An Experiment on the Effectiveness of Unit Testing when Introducing Gamification

Master's Thesis in Software Engineering

Marcus Johansson
Erik Ivarsson

Department of Computer Science & Engineering
Software Engineering
Chalmers University of Technology
Gothenburg, Sweden 2014

Master's Thesis 2014:1

An Experiment on the Effectiveness of Unit Testing when Introducing Gamification

MARCUS JOHANSSON,
ERIK IVARSSON

Examiner: MATTHIAS TICHY
Supervisor: ERIC KNAUSS

Chalmers University of Technology
Department of Computer Science and Engineering, SE-412 96, Gothenburg Sweden
Telephone +46 (0)31-772 10 00

Department of Computer Science and Engineering Gothenburg, Sweden June 2014

Abstract

Software testing has been associated with both technical and socio-technical difficulties. While software systems grow larger and are becoming more complex, there is simultaneously a problem with lack of motivation and interest in the software testing process. The purpose of this study is to research how the concept of Gamification, which has shown positive effects in varying areas of application, can be used to increase the effectiveness of unit testing, as well as increasing the motivation and interest among the software testers. The focus of the study was put on the number of defects found, as well as the percentage of requirements covered by tests when measuring effectiveness. The effect of Gamification in unit testing was researched through an experiment involving 24 subjects. The subjects were sampled into two groups, one group with Gamification applied to their testing process, and one group without Gamification applied to their testing process. The subjects conducted, during a one hour testing session, unit testing on a selected system, which had been seeded with a specific number of faults. The results show that the Gamification group found significantly more defects and received a significantly higher percentage of requirements covered by tests, than the Control group. Moreover, the results show that the Gamification group found the task significantly more interesting than the Control group. We conclude that, even if the significantly higher number of found defects and requirements covered by tests are hard to correlate to specific Gamification elements; Gamification might be used to improve the stated variables of effectiveness in software testing.

Acknowledgements

We would like to express our deepest gratitude to our supervisor, Eric Knauss, who has guided us through the work of the thesis. We thank Eric, not only for his useful remarks on the report, but also for his sharing of knowledge within the field, constant encouragement, and unabridged willingness to aid us during the work on the thesis. We would also like to thank all participants of the study for taking their time to participate in the experiment, as well as valuable discussions and input that followed after the experiment. Finally, we would like to thank Richard Torkar for sharing his knowledge and experience of conducting experiments within the field of software testing.

Marcus Johansson & Erik Ivarsson, Gothenburg 30/5/2014

Contents

1 Introduction **1**
1.1 Purpose of Study . . . 2
1.2 Statement of Problems . . . 3
1.3 Research Questions . . . 3
1.4 Scientific Contribution . . . 3
1.5 Disposition of the Report . . . 3

2 Background **5**
2.1 Software Testing . . . 5
2.1.1 Evaluation of Testing Techniques . . . 5
2.2 Requirement Coverage . . . 7
2.3 Gamification . . . 7
2.3.1 Software Engineering and Gamification . . . 9
2.3.2 Gamification and Psychology . . . 11
2.3.3 Framework for applying Gamification . . . 13
2.3.4 Game Elements and Game Mechanics . . . 14
2.4 Game-based Adaptations . . . 16
2.5 This thesis and previous research . . . 17

3 Method **18**
3.1 Scope . . . 18
3.1.1 Hypotheses . . . 19
3.1.2 Experiment variables . . . 19
3.1.3 Subjects . . . 20
3.2 Instrumentation . . . 21
3.2.1 Eclipse Plugin . . . 22
3.2.2 Intrinsic Motivation Inventory Questionnaire . . . 23
3.3 Operation . . . 24
3.4 Gathering of Data and Reviews . . . 25
3.4.1 Defect Detection Review . . . 25

3.4.2 Requirement Coverage Review 25
3.4.3 Test Coverage Review 28
3.4.4 Review of Invalid Test Cases 28

4 Results and Analysis **29**
4.1 Distributions of Data 29
4.2 Detection of Defects 30
4.3 Requirements Coverage 31
4.4 Testing Coverage 32
4.5 Invalid Test Cases 33
4.6 Results From the Survey 34
4.6.1 Perception of the Testing Session 34
4.6.2 Perception of the Gamification Elements 35

5 Discussion **39**
5.1 Research questions 39
5.2 Usage of the Plugin 40

6 Validity Threats **42**
6.1 Conclusion Validity 42
6.2 Internal Validity 42
6.3 Construct Validity 43
6.4 External Validity 43

7 Conclusion and Future Work **45**
7.1 Future Work 46

Bibliography **52**

Appendices **53**

A Plugin **54**
A.1 Standard Version 55
A.2 Gamification Version 55
A.2.1 Fictional Scenario 56
A.2.2 Badges 56
A.2.3 Point system 57
A.2.4 Leaderboard 58
A.2.5 Trace to Test Linkage Tool 58
A.2.6 Instant Feedback 58

B Questions for the Intrinsic Motivation Inventory **60**
B.1 General Information 60
B.2 General Questions (both groups) 61
B.3 Questions About the Task Performed (both groups) 62

B.4 Gamification Elements (gamification-group) 63
B.4.1 The Leaderboard 63
B.4.2 The Badges 64
B.4.3 The Trace To Test Linkage Tool 65
B.4.4 The Welcome Screen 66

C Histograms of Data From the Experiment **68**

1 Introduction

Software testing is a critical task in software development, from where the quality and reliability of the software strictly depends. It is important to test the software during development to check how well the software adheres to its requirements and to increase the developers' confidence that the software behaves in its intended manner. Software testing can take as much as 50 percent of the project time and cost [1], but although code testing occupies so much time and resources within software projects, it is still a major problem with code of low quality. In 2002 it was estimated that the annual cost of poor infrastructure for software testing alone in the United States was $22.2 billion to $59.5 billion, whereof half of the cost were borne by users in form of mitigation costs of poor software quality, and half of the cost was borne by software developers during development [2].

While simultaneous as being expensive, software testing is also being considered as a time-consuming, difficult and inadequate practice [3]. Testing is also a heuristic practice that can prove the presence of defects but not its absence[1], which might further demotivate programmers to not conduct testing during development. There is also arguments that software testing does not reach the maturity level that one would expect from an engineering discipline [4].

