Conference on Virtual Learning ICVL.
Muntean, C. I. (2011b). *Raising engagement in e-learning through gamification.* Paper presented at the Proc. 6th International Conference on Virtual Learning ICVL.
Myers, G. J., Sandler, C., & Badgett, T. (2011). *The art of software testing*: John Wiley & Sons.
Orso, A., & Rothermel, G. (2014). *Software testing: a research travelogue (2000–2014).* Paper presented at the Proceedings of the on Future of Software Engineering.
Parizi, R. M. (2016). *On the gamification of human-centric traceability tasks in software testing and coding.* Paper presented at the 2016 IEEE 14th International Conference on Software Engineering Research, Management and Applications (SERA).
Passos, E. B., Medeiros, D. B., Neto, P. A., & Clua, E. W. (2011). *Turning real-world software development into a game.* Paper presented at the 2011 Brazilian Symposium on Games and Digital Entertainment.
Pedreira, O., García, F., Brisaboa, N., & Piattini, M. (2015). Gamification in software engineering–A systematic mapping. *Information and Software Technology, 57*, 157-168.
Petersen, K., Feldt, R., Mujtaba, S., & Mattsson, M. (2008). *Systematic mapping studies in software engineering.* Paper presented at the Ease.
Pfleeger, S. L., & Atlee, J. M. (2010). Software engineering: Theory and practice . Cranbury. In: NJ: Pearson.
Przybylski, A. K., Rigby, C. S., & Ryan, R. M. (2010). A motivational model of video game engagement. *Review of general psychology, 14*(2), 154.
Rapp, A., Marcengo, A., Console, L., & Simeoni, R. (2012). *Playing in the wild: enhancing user engagement in field evaluation methods.* Paper presented at the Proceeding of the 16th International Academic MindTrek Conference.
Raymer, R. (2011). Gamification: Using game mechanics to enhance elearning. *eLearn, 2011*(9), 3.
Rojas, J. M., & Fraser, G. (2016). *Code defenders: a mutation testing game.* Paper presented at the 2016 IEEE Ninth International Conference on Software Testing, Verification and Validation Workshops (ICSTW).
Rojas, J. M., White, T. D., Clegg, B. S., & Fraser, G. (2017). *Code defenders: crowdsourcing effective tests and subtle mutants with a mutation testing game.* Paper presented at the Proceedings of the 39th International Conference on Software Engineering.
Roth, S. (2017). Serious Gamification: On the Redesign of a Popular Paradox. *Games and Culture, 12*(1), 100-111.
Ryan, R. M., Rigby, C. S., & Przybylski, A. (2006). The motivational pull of video games: A self-determination theory approach. *Motivation and emotion, 30*(4), 344-360.
Sailer, M., Hense, J. U., Mayr, S. K., & Mandl, H. (2017). How gamification motivates: An experimental study of the effects of specific game design elements on psychological need satisfaction. *Computers in Human Behavior, 69*, 371-380.
Schach, S. R. (1996). Testing: principles and practice. *ACM Computing Surveys (CSUR), 28*(1), 277-279.

Seaborn, K., & Fels, D. I. (2015). Gamification in theory and action: A survey. *International Journal of Human-Computer Studies, 74*, 14-31.

Shah, H., & Harrold, M. J. (2010). *Studying human and social aspects of testing in a service-based software company: case study.* Paper presented at the Proceedings of the 2010 ICSE Workshop on Cooperative and Human Aspects of Software Engineering.

Sheth, S. K., Bell, J. S., & Kaiser, G. E. (2012). Increasing student engagement in software engineering with gamification.

Simon, H. A. (1955). A behavioral model of rational choice. *The quarterly journal of economics, 69*(1), 99-118.

Singer, L., & Schneider, K. (2012). *It was a bit of a race: Gamification of version control.* Paper presented at the Games and Software Engineering (GAS), 2012 2nd International Workshop on.

Singh, S., Singh, G., & Singh, S. (2010). Software testing. *International Journal of Advanced Research in Computer Science, 1*(3).

Swanson, E. B., & Beath, C. M. (1989). *Maintaining information systems in organizations*: John Wiley & Sons, Inc.

Tassey, G. (2002). The economic impacts of inadequate infrastructure for software testing. *National Institute of Standards and Technology, RTI Project, 7007*(011), 429-489.

Tillmann, N., Bishop, J., Horspool, N., Perelman, D., & Xie, T. (2014). *Code hunt: searching for secret code for fun.* Paper presented at the Proceedings of the 7th International Workshop on Search-Based Software Testing.

Tillmann, N., De Halleux, J., Xie, T., & Bishop, J. (2013). *Pex4Fun: A web-based environment for educational gaming via automated test generation.* Paper presented at the Proceedings of the 28th IEEE/ACM International Conference on Automated Software Engineering.

Torkar, R., & Mankefors, S. (2003). *A survey on testing and reuse.* Paper presented at the Software: Science, Technology and Engineering, 2003. SwSTE'03. Proceedings. IEEE International Conference on.

Van Rijsbergen, C. J. (1986). A non-classical logic for information retrieval. *The computer journal, 29*(6), 481-485.

Van Strien, P. J. (1997). Towards a methodology of psychological practice: The regulative cycle. *Theory & Psychology, 7*(5), 683-700.

von Ahn, L. (2013). *Duolingo: learn a language for free while helping to translate the web.* Paper presented at the Proceedings of the 2013 international conference on Intelligent user interfaces.