There are several reasons why software still can contain defects although it has been tested, two of them are that the practice of testing is not done to the desired extent because of time and budget constraints, or that the actual tests are of poor quality [5]. The importance of software quality and the challenges within software testing have resulted in a vast amount of research within software testing, and how the process of software testing might be improved. However, although much research has been conducted, it remains a big challenge to improve software quality, and simultaneously do it within the time and budget constraints imposed on projects [1].

A majority of the previous research has focused on the development of new tools, practices and software development processes [1], but since the problems within software quality remains, it might be other factors in play that work against the improvement of software quality. Such factors might be that developers within industry are slow to adopt testing in their development process, that they might skip testing because of time and budget constraints or because it is seen as an uninteresting practice.

Lawrence et al. [6] are ranking the absence or overlooking of a requirement as one of the greatest risks in requirements engineering. Keeping the reason for testing software in mind leads to the question what happens when a requirement is not tested enough, or not tested at all. This is directly connected to the fact that a certain aspect, functional or nonfunctional, of the system has not been tested. This might not be as crucial as missing the actual implementation of the requirement, but the absence of tests for a specific requirement can increase the risk of unwanted behavior of the system, and there are no guarantees that the program executes in the desired manner. A focus in testing of only a few of the stated features, can furthermore lead to an inconsistency in quality throughout the software. This can lead to some features being of good quality, while other features lacking in quality [6]; increasing the percentage of requirements covered by tests is one way to mitigate these problems.

A term that has received an increasing amount of attention during the last few years is Gamification. Gamification is a concept that aims to increase the users' motivation, involvement, and interest in a specific task. The most common definition is from [7] and defined as "... the use of game design elements in non-game contexts". The theory of Gamification has successfully been applied in different areas to deal with tasks that have been considered demotivating and been lacking user engagement, as well as increasing an already high motivation and user engagement among users or customers. The argumentation is that by introducing game elements and game mechanics, motivation and interest in performing a task can be increased, which in turn can improve the results of a performed activity or task.

1.1 Purpose of Study

The intention of the study is to investigate how Gamification can be used in software testing to increase the effectiveness of the testing process. The research consists of an experiment that is designed to compare a testing technique with Gamification applied to it, against the same testing technique without Gamification. From the existing research on Gamification there have been results confirming the idea that it increases the motivation, involvement, and interest for the activity [8][9][10][11]. Given the small amount of research of Gamification within software engineering practices, and especially within software testing, the study aims to analyze the possible effects of introducing Gamification in software testing. The analysis of the results from the experiment will allow for conclusions whether Gamification can be used to increase the effectiveness of software testing by increasing motivation and engagement among software testers.

1.2 Statement of Problems

The main problems that will be addressed in the research are how software testing might be improved by the application of Gamification to the testing process, regarding defect detection as well as percentage of requirements covered by tests. The research will investigate how the documented benefits of Gamification, regarding increasing motivation and user involvement, can be used to mitigate the stated issues affiliated with software testing.

1.3 Research Questions

To investigate and provide conclusions about how Gamification can be used to improve software testing, two research questions have been constructed. These research questions are directly connected to the possible effects that Gamification might have on the selected testing technique, in this case unit-testing.

RQ1: Is there a difference in the number of defects detected by testers when performing unit testing with Gamification applied, than without Gamification applied, in a given time frame?

RQ2: Is there a difference in the percentage of requirements covered by tests when performing unit testing with Gamification applied, than without Gamification applied, in a given time frame?

1.4 Scientific Contribution

The study aimed at connecting the field of software testing with the concept of Gamification and investigate what possible effects Gamification have on the effectiveness of unit testing. Before the study, there were no studies found about introducing Gamification in unit testing, or software testing, to measure the possible effect on the effectiveness with special regard to the number of defects found and the percentage of requirements covered by test cases. Therefore, this study should be seen as a first step towards connecting the mentioned fields, and provide a foundation of knowledge for future research within this field.

The results presented in the study suggest that Gamification can be used to increase the effectiveness of unit testing under the stated conditions. The authors hope that this study will lead the way for more experiments in controlled environments, and actual case studies in industry, regarding implementations of Gamification in testing processes.

1.5 Disposition of the Report

Chapter two introduces the reader to the background of the research by presenting past and current research within software testing, requirements coverage, and Gamification.

Chapter three presents the methodology used for the experiment, and how the study has been conducted. Chapter four presents the results from the experiment, and an analysis of the results. The results are discussed in Chapter five. Chapter six discuss threats to validity, and Chapter seven summarizes the study with conclusions, and discussion of future work.

2

Background

The following sections introduce the background of this study in the areas of software testing, requirement coverage, Gamification as well as game-based adaptations. The chapter also introduces the reader to the related work in the stated fields.

2.1 Software Testing

The activity of software testing can be seen from two perspectives: a technical and a socio-technical perspective [1]. Although it is easy to regard most phases of software engineering as purely technical, it is important to note that testing to a high degree is socio-technical [12]. Software testing is often seen as a monotonous [13], tedious and boring practice [14], as well as time-consuming, difficult and often inadequate [3]. The complexity and level of difficulty of software testing often perceived by developers might also partly aid in explaining the existent gap between testing in research and in practice [4], which consequently lead to the test suites being of poor quality [5]. Instead of relying on fact-based guiding when doing software testing, testers do instead rely on intuition, fashion, and market speak [15]. The difficulty of software testing is also expected to increase in the future, along with increased total costs of low software quality. This is a result of increasing complexity, increased usage of IT in society and a larger number of software critical systems [16]. By considering the socio-technical aspects of software testing, it becomes evident that skill, commitment, and motivation of software testers are vital for successful software testing [16].

2.1.1 Evaluation of Testing Techniques

The main objective of this study is to evaluate how Gamification applied to a testing technique, can improve the effectiveness of the testing, compared to the same testing technique without *Gamifiacation* applied. Vos et al. [17] and Eldh et al. [18] present two frameworks for evaluating different testing techniques regarding effectiveness, efficiency and applicability. The outline of the frameworks consists of four steps [17][18]:

1. Prepare code samples with a known number of injected faults.
2. Decide the hypotheses and proposition used in the evaluation/experiment.
3. Identify variables to measure
4. Collect and analyze data

The first step in such a framework according to Eldh et al. [18] is to prepare code samples with a known number of faults, which will be used to evaluate the testing techniques. The second step in the framework is to decide the hypotheses/propositions and which data to collect. Vos et al.[17] propose three concrete hypotheses for evaluating and comparing two testing techniques:

P_1 The testing technique T_1 is more effective in finding faults than the current testing technique T_0.

P_2 The testing technique T_1 is more efficient in finding faults than the current testing technique T_0.

P_3 The resources needed to use, implement and learn T_1 is worthwhile.

This study focuses on the effectiveness of software testing, hence, the efficiency, P_2, is disregarded in the analysis as well as the applicability, P_3. The third step involves identifying the independent and dependent variables that will be of use to either accept or reject the hypotheses P_1. The dependent variables represent the results of applying a testing technique and depends on the independent variables. The dependent variables presented by Vos et al.[17] to evaluate different testing techniques is *effectiveness* and *efficiency*. Examples of independent variables that affect the efficiency and effectiveness are the testing technique used, project setting and experiences of the testers[17].

There are several approaches to measure testing effectiveness and efficiency as proposed by Vos et al.[17] and Eldh et al.[18]. The measurements that are relevant for this study are the ones related to effectiveness, which are listed below.

1. Effectiveness
   (a) The number of valid test cases created.
   (b) The number of defects found by the testing technique T.
   (c) Test code coverage reached using the testing technique T.
   (d) Requirement coverage reached using the testing technique T.

The last step in the process is to analyze the data collected from the experiment to accept or reject the above hypotheses. To evaluate proposition P_1 we have to measure whether the testing method T_1 performs better than the current testing method in use for the variables 1b, 1c and 1d.

2.2 Requirement Coverage

A verification technique used to visualize and measure how the test cases are connected to the requirements is *Trace to Test Linkage* [19]. Visualizing the test cases in regard to which requirements these cover, increases the traceability and can also be used to observe problematic situations where a requirement is missing test cases [19]. Furthermore, if the requirements coverage is high, there is less risk that features not adhering to its requirements are missed. Each test case can be connected to one or more requirements. This visualization, as stated earlier, may verify that each requirement has test cases connected to it and as a result minimizing the risk of overlooking requirements that have not been tested. This is done to mitigate the risk that software is released without all functionality properly tested. An example of the *Trace to Test Linkage* is shown in Figure 2.1. From this picture we can see the potential problem occurring for *Requirement 6* which has no associated test case(s).

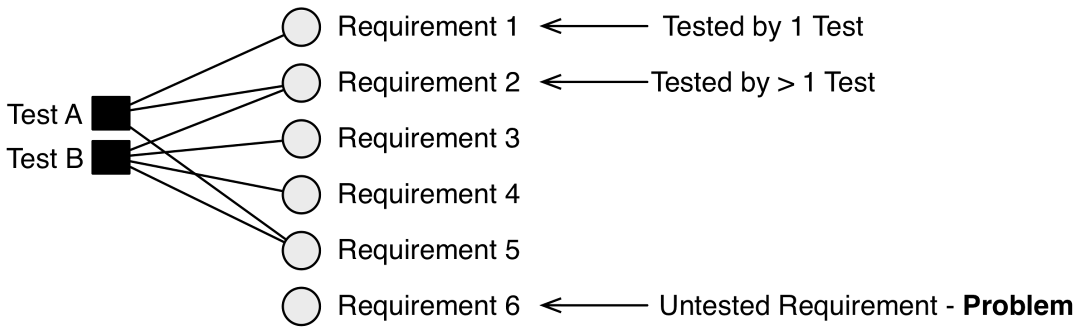


Figure 2.1: Trace to Test Linkage [19]

2.3 Gamification

Even if Gamification as a concept has been around for a long time, the term Gamification was not coined until 2002 by the British computer programmer Nick Pelling [20]. The term did however not gain momentum until around 2010 [21], which consequently leads to a limited amount of research within the field. Deterding et al. [7] came 2011 up with one of the more common definitions of Gamification: *"Gamification is the use of game design elements in non-game contexts"*. By their definition the focus is put on the use of game elements, rather than complete games, in areas not directly associated to games. These elements are by the authors stated to be seen as elements which possesses a key role in games, elements found in games or associated with games. Examples of frequently used characteristics of games are reward systems (achievements or points), leaderboards, immediate feedback (as a response of user actions) and a competition factor (either between the user and the game itself or between multiple users). By incorporating a set of game elements into a context which are not directly associated with games, it can be stated according to the definition by Deterding et. al. that Gamification has been introduced into the selected context.

The definition of Gamification opens doors for delimitations, for example in the given area of application and the desired goal for applying it. This can be seen in [22] where Domínguez et al. use another definition for their application of Gamification within an e-learning platform. Their definition is narrowed down to emphasize the area of use; within software applications. The definition they use includes the area of application as well as the goal for introducing Gamification, and is defined as "Incorporating game elements into non-gaming software applications to increase user experience and engagement".

Another example of an alteration to the original definition is from Huotari et al. [23] where the definition is defined from the service marketing's perspective, as "... a process of enhancing a service with affordances for gameful experiences in order to support user's overall value creation". The aim of their definition is to highlight the goal of Gamification instead of the methods used to reach these goals. The definition also gives the context for the application of Gamification, marketing, which the authors see as the main area of Gamification. This definition can be seen as a step to further emphasizing the value that Gamification might bring upon application, in contrast to the original definition, which focuses on the application instead of the actual result of it. The definition does not consider any non-gaming contexts as the original definition or the refinement by Domínguez's et al. This introduce possibilities for what the authors are calling "meta games" where the main service, e.g. a game, can be further gamified.

Hertel et al. [24] are looking closer into what motivates users to contribute to open source projects, where one of the suggested motivational classes is the social comparison. One example of the social comparison is the competitive factor, which is identified as taking place either between the developers in a team working on the same project, or between different teams working on different projects. Moreno [25] identifies the scoring mechanisms and leaderboard as factors that can encourage students in programming courses to improve, but also review their work.