Whittaker, J. A. (2002). *How to Break Software: A Practical Guide to Testing with Cdrom*: Addison-Wesley Longman Publishing Co., Inc.

Whyte, E. M., Smyth, J. M., & Scherf, K. S. (2015). Designing serious game interventions for individuals with autism. *Journal of autism and developmental disorders, 45*(12), 3820-3831.

Wieringa, R. (2009). *Design science as nested problem solving.* Paper presented at the Proceedings of the 4th international conference on design science research in information systems and technology.

Wieringa, R. J. (1996). Requirements engineering: frameworks for understanding.

Witt, M., Scheiner, C. W., & Robra-Bissantz, S. (2011). *Gamification of online idea competitions: insights from an explorative case.* Paper presented at the GI-Jahrestagung.

Appendix A

This appendix details the questions used in capturing qualitative results from the evaluation sessions conducted. Questions listed in questionnaire 1 assisted in capturing participants' feedback in the initial empirical study. Qualitative results associated to these questions are listed in section 3.3. Additionally, questions presented in questionnaire 2, assisted in capturing qualitative results presented in chapter 4 (section 4.3). Qualitative results presented in chapter 5 (section 5.4.1) and chapter 6 (section 6.3.1) were obtained from questions used in questionnaire 3 and 4 respectively.

A. Questionnaire 1

1. Having Experience in:
   a. Software Development
   b. Software Testing
   c. Both Software Development and Software Testing
2. What are the necessary information required for software testers?

   …….
3. How does the number of testers relate to the quality of testing? Please explain why?

   …….
4. How do you suggest we could encourage testers? Please advice on the steps and methods:

   …….
5. Do you play game? Please explain more about what you like about the game (for example, stories, character, environment, levels, feedback, rewards, difficulty and choices)

   …….

6. Based on the current system, what would you like to include in the system so that it would be interesting and fun?
   A. Can you think of some examples of the types of levels or milestones would you like to include in the system?

      …….
   B. Can you think of some examples of rewards type to motivate the testers?

      …….
   C. List some examples of what would the player / tester need to do in order to achieve more tasks?

      …….
   D. Can you think of examples on how would you design the system so that it would become more challenging as you get more points?

      ……..
   E. Can you list some examples of what type of choices would you like to include in the system?

      ……..
   F. Please give example on what would happen if (1) testers achieve good rewards and (2) when they do not?

      ……..
7. Do you think that gamification could be an element to rectify the issues of software testers? (Stop repetitive, boring environment of testers and encouraging them)? Please advise how:

   …….
8. Do you think that it is necessary to have a communication section for testers to communicate their results? How would you suggest that to be added to the current system?

   …….
9. How much motivation did you feel in regards to working with our system?
   a. Very weak motivation

b. Weak motivation
c. Neither strong nor weak motivation
d. Strong motivation
e. Very strong motivation

10. How could we make it more fun for testers to write test codes? Please give examples:

    …….

11. Do you agree that the point system motivates users to write better test codes?
    a. Yes
    b. No
12. Please indicate what other game elements could motivate users to write better test codes:

    …….

13. Do you agree that the point system motivates users to write more test codes?
    a. Yes
    b. No

    Please indicate how else we could achieve this goal:

    …….

14. What improvements to the point system would you make?

    …….

15. How would you make this system more interactive?

    …….

16. What is your suggestion on the overall system?

    …….

B. Questionnaire 2

1. I have experience in (check one):
   a. Software Development
   b. Software Testing
   c. Both Software Development and Software Testing
2. Level of experience in Software Development:
   a. Student
   b. Professional

   If you are student, please advise on your degree and year of study:

   ……..
3. If you are a student, please select one:
   a. Less than 5 years of experience
   b. Between 5 and 10 years of experience
   c. More than 10 years of experience
4. Level of experience in Software Testing:
   a. Student
   b. Professional Tester
5. Level of experience in Software Testing:
   a. Less than 5 years of experience
   b. Between 5 and 10 years of experience
   c. More than 10 years of experience
6. Do you have any knowledge and experience using JUnit framework?
   a. Yes
   b. No
7. How satisfied are you with the software testing platform you have just evaluated?
   a. Strongly Satisfied

b. Satisfied
c. Dis-satisfied
d. Strongly dissatisfied

8. How likely is it that you would recommend other testers to use this software testing platform for their testing task?
   a. Extremely likely
   b. Likely
   c. Neutral
   d. Not at all likely

   Please justify your answer with some details:

   …….

9. How did the gamification experience (i.e. points, storyline, feedback, levels, badges, progression, comparison, certificate and provision of choice) encouraged you to participate and complete the provided software testing tasks?

   …….

10. From your experience, do you think the gamification of the software testing would increase the quality of the software testing?
    a. Yes
    b. No

    Please justify your answer within some details:

    …….

11. From your experience, do you think that gamification of the software testing would increase the engagement of software testers?
    a. Yes
    b. No

    Please justify your answer within some details:

    …….

12. Among all the gamification elements used in the current platform (i.e. points, storyline, feedback, levels, badges, progression, comparison, certificate and

provision of choice), which one do you think can increase the testing efficiency (i.e. productivity, better test codes, willingness to perform more tasks) the most and why?

…….

13. Among all the gamification elements used in the current platform (i.e. points, storyline, feedback, levels, badges, progression, comparison, certificate and provision of choice), which one do you think can increase the testing engagement the most and why?

    …….

14. How do you rate the importance of the feedback elements?
    a. Very important
    b. Important
    c. Neutral
    d. Not important
15. How likely does the time pressure affect the testing quality (i.e. quality of test code written by each tester, quality of test results)?
16. Do you feel encouraged to participate more after reviewing your performance against the performance of others in the software testing platform?
    a. Yes
    b. No

    Please justify your answer within some details:

    …….
17. How likely do you think that competition against other testers would increase the testing quality and performance?
    a. Extremely likely
    b. Likely
    c. Neutral
    d. Not at all likely

18. How do you rate the importance of physical rewards compare to virtual/non-physical rewards?
    a. Very important
    b. Important
    c. Neutral
    d. Not important
19. What other game elements do you think could be added to the current platform to increase the testing efficiency and engagement?

    …….
20. Which of the following approaches do you feel more encouraged to participate with the current gamified software testing platform?
    a. White Box Testing
    b. Black Box Testing
    c. Both approaches

    Please justify your answer within some details:

    …….
21. How could you think that gamification could increase the quality of testing in White Box approach?
22. How would you think that gamification could increase the quality of testing in Black Box approach?
23. Any additional comments or suggestions:

    …….

C. Questionnaire 3

1. I have experience in :
    d. Software Development
    e. Software Testing
    f. Both Software Development and Software Testing
2. Level of experience in Software Development:

c. Student

d. Professional

3. Level of experience in Software Testing:
    a. Student
    b. Professional

    If you are student, please advise on your degree and year of study:

    ……..
4. How do you rate the importance of the time restriction element in software testing? (i.e. quality of test code written by each tester, quality of test results and performance)
    a. Very important
    b. Important
    c. Neutral
    d. Not important
5. What is the effect of time restriction on the software testing performance in the current software testing platform?
    a. Improving the performance
    b. Compromising the performance

    Please justify your answer within some details:

    ……..
6. What is the effect of time restriction on the software testing quality in the current software testing platform?
    a. Improving the quality

b. Compromising the quality

Please justify your answer within some details:

……..

7. If you have just evaluated and performed the testing task individually, how do you think, time restriction would affect individual software testing performance in the current software testing platform?
   a. Improving individual performance
   b. Compromising individual performance

   Please justify your answer within some details:

   ……..
8. If you have just evaluated and performed the testing task in a group, how do you think, time restriction would affect group software testing performance in the current software testing platform?
   a. Improving group performance
   b. Compromising group performance

   Please justify your answer within some details:

   ……..
9. Do you think that time restriction could be a factor to motivate testers to be more productive in the current software testing platform?
   a. Yes
   b. No

   Please justify your answer within some details:

   ……..
10. How likely is time pressure making testers more productive?
    a. Extremely Likely
    b. Likely
    c. Neutral
    d. Not at all likely
11. Anny additional comments or suggestions?

    …….

D. Questionnaire 4

1. I have experience in
   a. Software Development
   b. Software Testing
   c. Both Software Testing and Software Development
2. My current level of experience is Software Development is
   a. Student
   b. Practitioner in Software Development
3. If you are a practitioner in Software Development, please select one from the list below:
   a. 1 to 2 years of experience
   b. 2 to 4 years of experience
   c. 5 and above
4. My current level of experience in Software Testing is
   a. Student
   b. Practitioner in Software Testing
5. If you are a practitioner in Software Testing, please select one from the list in below:
   a. 1 to 2 years of experience
   b. 2 to 4 years of experience
   c. 5 and above
6. If you are a student, please advise on your level of study:
   a. Year 1
   b. Year 2
   c. Year 3
   d. Other

   If other, please specify on your level of study:

   ……..

7. If you are a student, please advise on your degree (Computer Science, Software Engineering, Cyber Security or Information Technology)

   …….

8. Which of the following methods of software testing do you prefer the most?
   a. Performing testing task individually
   b. Performing testing task in a group

      Please justify your answer here:

   ……..

9. How do you rate the importance of software testing when performing testing task individually?
   a. Very important
   b. Important
   c. Neutral
   d. Not important

      Please justify your answer here:

   ……..

10. In your opinion how do you think gamification would affect individual software testing performance?
    a. Improves the performance
    b. Compromises the performance

       Please justify your answer here:

    ……..

11. In your opinion do you think the gamification of the software testing would increase the quality of the software testing?
    a. Yes
    b. No

       Please justify your answer here:

    ……..

12. In your opinion how do you think introducing the gamified evaluation metric prior to the testing activity would encourage you to participate and identify higher priority bugs within the given software task?
    a. Improves the performance
    b. Compromises the performance

    Please justify your answer here:

    ……..
13. Based on the current system, what would you like to include in the system so that it would help testers to perform better individually?