Another aspect of Gamification is that it can function as a way of steering user behavior towards a desired direction [26]. An example how Gamification can steer user behavior is researched by Anderson et al. [27], who studied how badges are used on Stack Overflow[1], and how the users' behavior change before and after receiving certain badges. Their results include observations of an increase in the users' activity before receiving the badge which is in-lined with the assumption that external rewards motivate people to increase their activity to receive the award.

Providing the player with rewards for desired behavior, and a type of punishment for undesired behavior because of the player's actions, can be seen as a key aspect of games. In Hamari et al. [28] a literature review was conducted where the authors are identifying the most common motivational artifacts in the research literature of Gamification as Point systems or Achievements/Badges. A normal approach after implementing such reward system is to summarize the results in a leaderboard and present this to the players where they can see their position in comparison with other players [29]. The

[1]http://stackoverflow.com/

introduction of this leaderboard is a way of increasing the players motivation by allowing them to compete, not only against themselves, but also against each other, as suggested to increase motivation in [24].

2.3.1 Software Engineering and Gamification

Passos et al. [30] developed a framework for applying game mechanisms into software development. The framework developed by Passos et al. is based around the notion of a hierarchical challenge framework, which is a graph where each node is a challenge or level, and each edge is a transition between two challenges (levels). Each challenge (level) can in turn also contain sub-challenges (levels). The Hierarchical Challenge Framework is then applied to an agile software development project where the nodes in the graphs correspond to releases of the software, and each node contains sub-challenges made up of development iterations. Each challenge (level) in the game is assigned a binary number upon completion, which acts as basis for points, achievements, and rewards. The Hierarchical Challenge Framework is illustrated in Figure 2.2.

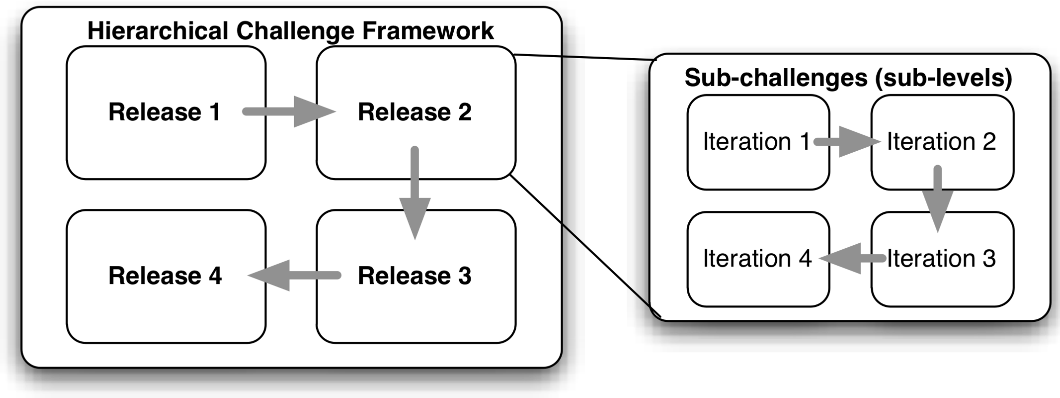


Figure 2.2: Hierarchical Challenge Framework [30]

Passos et al. [30] tested their framework on four software development projects involving 13 developers. The study had achievements that gave each developer medals or levels of experience depending on the level reached within each achievement. The achievements implemented by Passos et al. were:

- The number of tasks completed by an individual developer within a time frame.
- The aggregated number of tasks completed by an individual developer during her time at the company.
- The test coverage achieved by the team.
- The number of iterations the team completed within a time frame.

	LOC	Rules	Test. coverage	Dupl. code	Javadoc	Mark
No competition	3678	90.0%	38.8%	4,8%	79,5%	25.13
Competition	3247	90.6%	56.9%	4.3%	88.9%	25.53
Change %	-11.7%	0.6%	18.1%	-0.5%	9.4%	1.6%

Table 2.1: The improvement in code quality when competition was introduced.

Passos et al.[30] concluded on the basis of the case study that Gamification applied in software development could help in improving results by fostering competition among the developers, and that rewards, such as medals, can be used by managers to measure performance of team members.

Dubois and Tamburrelli [31] did further research of gamified software development by specifically study competition as a game mechanism in software engineering. The study investigated how competition between undergraduate students, enrolled in a computer science course, could lead to increased performance on programming assignments in Java. The researchers evaluated the performance regarding the number lines of code (LOC), the use of good programming practices (rules), testing coverage, code duplication, Javadoc, and the final mark on the assignment. The study involved 64 subgroups with two to three students each, which were divided into two groups, group A and group B. The students were given quality reports of their code from the code quality tool Sonar[2], with the difference that group A only could see their own reports, while group B also got to see the reports of their classmates. The research showed positive results, especially regarding the number lines of code, testing coverage, and the number of Javadoc comments. The number lines of code were 11.7% lower, the testing coverage was 18.1% higher, and the number of Javadoc comments were 9.4% higher in group B, which experienced competition compared with group A. The results are summarized in Table 2.1.

Sheth et al. [32] did work in Gamification of software development by their game HALO (Highly Addictive, socially Optimized software engineering). The game mechanics of HALO are based on psychological research in FLOW theory [33] and operand conditioning [34]. HALO, which was the main result from their research, takes inspiration from Massively Multiplayer Online Role-Playing Games (MMORPGs) by having the players conducting so called *quests*. *Quests* are missions and challenges that players try to complete, which in the context of software development, can range from small challenges such as fixing a bug to larger ones such as developing a new feature. If a player needs assistance in completing a challenge she can invite team members into a "party" where players collaborate on a challenge. The players will, by completing challenges, earn rewards in the form of achievements, currencies, and experience points. Larger challenges reward the players with more experience, and the longer a developer has been in the company; the harder it is to get more experience, which makes the game a challenge for all players independent of previous experience.