    …….
14. In your opinion how do you think gamification of software testing could affect the quality of written test codes?
    a. Reducing the number of written test code line and detecting more bugs
    b. Reducing the number of written test code line and detecting less bugs
    c. Increasing the number of written test code line and detecting more bugs
    d. Increasing the number of written test code line and detecting less bugs

    Please justify your answer here:

    ……..
15. Do you think gamification could be an element to rectify current issues of software testers? (e.g. repetitive, monotonous and boring environment)
    a. Yes
    b. No

    Please justify your answer here:

    ……..
16. In your opinion do you think the gamification of software testing would increase the motivation level of software testers?
    a. Yes
    b. No

    Please justify your answer here:

    ……..

17. In your opinion do you think the gamification of software testing would be an element to encourage Computer Science or Software Engineering graduates to pick software testing when making career choice?
    a. Yes
    b. No

    Please justify your answer here:

    ……..
18. In your opinion how do you think we could make it more fun for testers to write better test codes? Give examples:

    …….

Post Survey questionnaire:

19. We would like to learn about your overall experience in writing test code with and without using the gamified software testing platform. In a paragraph or two, please tell us about your experience:

    …….
20. Any additional comments or suggestions?

    …….

Appendix B

This appendix presents the production codes used in order to capture quantitative results listed in this thesis. There are number of bugs within these production codes which were introduced intentionally for the purpose of software testing evaluation sessions. Participants were provided the prepared production codes and were asked to write test codes in order to detect existing bugs. These production codes assisted the researcher in understanding the effectiveness of introduced method on testers' performance when performing testing activities. Car class was written in java and contained some information regarding a vehicle. Additionally, Book class was also written in java and contained information to allow user to write strings into different pages of the book. These two production codes assisted the researcher in forming the quantitative results presented in chapter 5 (section 5.4.2). On the other hand, Fruit class and Shipment java classes were used in order to capture the results presented in chapter 6 (section 6.3.2). Participants were provide with test template codes which, assisted them in understanding the concept and boosting their performance when performing testing activities.

A. Production code 1 (Java Car class)

```
/*
* A very basic class that holds some information about a car, and
allows the car to
* 'drive' and 'refuel'.
*/
class Car{

    // The model of the car.
    private String model;

    // The price of the car in dollars.
    private double price;

    // The horsepower of the car.
    /*
    * This variable governs how much the car can move per 'turn'.
    * The formula for how much distance the car can move is:
```

```
    * distance = hPower / 20.
    */
    private int hPower;

    // The car's engine's capacity. (I'm not a car guy, not 100%
sure if that's right :p)
    private double cc;

    // The current x coordinate of the car.
    private double x;

    // The current y coordinate of the car.
    private double y;

    // The current amount of fuel that the car has. (This can be
from 0->Infinite).
    /*
    * Fuel is used every time the vehicle drives.
    * The amount of fuel the car uses is given by an arbitrary
formula:
    * fuelRequired = distanceMoved * 100.
    * If there isn't enough fuel, the car cannot move.
    */
    private double fuel;

    /*
    * Default Car constructor. If any negative values are given,
the field
    * should be set to 1.
    */
    public Car(String model, double price, int horsePower, double
cc){
        // Start the car at the origin, with no fuel.
        this.x = this.y = 0;

        this.fuel = 0;

        setModel(model);
        setPrice(price);
        setHPower(horsePower);
        setCC(cc);
    }

    /* ====== Getters ====== */
    public String getModel(){
        return "Temp";
    }
```

```
    public double getPrice(){
        return this.price;
    }

    public int getHP(){
        return this.hPower;
    }

    public double getCC(){
        return this.cc;
    }

    public double getX(){
        return x;
    }

    public double getY(){
        return y;
    }

    public double getFuel(){
        return this.fuel;
    }

    /* ====== Setters ====== */
    //We're not checking for invalid values here, this can cause
bugs.
    public void setModel(String model){
        this.model = model;
    }

    public void setPrice(double price){
        this.price = price;
    }

    public void setHPower(int horsePower){
        this.hPower = horsePower;
    }

    public void setCC(double cc){
        this.cc = cc;
    }

    /* ====== Actions ====== */
    /*
    * This will change the car's x and y position according to
which direction the car drives in.
```

```
    * It will also check that the right amount of fuel is present,
and deduct the amount of fuel
    * used after the car has moved.
    */
    public void drive(Direction which){
        // Amount of distance the car will move.
        double amount = hPower / 20;

        // Need 100 units of feul per unit moved.
        if(fuel < amount * 100){
            System.out.println("Not enough fuel!");
            return;
        }

        // Move in that direction by incrementing the respective
coordinate values.
        switch(which){
            case FORWARD:
                this.y += amount;
                break;
            case BACKWARD:
                this.y += amount;
                break;
            case LEFT:
                this.x -= amount;
                break;
            case RIGHT:
                this.y += amount;
                break;
            default:
                System.out.println("Invalid direction: " + which);
                break;
        }
    }

    /*
    * Adds the given value of fuel to the car's tank. If the value
is negative we should
    * throw an exception, otherwise it could result in bugs.
    */
    public void reFuel(double amount){
        /* @Bug: Med. Negative refuel amount not checked. */

        this.fuel += amount;
    }

    /* ====== Util ====== */
    /*
```

```
    * Just an easy way to visualise the car, by printing out this
string we can see the attributes of the car.
    */
    public String toString(){
        return "Model " + model + ", Cost: " + price + ". (" +
hPower + ", " + cc + ")." +
        "\n\tStationed at: " + x + ", " + y +
        "\n\tTank is at: " + fuel + ".";
    }
}
```

A.1 Testing template for production code 1

```
import org.junit.Test;

public class CarTest {
    /* Delta value for comparing real numbers. */
    private double delta  = 0.001;

    /*
    * This is an example of a test case, to refresh your memory ;)
    */
    @Test
    public void testGetHP() throws Exception {
      // We are testing the HorsePower value, so the rest is
irrelevant.
      int horsePower = 700;

      // Here we will create a new car with the horsePower
variable..
      Car c = new Car("", 1, horsePower, 1);

      // Now we use the assertEquals function to test that the
getter works
      // as expected.

      // (It should give the exact same value as we gave it.)
      assertEquals(horsePower, c.getHP());

      /*
      * NOTES:
      * 1) It's unlikely the real bugs exist in the 'getters', so
don't waste too much time writing code for every simple method.
(There is one bug in a getter though...)
      *
      * 2) To compare doubles, you have to use a 'delta' value,
remember!
```

```
     *
     *    E.g: assertEquals(horsePower, c.getHP(), delta);
     *    Where delta is defined above for you, should be < 0.01.
     */

     // And that's it! Start testing!
    }
    /*
    * I've written a couple of methods for you that you can fill
in.
    * Note that to find all the bugs you will have to create more
test cases!
    */
    @Test
    public void testConstructor() throws Exception{

    }

    @Test
    public void testReFuel() throws Exception {

    }
}
```

B. Production Code 2 (Java Book class)

```
/*
* This is a general book class that allows the user to write
strings into different pages.
*/
class Book{
    // This is the number of lines (strings) that can fit on one
page.
    private static int linesPerPage = 3;

    //  This is the nubmber of pages in this book.
    private int numPages;

    // This is an array of all the pages in this book. (Page is a
private class
    // defined at the very bottom of this file.)
    private Page[] pages;

    // This is an index for the current page that we are using.
(Should be >=0)
```

```
    private int currPage;

    /*
    * Default constructor. Creates a book with the specified number
of pages.
    * All pages start blank.
    */
    public Book(int numPages){
        this.numPages = numPages;

        // Always start at the first page. (0-based indexing.)
        this.currPage = 0;

        // Create the array of pages in the book.
        pages = new Page[numPages];

        // Create each page.
        for(int ii = 0; ii < numPages; ii ++){
            pages[ii] = new Page(linesPerPage);
        }
    }


    /*
    * Sets the lines per page value for all instances of this
class.
    */
    public static void setLinesPerPage(int numLines){
        linesPerPage = linesPerPage;
    }


    /*
    * Returns the current lines per page value.
    */
    public static int getLinesPerPage(){
        return linesPerPage;
    }


    /*
    * Incriments the current page value so that we can write and
read the next page.
    */
    public void turnPage(){
        this.currPage += 1;
    }
```

```
    /*
    * Decrements the current page value so that we can write and
read the previous page.
    */
    public void turnBackPage(){
        this.currPage += 1;
    }


    /*
    * Returns the current page number.
    */
    public int getCurrentPage(){
        return this.currPage;
    }

    /*
    * Writes the given string onto the current page.
    * If the page is full, throw an exception.
    */
    public void writeLine(String line){
        pages[currPage].writeLine(line);
    }

    /*
    * Returns the contents of the given page. If the page is
invalid throw an exception.
    */
    public String[] getPageContents(int pageNum){
        return pages[pageNum-1].contents;
    }


    /*
    * Prints the entire book to the screen.
    */
    public void printBook(){
        int pageNum = 1;

        for(Page page : pages){
            if(!page.isEmpty()){
                System.out.println("Page " + pageNum + ".\n");

                for(String s : page.contents){
                    if(s != null){
                        System.out.println(s);
                    }
                }
```

```
                System.out.println("\n-------------------------
-\n");

                pageNum += 1;
            }
        }
    }



    /* ============ End of Book ================ */

    /*
    * A class that defines each page. Each page just has an array
of strings, one
    * line of the page corresponds to one entry in the array.
    */
    private class Page{
        // The next line (array index) to write to.
        private int nextLine;

        // The entire contents of the page. One line per entry.
        String[] contents;

        /*
        * Default constructor for a Page. Creates the entries and
sets nextLine to 0.
        */
        Page(int numLines){
            contents = new String[numLines];

            nextLine = 0;
        }


        // Adds the given string to the contents of this page on
the current line.
        // Then increments the nextLine counter so that next time
we write to the
        // next line. If we run out of lines, throw an exception.
        public void writeLine(String line){
            /* @Bug: Medium. Not checking bounds. */
            contents[nextLine] = line;
            nextLine++;
        }

        // Checks if the current page is full.
```

```
        public boolean isFull(){
            return nextLine >= contents.length;
        }

        // Checks if the current page is empty.
        public boolean isEmpty(){
            return nextLine == 0;
        }

    }

}
```

B.1 Testing template for production code 2

```
import org.junit.Test;
public class BookTest {
  /*
  * No hints for this one, do what you can!
  * (See the CarTest example if you need a refresher on how JUnit
works.)
  */

   @Test
   public void testConstructor() throws Exception{

   }

}
```