[2]http://www.sonarqube.org/

Furthermore, Sheth et al. researched the applicability of HALO in software engineering by using the framework to design a game for teaching testing to students [35]. The authors are proposing a Plugin for the Eclipse[3] IDE to "disguise" the testing from undergraduate students. The approach aims at disguising the testing so that the students do not directly see that they are actually testing the software. The idea is that the testing is hidden in certain *quests* which the students can complete. These *quests* are parts of assignments and by completing these, the students gain experience points and can also achieve a variety of achievements. In [36] the same authors are showing results from the application of their plugin in the class *Object Oriented Programming and Design with Java* for undergraduate students at Columbia University; the use of HALO in this course was optional for the students. The plugin visualized the progress of the students and provided a leaderboard to see how the students were positioned in comparison with other students given the experience points. This introduced the students to the competitive aspect of Gamification. The use of HALO in the course resulted in mixed results, given the feedback from the students. The main positive aspects were that it tracked the students progress and that it assured they were on track in the course. The negative aspects included too trivial tasks and that the only reason and benefits for using the tool was the extra credits given to each students. The last part of the negative feedback can be connected to the following statement by the authors: *"While the students think that they are competing just for points and achievements, the primary benefit of such a system is that the students' code gets tested much better than it normally would have".* Instead of focusing on increasing the students' intrinsic motivation for the testing, all focus is put on the external rewards from the disguised testing. Focusing on the external rewards could introduce problems when the reward system is removed. This is not discussed in the paper, but can be assumed to happen in future courses the students are participating in. For these courses, a decrease in the motivation for testing could possibly be observed if the students are not understanding the underlying reason for conducting testing, other than receiving extra credits.

2.3.2 Gamification and Psychology

A concept that has resulted from Gamification is the term Meaningful Gamification, which is defined as "the integration of user-centered game design elements into non-game contexts" [37]. The definition emphasizes, in comparison with the original definition, the centering of the user and how to incorporate game elements to have positive effect from the user's perspective and mindset. The focus is put into the underlying activity and how to increase the user's motivation to carry out the activity, instead of focusing on the external rewards and decrease the focus for the underlying activity. This can be directly connected to the previous discussion about Sheth's et al. [35] disguise of the testing process, where the focus was only on external rewards. This concept of Meaningful Gamification is connected to intrinsic motivation and internal motivation, where focus is put on the user's underlying motivation for the activity, and not the motivation for

[3]http://www.eclipse.org/

external rewards. By using a more user-centered design, Nicholson [37] argues that the designer can avoid the pitfall with Meaningless Gamification. Meaningless Gamification occurs when external rewards are used to control user behavior and to steer this behavior towards short terms benefit on an organizational level. Designing for Meaningless Gamification has the opposite focus from Meaningful Gamification, instead of a user-centered design focus is put into designing a more organization-centered perspective. Consequently, the focus is not to increase the user's understanding of the underlying activity, but instead increase an observable result of an activity on an organizational level.

In [38] Bitzer et al. are studying intrinsic motivation in open source software development, and how it attracts highly skilled and young developers worldwide to "public good". The authors are raising the awareness about the fact that in other areas this often resulted in low-quality results while in the open source community it is quite the opposite. The authors have identified some of the main reasons for developers participation in open source projects as personal need of software, the fun of play, and gift culture. The fun of play is one aspect that can be connected to Meaningful Gamification, where you emphasize the aspect of a game without scoring, which can be seen as play. Nicholson [37] is proposing his own term for the fun of play as: Playification, "the use of play elements in non-play context". The idea is that the player can create her own game with her own rules, and play for the pure pleasure without external rewards. One example, which is described by Nicholson [37], is a display inside hybrid electric cars from Toyota, which provides the driver with information whether the car is currently powered by the battery or the fuel, it also shows how the driving is affecting conduction of power back to the battery, which can be seen as examples of eco-friendly driving. By providing this information, the driver has the opportunity to set personal objectives for her driving, and can see the impact of her driving style. If there would just be an indicator to indicate to what extent the driver is driving eco-friendly for example, Nicholson[37] states that the driving experience would not be as meaningful as it is in the current setting.

External rewards, such as discussed in the previous section, need to be combined with internal rewards to avoid the severe problem resulting from the user only being motivated by these external factors and lose her internal motivation. Focusing on external rewards may implicitly lead to problems if these rewards are removed from the context. If there are no internal motivational factors, and the external motivational factors are removed, the user is more likely to lack motivation for the activity. Lepper et al. shows the existence of this shift in motivation for the activity in [39]; a phenomenon known as Overjustification.

Although the research in Gamification is limited compared to other more mature fields, it has been conducted an increasing number of studies the past few years. As discussed in section 2.1 there are some major socio-technical challenges within software testing and requirements coverage associated with skills, learning, motivation and commitment of developers. Recent research has studied and showed positive results for the potential influence of Gamification in a knowledge and learning setting (e.g. [40], [41], [42]). Other studies have also shown positive results for the influence of Gamification in motivating and engaging workers to increase productivity and improve quality of work (e.g [8], [9]), innovation/idea creation (e.g. [10], [11]), and intra-organizational systems (e.g. [43], [44], [45]). The positive results of Gamification within learning, work and collaborative contexts indicate that Gamification potentially could be applied to deal with socio-technical challenges within software testing.

2.3.3 Framework for applying Gamification

When introducing Gamification to an activity there are suggestions for the methodology of applying it, one of these frameworks has been constructed by Aparicio et al. [46]:

1. Identify the tasks to be gamified.
2. Identify several interests of the players; which later on will be used to develop the game mechanics in the game.
3. Select and develop the game mechanics of the game. These should match the three pillars of motivation described in [47]:
    - **Autonomy**
    - **Competence**
    - **Relatedness**
4. Analyze the performance of the gamified task.

When selecting the suitable game mechanics for the activity, this framework emphasis three main areas of motivation which is described in [47] as *Autonomy*, *Competence* and *Relatedness*. These three areas are from the *Self-determination theory* [48] which is a theory about the motivation of people and how this is affected by peoples' drive to grow, as well as the motivational needs of humans [29]. It is stated in [46] that in order to maintain a high level of intrinsic motivation it is important to cohere to these social and psychological needs. This is connected to the previous discussion on intrinsic motivation and *Meaningful Gamification* to increase the users' internal motivation for the activity without the sole focus on the external motivational artifacts.