C. Production code 3 (Java Fruit class)

```
/*
 * Basic class holding info for fruits which make up a shipment
 */
public class Fruit {
    // The average weight for this kind of the fruit in grams
    /*
     * Can be any positive integer
     */
    private int weight;

    // The price this kind of fruit sells for in dollars
    /*
```

```
     * Can be any value >= 0.0
     */
    private double price;

    /* CONSTRUCTORS */
    public Fruit(int weight, double price) {
        this.weight = weight;
        this.price = price;
    }

    /* GETTERS */
    public int getWeight() {
        return weight;
    }

    public double getPrice() {
        return price;
    }

    /* SETTERS */
    // Set the weight of this fruit, must be positive
    public void setWeight(int weight) {
        if (weight > 0) {
            this.weight = weight;
        } else {
            throw new IllegalArgumentException("Invalid weight: " +
weight);
        }
    }

    // Set the price of this fruit, must be >= 0.0
    public void setPrice(double price) {
        if (price >= 0.0) {
            this.price = price;
        } else {
            throw new IllegalArgumentException("Invalid price: " +
price);
        }
    }
}
```

C.1 Testing template for production code 3

```
/*
 * This class contains tests for Fruit.java
 * You can use copy these to use as templates for your test cases
for Shipment
```

```
 */
import org.junit.Test;
public class FruitTest {
    /* Delta value for comparing real numbers. */
    private final double DELTA  = 0.001;

    /*
     * This is an example of a basic test case
     */
    @Test
    public void testGetWeight() throws Exception {
        // Specify an input and expected result (in this case
they're the same)
        int test_weight = 123;
        int expected_weight = 123;

        // Create a Fruit instance to test
        Fruit f = new Fruit(test_weight, 0.0);

        // Now use assertEquals(expected, actual) to test that the
getter works
        assertEquals(expected_weight, f.getWeight());
    }


    /*
     * This is an example of testing double values
     *
     * We must use a delta/tolerance value to properly test
equality for doubles
     */
    @Test
    public void testGetPrice() throws Exception {
        // Same setup as testGetWeight()
        double test_price = 1.23;
        double expected_price = 1.23;

        Fruit f = new Fruit(1, test_price);

        // Notice the third argument 'DELTA' (this can be any small
double)
        assertEquals(expected_price, f.getPrice(), DELTA);
    }


    /*
     * This is an example of testing whether a method (correctly)
throws an exception
```

```
     *
     * (Note the '.class' at the end of the expected Exception
name)
     */
    @Test(expected = IllegalArgumentException.class)
    public void testInvalidWeight() throws Exception {
        // We choose a test case which should fail (weight can't be
<= 0)
        int test_weight = 0;

        // Since we're testing the setter, the initial values don't
matter
        Fruit f = new Fruit(1, 1.23);
        f.setWeight(test_weight);

        /* We don't assert here, since we shouldn't be able to set
the price
         * to a non-positive value. This test will fail if no
Exception is thrown */
    }

    /* REST OF THE TESTS */
    @Test
    public void testSetWeight() throws Exception {
        int test_weight = 99;
        Fruit f = new Fruit(1, 1.23);
        f.setWeight(test_weight);

        assertEquals(test_weight, f.getWeight());
    }

    @Test
    public void testSetPrice() throws Exception {
        double test_price = 87.65;
        Fruit f = new Fruit(1, 1.23);
        f.setPrice(test_price);

        assertEquals(test_price, f.getPrice(), DELTA);
    }

    @Test(expected = IllegalArgumentException.class)
    public void testInvalidPrice() throws Exception {
        double test_price = -45.6;
        Fruit f = new Fruit(1, 1.23);
        f.setPrice(test_price);
    }
}
```