Autonomy is the theory about the user's feeling of will, and wish to perform an activity [48]. In [47] the authors are identifying that the feeling for personal value and interest for the activity increases the feeling of autonomy. One important aspect that is described by the authors in [47] is when external rewards are used, that these should be used as informational feedback for the users. This is described as an important alternative opposed to using these to steer user behavior. As discussed in previous sections, in order to increase the feeling of autonomy and free will. A loosely controlled environment which allows the user to make own choices as a result of not over-controlling instructions is another aspect that is used to increase the feeling of autonomy [47].

Competence is about the need to feel efficient and competent [46], an example is from [49] which describes that students feel competence when they are able to meet the challenges of their education. Feedback as a result of user-action, new skills or newly obtained knowledge by challenges are examples that increase the level of competence for the user [46]. An important aspect of the challenges for the user is to keep this at a certain level throughout the activity. The preferred state of mind called FLOW [33], which is a state of mind of full focus and motivation for the activity at hand, occurs when the user is neither over-challenged nor under-challenged, referred to as being optimally challenged by the task [50].

Relatedness is about feeling connection to other people and in the sense of games, in relation to other players [46], it is also defined as the need to interact with players in the game and to connect to personal goals [51]. A concept connected to relatedness is *Meaningful community* [51] which can be seen as a community consisting of users with equal interests, by ensuring that the users are a part of and feel connection to this *Meaningful community*, designers can introduce achievements, point systems, and badges with the insurance that these fill their purpose. Groh [51] argues that if you have no one to show your achievements, points or badges to, these will not fulfill their purpose and will not mean anything. On the contrary, if you are in a community with a common interest, a badge or achievement might be considered of high value given the context and the environment.

2.3.4 Game Elements and Game Mechanics

The following section will introduce the set of Game Elements and Game Mechanics that have been selected for the experiment and which was implemented in a plugin to Eclipse. The set of game elements have been selected and motivated by the *Autonomy*, *Competence*, and *Relatedness*, described in section 2.3.3.

Rewards

Rewards, point systems, or achievements, are the most frequently used game elements in Gamification, and it is often used to steer user behavior in a desired direction [52]. The achievements are closely connected to badges which is a type of achievement, these are also widely used as external rewards within Gamification [28].

Reward expectancy [53], indicates whether the user is aware that she will receive a reward for the task. It can either be the case that the user is aware of the reward, expected reward, or that the user is unaware of the reward, unexpected reward. An example where rewards and achievements are known to the user, expected rewards, is the badge system on Stack Overflow [4], which is a Q&A forum especially for programmers and developers. From the badge section, a user can see the available badges and what is required from her to receive a specific badge. Users of Stack Overflow have a common interest in programming and development, which creates a *Meaningful Community* as discussed in [51]. As proposed in the above section about relatedness, this creates an environment where the use of achievements and badges make sense. Andersson et al.

[4]http://stackoverflow.com/

[27] investigated the badge system on Stack Overflow and conclude in their research that a badge system can increase the user activity and strengthen the incentive effect on the domain's users.

Unexpected rewards were introduced in Domínguez's et al. [22] as part of their set of achievements for their experiment, when investigating the effects of Gamification in education. These achievements were rewarded to the user upon reaching a milestone in the program, which was not described as an available milestone. The hidden achievements were added to the game elements as an attempt to increase the users' will to further explore the system to find more of these hidden rewards. There is no discussion how the hidden achievements effected the students, but since the designers introduced a leaderboard based on the achievements, it is assumed that once a hidden achievement was found by a student, other students would be aware of the existence of these achievements. In an educational setting where the desire is a general increase of the learning outcomes for all students, it can be preferable to handle hidden achievements in this sense, while in a more competitive setting these achievements could be kept a secret to other users who have not accomplished the desired milestone. This would increase the exploratory user's advantage in the activity, while also not affecting the other users' conditions of finding these. If a middle ground between the two approaches to visualize hidden badges to other users when one user is receiving it, there is the option to inform the other users that a hidden badge has been achieved. While simultaneously leaving the necessary milestone a secret.

Feedback

In [54], Muntean argues that because of the positive feedback students get from Gamification they become more interested in the activity, following an increased desire to learn. The underlying reason for the increase and willingness to learn is stated by Muntean as the push forward that the feedback brings. Positive feedback can also be related to the competence, which was discussed in previous sections and in [46]. In [53] Cameron et al. shows evidence for the positive effects from verbal positive feedback and verbal praise on the intrinsic motivation and the interest of the user. Cameron et al. further state that the intrinsic motivation for the task is kept even when the positive feedback is removed.

Another aspect of the general feedback from the system to the user is about informational feedback, which was stated as one of the important aspects to enhance Autonomy in [47]. Ryan argues that informational feedback should be used in regard to external rewards to decrease the risk that external rewards are used to steer user behavior, which in turn can decrease the intrinsic motivation. In this sense, the feedback from the system should increase the intrinsic motivation of the user instead of trying to steer it in a certain direction.

Feedback is however often used in the sense of external rewards because of a desired behavior of the user [42] [22] [55], this is a common aspect of Gamification implementations [28]. Steering user behavior for a productive task could be seen as beneficial if the user realizes, through deeper knowledge and because of increased commitment in the task, the benefits of conducting the activity.

Dorling et al. identify that the feedback is also used as an indication to the user of her progress [56]. The authors further state that it is important to measure the progress at multiple levels in the game. This can be directly connected to the competence aspect, the progress will indicate to what extent the user gains deeper knowledge and competence in the activity.

Competition

Competition has shown a mix of results within games and gamification. Dubois and Tamburrelli show positive results of competition within software development regarding some variables such as lines of code, testing coverage, and number of Javadocs written [31]. Tauer and Harackiewicz [57] showed that achievement orientation moderates the effect of competition, and divides users of games into two types of people: people that has High in Achievement Motivation (HAM) and people that has Low in Achievement Motivation (LAM). HAMs are defined as people who like challenges, have a desire to attain higher level of competence and who have a wish to outdo others [57]. Furthermore, Tauer and Harackiewicz [57] showed that HAMs both performed better and enjoyed the game more than LAMs when competition was implemented in a game. Moreover, competence valuation and perceived challenge affected the intrinsic motivation in playing the game.