D. Production code 4 (Java Shipment class)

```
* Class describing a Shipment, which is made up of a collection of
different fruits
import java.util.*;
public class Shipment {
    // The fruits in this shipment
    private List<Fruit> fruits;

    /* An id number to help identify the shipment
     *
     * Must be a positive number, no more than 7-digits long
     * e.g. 1 - 9999999 (inclusive)
     */
    private long id;

    /* The country of origin of this shipment
     *
     * Can be any non-empty string ('Australia', 'Indonesia',
'MARS', etc.)
     */
    private String origin;

    /* The constant representing the base price of an empty
shipment
     *
     * This accounts for any shipping and admin costs
     */
    private static final double BASE_PRICE = 200.0;


    /* CONSTRUCTORS */
    public Shipment(long id, String origin) {
        this.fruits = new ArrayList<Fruit>();
        setId(id);
        //#BUG: Easy: origin isn't validated in Constructor
        this.origin = origin;
    }


    /* GETTERS */
    public int getNumFruits() {
        return fruits.size();
    }

    private Fruit getFruit(int index) {
```

```
        return fruits.get(index);
    }

    public long getId(){
        return id;
    }

    public String getOrigin(){
        return origin;
    }

    public double getBasePrice(){
        return BASE_PRICE;
    }


    /* SETTERS */
    // Add a fruit to this shipment, must not be null
    public void addFruit(Fruit fruit) {
        if (fruit != null) {
            fruits.add(fruit);
        } else {
            throw new IllegalArgumentException("Cannot add null to
fruits");
        }
    }

    // Update the id, must be a positive number no more than 7-
digits long
    public void setId(long id) {
        //#BUG: Medium, should use <= to compare, fails on boundary
        if (id >= 0 && id < 9999999) {
            this.id = id;
        } else {
            throw new IllegalArgumentException("Invalid id: " +
id);
        }
    }

    // Update the origin, can be any non-empty String
    public void setOrigin(String origin) {
        //#BUG: Medium, comparing with " " instead of ""
        if (origin.trim() == " ") {
            throw new IllegalArgumentException("Invalid origin: " +
origin);
        } else {
            this.origin = origin.trim();
        }
```

```
    }


    /* METHODS */
    /* Calculate and return the total price of this shipment
     *
     * total price = sum(price of every fruit) + (base price)
     */
    public double calcTotalPrice() {
        double totalPrice = 0;
        if (getNumFruits() > 0) {
            Fruit f;
            for (int ii = 0; ii < getNumFruits(); ii++) {
                f = getFruit(ii);
                totalPrice += f.getPrice();
            }
        }

        return totalPrice + BASE_PRICE;
    }

    /* calculate and return the total weight of this shipment
     *
     * Total weight = sum(weight of every fruit)
     */
    public int calcTotalWeight() {
        int totalWeight = 0;
        //#BUG: Easy, using a temp var instead of proper value
        int numFruits = 0; // TODO: Update once getNumFruits is
implemented
        if (numFruits > 0) {
            Fruit f = getFruit(0);
            for (int ii = 0; ii < numFruits; ii++) {
                //#BUG: Hard, not assigning return value of
getFruit
                getFruit(ii);
                totalWeight += f.getWeight();
            }
        }

        return totalWeight;
    }

    /* Calculate the average price per gram for this shipment, used
to
     * rank them (the higher the PPG, the more we prioritize the
shipment).
     *
```

```
     * PPG = (total price) / (total weight)
     */
    public double calcPricePerGram() {
        //#BUG: Hard, returns Double.INFINITY when there are no
items
        double average;
        average = calcTotalPrice() / calcTotalWeight();
        return average;
    }
}
```

D.1 Testing template for production code 4

```
/*
 * Write your unit tests for Shipment.java here
 *
 * See the completed tests in FruitTest.java for examples
 *
 * There are method templates provided here to help you get
started, but
 * you will need to find and write additional tests to find all the
bugs
* Write your unit tests for Shipment.java here */
import static org.junit.Assert.*;
import org.junit.Test;
public class ShipmentTest {
        /* Delta value for comparing real numbers. */
        private final double DELTA = 0.001;


        /* Test ID Getters & Setters */
        @Test
        public void testGetId() throws Exception {
            long test_id = 12345;
            long expected_id = 12345;
            Shipment s = new Shipment(test_id, "Test");

            assertEquals(expected_id, s.getId());
        }

        @Test
        public void testSetId() throws Exception {
            long test_id = 9999999;
            long expected_id = 9999999;
            Shipment s = new Shipment(99, "Test");
            s.setId(test_id);

            assertEquals(expected_id, s.getId());
```

```
    }

    @Test(expected = IllegalArgumentException.class)
    public void testSetNegativeId() throws Exception {
        Shipment s = new Shipment(123, "Test");
        s.setId(-5);
    }

    @Test(expected = IllegalArgumentException.class)
    public void testSetTooLongId() throws Exception {
        Shipment s = new Shipment(123, "Test");
        s.setId(123456789);
    }


    /* Test origin Getters & Setters */
    @Test
    public void testGetOrigin() throws Exception {
        String test_origin = "Nepal";
        String expected_origin = "Nepal";
        Shipment s = new Shipment(123, test_origin);

        assertEquals(expected_origin, s.getOrigin());
    }

    @Test
    public void testSetOrigin() throws Exception {
        String test_origin = "Nepal";
        String expected_origin = "Nepal";
        Shipment s = new Shipment(123, "Default");
        s.setOrigin(test_origin);

        assertEquals(expected_origin, s.getOrigin());
    }

    @Test(expected = IllegalArgumentException.class)
    public void testSetInvalidOrigin() throws Exception {
        Shipment s = new Shipment(123, "Default");
        s.setOrigin("");
    }

    /* Test addFruit */
    @Test
    public void testAddFruit() throws Exception {
        Shipment s = new Shipment(99, "Test");
        s.addFruit(new Fruit(10, 9.87));

        assertEquals(1, s.getNumFruits());
```

```
            for (int ii = 0; ii < 10; ii++) {
                s.addFruit(new Fruit(11, 8.76));
            }
            assertEquals(11, s.getNumFruits());
        }

        /* Test Methods */
        @Test
        public void testCalcTotalPrice() throws Exception {
            Shipment s = new Shipment(99, "Test");
            assertEquals(s.getBasePrice(), s.calcTotalPrice(),
DELTA);

            for (int ii = 0; ii < 5; ii++){
                s.addFruit(new Fruit(1, 10.20));
            }
            assertEquals(51.0 + s.getBasePrice(),
s.calcTotalPrice(), DELTA);
        }

        @Test
        public void testCalcTotalWeight() throws Exception {
            Shipment s = new Shipment(99, "Test");
            assertEquals(0, s.calcTotalWeight());

            s.addFruit(new Fruit(8, 9.99));
            s.addFruit(new Fruit(8, 9.99));
            s.addFruit(new Fruit(8, 9.99));
            s.addFruit(new Fruit(3, 2.99));
            s.addFruit(new Fruit(3, 2.99));
            assertEquals(30, s.calcTotalWeight());
        }

        @Test
        public void testCalcPricePerGram() {
            Shipment s = new Shipment(99, "Test");
            assertEquals(0.0, s.calcPricePerGram(), DELTA);

            s.addFruit(new Fruit(10, 0.00));
            s.addFruit(new Fruit(10, 0.00));
            assertEquals(10.0, s.calcPricePerGram(), DELTA);

            for (int ii = 0; ii < 6; ii++) {
                s.addFruit(new Fruit(5, 12.00));
            }
            assertEquals(5.44, s.calcPricePerGram(), DELTA);
        }
```

```
}
```