2.4 Game-based Adaptations

Gamification can be seen as a middle ground between the traditional non-gaming approach and Game-based adaptation of a task or desired learning outcome. While Gamification aims towards enhancing user involvement and increase the motivation of users by introducing game design elements in non-game contexts, game-based adaptations seek the same outcomes while incorporating the task into a complete game. An example of the difference can be seen in Jayasinghe and Dharmaratne [58]. In their work the authors researched the difference in learning outcome between a fully implemented game, game-based learning, and a tool consisting of a set of game-mechanics, Gamification. The task was about learning and understanding sorting algorithms and after each “session” a quiz was conducted to see to what extent the students had understood the underlying theory of the specific algorithm. Results showed that the students who used the Gamification tool scored higher on the quizzes than the students who had been using the game-based tool.

Game-based adaptations have been introduced into education and other areas where the desired outcome has been enhanced learning, Game-based learning is used as the term for this case. Game-based learning has been applied in education [58] among students to increase the motivation and learning outcomes in education. In [59] the authors are describing an experiment where an educational game was used to teach civil engineering to university students. In their experiment the authors showed results that confirmed their hypothesis that the learning outcome when using the game was at least equivalent to the students who learned without the game, and with an increase in the motivation.

In [60], games were created to increase the students' knowledge about the campus library and its services, it also functioned as an introduction for new students in gaining knowledge about the library's provided services. This is an example of the variety of possible application areas for Game-based adaptations.

Game-based adaptations can also be seen in other areas beyond education, an example is shown by Flatla et al. [61] where calibration games were created to increase the motivation and engagement in gathering of calibration data for interactive systems. Their results indicate an increase in the motivation for the calibration task, as a result of a more entertaining procedure.

2.5 This thesis and previous research

As presented in the previous sections, it has been conducted some research of Gamification within software engineering, e.g. [30][31][32][35][36]. The previous research that most resembles the research of this thesis is the work by Sheth et. al. [35], which tried to teach testing to undergraduate students through a game in Eclipse. In their research there was a focus on disguising the actual testing from the subjects, as a game, in order to make the students test more. However, the research of this thesis does not focus on creating a game, but rather implementing game elements in an existing testing process. Furthermore, the participants in this research already have knowledge in software testing, so the focus is on improving their testing instead of teaching them how to do testing. The work by Sheth at. al. [35] furthermore research how students perceive a game that teaches them testing, while the topic of this thesis, is the focus on how Gamification can improve the effectiveness of software testing, as well as motivate testers to improve their testing. The rest of the work about Gamification within software engineering is not related to software testing, but is mostly focused on transforming software development to a game, and thus motivate developers to write better code, as well as making their work more enjoyable.

This thesis is, through the methodology explained in the upcoming section, introducing the concept of Gamification into the field of software testing. In the previous sections there have been evidence showing the positive effect of implementing Gamification into other contexts, including software engineering. Our work is focused on investigating if these positive effects are present when Gamification is applied to software testing as well.

3 Method

The research of introducing Gamification in software testing involved comparing two groups of subjects, one using an Eclipse plugin which had Gamification elements applied to it and one plugin which did not have any Gamification elements applied to it. The subjects' result was gathered during and after the experiment session, and then further analyzed. When the methodology was established the frameworks from Section 2.3.3 were considered as well as the set of Gamification elements from Section 2.3.4.

3.1 Scope

The scope of the experiment was produced by following the method for conducting experiments in software engineering, proposed by Wohlin et al. [62]. The experiment scope served as a general structure of the experiment conducted.

Objects of study The objects of study were two processes of performing unit testing; one with Gamification introduced, and one without Gamification.

Purpose The purpose of the study was to compare a unit testing process incorporating Gamification, against the same unit testing process without Gamification. The goal was to research the impact Gamification might have on software testing regarding the number of defects found and percentage of requirements covered by the test cases. A secondary purpose of the study was also to measure the subjects' motivation and perception when using the testing process.

Quality Focus The experiment will focus on the effectiveness, regarding the number of defects detected and the percentage of requirements covered by at least one test, of unit testing with and without Gamification applied.

Perspective The results of the experiment are considered from the tester's and the researchers' points of view. The perspective of companies, managers and employers, who in the future may introduce Gamification in software testing, are other perspectives that are discussed in the study.

Context The experiment was conducted off-line with practitioners from industry and students from academia. The experiment involved unit testing of the LinkedList class from Java OpenJDK, which had been seeded with a specific number of faults.

3.1.1 Hypotheses

The research aimed to investigate two research questions as described in Section 1.3. Each research question was tested after the experiment by hypothesis testing. The research questions are reproduced in the following list, together with their corresponding null and alternative hypotheses.

RQ1: Is there a difference in the number of defects detected by testers when performing unit testing with Gamification applied, than without Gamification applied, in a given time frame?

- **H_0:** There is no difference in the number of defects detected by testers when Gamification is applied to their testing process.
- **H_1:** There is a difference in the number of defects detected by testers when Gamification is applied to their testing process.

RQ2: Is there a difference in the percentage of requirements covered by tests when performing unit testing with Gamification applied, than without Gamification applied, in a given time frame?

- **H_0:** There is no difference in the percentage of requirements covered by tests when Gamification is applied to their testing process.
- **H_1:** There is a difference in the percentage of requirements covered by tests when Gamification is applied to their testing process.

3.1.2 Experiment variables

The experiment consisted of dependent and independent variables. The dependent variables were the ones that were ought to be measured, and thus underlies the hypothesis-testing. The independent variables in turn differed between the two groups, since the research was to investigate how they were affected by the dependent variables. The hypothesis testing was then used to conclude whether a change in the independent variables significantly lead to a change in the dependent variables. The variables that have been identified for the experiment are presented in table 3.1.