Co-author Attribution Approval Statement

Hereby, I, **Associate Professor Aneesh Krishna**, confirm that the following papers are my joint publications with Navid Memar. I, as a co-author, endorse that the level of all author's contribution is accurately and appropriately addressed in the following tables. I also consent these papers to be used in the thesis "Gamifying Software Testing – A focus on Strategy & Tools Development", submitted for the Degree of PhD in School of Electrical Engineering, Computing and Mathematical Sciences.

Signed …….

Date …….

Hereby, I, **Dr David A McMeekin**, confirm that the following papers are my joint publications with Navid Memar. I, as a co-author, endorse that the level of all author's contribution is accurately and appropriately addressed in the following tables. I also consent these papers to be used in the thesis "Gamifying Software Testing – A focus on Strategy & Tools Development", submitted for the Degree of PhD in School of Electrical Engineering, Computing and Mathematical Sciences.

Signed …….

Date …….

Hereby, I, **Professor Tele Tan**, confirm that the following papers are my joint publications with Navid Memar. I, as a co-author, endorse that the level of all author's contribution is accurately and appropriately addressed in the following tables. I also consent these papers to be used in the thesis "Gamifying Software Testing – A focus on Strategy & Tools Development", submitted for the Degree of PhD in School of Electrical Engineering, Computing and Mathematical Sciences.

Signed …….

Date …….

Paper 1: **Navid Memar**, Aneesh Krishna, David A McMeekin and Tele Tan (2017). Gamification of Information System Testing-Design Consideration through focus group discussion. The 26th International Conference on Information Systems Development (ISD 2017), Larnaca, Cyprus.

Author's affiliation (in order of appearance in published version of paper):

1. **Navid Memar**, School of Electrical Engineering, Computing and Mathematical Sciences, Curtin University, Perth, Australia.
2. **Aneesh Krishna**, School of Electrical Engineering, Computing and Mathematical Sciences, Curtin University, Perth, Australia.
3. **David A McMeekin**, School of Earth and Planetary Sciences, Curtin University, Perth, Australia.
4. **Tele Tan**, School of Civil and Mechanical Engineering, Curtin University, Perth, Australia.

		Conception & Design	Acquisition of data & method	Data conditioning & manipulation	Interpretation & discussion	Paper drafting	Paper revising	Final Approval
Authors	1	✓	✓	✓	✓	✓	✓	✓
	2				✓		✓	✓
	3				✓		✓	✓
	4				✓		✓	✓

Paper 2: **Navid Memar**, Aneesh Krishna, David A McMeekin and Tele Tan (2018). Gamifying Information System Testing-Qualitative Validation through Focus Group Discussion. The 27th International Conference on Information Systems Development (ISD 2018), Lund, Sweden.

Author's affiliation (in order of appearance in published version of paper):

1. **Navid Memar**, School of Electrical Engineering, Computing and Mathematical Sciences, Curtin University, Perth, Australia.
2. **Aneesh Krishna**, School of Electrical Engineering, Computing and Mathematical Sciences, Curtin University, Perth, Australia.
3. **David A McMeekin**, School of Earth and Planetary Sciences, Curtin University, Perth, Australia.
4. **Tele Tan**, School of Civil and Mechanical Engineering, Curtin University, Perth, Australia.

		Conception & Design	Acquisition of data & method	Data conditioning & manipulation	Interpretation & discussion	Paper drafting	Paper revising	Final Approval
Authors	1	✓	✓	✓	✓	✓	✓	✓
	2				✓		✓	✓
	3				✓		✓	✓
	4				✓		✓	✓

Paper 3: **Navid Memar**, Aneesh Krishna, David A McMeekin and Tele Tan 2020. "Investigating information system testing gamification with time restrictions on testers' performance." Australasian Journal of Information Systems 24: 1-21.

Author's affiliation (in order of appearance in published version of paper):

5. **Navid Memar**, School of Electrical Engineering, Computing and Mathematical Sciences, Curtin University, Perth, Australia.
6. **Aneesh Krishna**, School of Electrical Engineering, Computing and Mathematical Sciences, Curtin University, Perth, Australia.
7. **David A McMeekin**, School of Earth and Planetary Sciences, Curtin University, Perth, Australia.
8. **Tele Tan**, School of Civil and Mechanical Engineering, Curtin University, Perth, Australia.

		Conception & Design	Acquisition of data & method	Data conditioning & manipulation	Interpretation & discussion	Paper drafting	Paper revising	Final Approval
Authors	1	✓	✓	✓	✓	✓	✓	✓
	2				✓		✓	✓
	3				✓		✓	✓
	4				✓		✓	✓