Dependent variables	**Independent variables**
Number of detected defects	{Gamification Elements}
Requirements coverage	

Table 3.1: Dependent and independent variables of the experiment

	Control-group	**Gamification-group**
Academia	7	10
Industry	3	4
Total	10	14

Table 3.2: The distribution of subjects from academia and industry.

3.1.3 Subjects

The subjects in the experiment originated from both academia (students at Chalmers Univeristy of Technology) and industry, but with a larger number of subjects from academia. In order to participate, subjects were required to have knowledge in Java and unit testing, which was checked through a questionnaire when subjects signed up for the experiment. From respondents of this questionnaire, an invitation was sent out to subjects meeting the stated prerequisites to participate in the experiment; these were asked to sign up on one of the available experiment dates. The first questionnaire resulted in 33 respondents.

The distribution of people from academia and industry for both groups can be seen in table 3.2. The dividing of subjects was conducted through random sampling to even out the difference in knowledge of testing and programming between the groups, and thus reduce the threat to validity that difference in skills that subjects might have.

Power analysis

A power analysis was performed before the invitation to subjects was distributed; to better understand the preferred number of subjects for the experiment. The statistical power, the probability of correctly rejecting H_0, was chosen to 0.95. Cohen's d was assumed to be 0.8; a large practical significance in the difference between the mean of the control group and the Gamification group[63].

With Cohen's d = 0.8 and statistical power $\beta = 0.95$, a sample size of 42 subjects for each group was calculated. The desired total number of subjects for the experiment would then be $2 \times 42 = 84$.

Although the aim was to have 84 subjects in the experiment; the final number of subjects in the experiment was 24 subjects, the sample is further discussed in Chapter 6.

3.2 Instrumentation

The System under test, SUT, used for the experiment was a standard Java class, LinkedList. This standard class had been seeded with twenty-three faults; a number of seeded faults that was unknown to the subjects. The class with the seeded faults was named MyLinkedList. The seeded faults were introduced into the class by applying the following algorithm.

1. Calculate a random number to determine the line number where the fault should be introduced.

   If the line does not contain code where a fault can be introduce, choose the nearest line below the randomized line where a fault can be introduced.

2. If a fault can be introduced in the selected line, modify the code according to one of the mutation operators in Table 3.3

   If several mutation operators can be used (e.g. change '+' to '*', '/' or '-'), then calculate a random number to choose the mutation operator.

Examples of where a fault could not be introduced in the code, as described in the algorithm, were in a private method which is not visible to the subjects, a core method that the tester is likely to use repeatedly, or a method in which introducing a fault makes the class unusable. The core methods included: boolean add(E e), E get(int index), int size() and E remove().

The reasons for choosing the LinkedList class as the SUT were based upon the fact that it is a commonly used java class and subjects from both academia and industry are most likely to have knowledge in this class as well as experience in using it. The subjects had no knowledge of which software under test that was chosen before the introduction to the experiment.

Each subject was given a set of four artifacts in addition to the SUT, an introduction to the plugin (user manual), a requirement specification for the MyLinkedList class, access to the JUnit API through the plugin, and a workspace with a predefined project included. The user manual for the plugin was different depending on the group each subject was enrolled in, explaining the plugin's functionality and layout. The manual functioned as the subjects' main source of information about the plugin with the introduction by the experiment supervisor as a summary of the stated information. The requirement specification used was combined using the available Javadoc for each of the methods included in the LinkedList. The Javadoc was split up into functional requirements and resulted in a requirement specification containing 81 requirements.

The experiment artifacts were not changed between experiment iterations, to ensure that all subjects were given the same conditions to carry out the experiment. The predefined workspace and the containing project were distributed and imported in the Eclipse instance before the experiment started and included short JUnit examples and a recap of the task to be completed.

Description	Example Mutation
Absolute Value Insertion	$x = 2 * a; \rightarrow x = 2 * abs(a);$
Arithmetic Operator Replacement	$x = a + b \rightarrow x = a * b;$
Logical Connector Replacement	$x = a\&\&b; \rightarrow x = a\|\|b;$
Relational Operator Replacement	$if(a > b); \rightarrow if(a < b);$
Unary Operator Insertion	$x = 2 * a; \rightarrow x = 2 * -a;$
Unary Operator Deletion	$if(a < -b); \rightarrow if(a < b)$
Shift Operator Replacement	$x = a >> b; \rightarrow x = a << b;$
Logical Operator Replacement	$x = a\&b; \rightarrow x = a\|b$
Conditional Operator Replacement	$if(a\&\&b); \rightarrow if(a\&b);$
Assignment Operator Replacement	$x+ = 2; \rightarrow x- = 2;$
Reference Assignment and Content Assignment Replacement	$List\ l1, l2;\ l1 = newList();\ l1 = l2; \rightarrow l1 = l2.clone();$
Reference Comparison and Content Comparison Replacement	$Integer\ b = new\ Integer(1); boolean\ x = (a == b); \rightarrow boolean\ x = (a.equals(b));$
this keyword deletion	$this.r = r; \rightarrow r = r;$
this keyword insertion	$this.r = r; \rightarrow this.r = this.r;$
Accessor Method Change	$circle.getX(); \rightarrow circle.getY();$
Modifier Method Change	$circle.setX(1); \rightarrow circle.setY(1);$

Table 3.3: Mutation operators [64]

3.2.1 Eclipse Plugin

The developed Eclipse plugin's purpose was to function as a data gathering tool for the experiment as well as visualizing the collected data and provide feedback to the tester. The plugin came in two different versions, one for the Gamification group and one for the Control group, the difference between these two plugins was the Gamification aspects which only were available in the Gamification group's plugin. This set of Gamification aspects are introduced and motivated in Section 2.3.4, and a visual representation of the plugin and further explanation of its Gamification aspects are explain in Appendix A, Section A.2.

The two plugins gather the same data and used the same database to store the data for the analysis. By providing the same architecture, backend and way of parsing the data for the plugins the two groups were conducting the experiment under the same conditions. The conditions which needed to be as equal as possible between the groups, and not favor one of the groups in any way, are the execution time for the running, parsing and uploading of data from the subject's test suite as well as the way that data were parsed.

The amount of feedback provided to the tester varied between the plugins, but the data which the feedback relied on was the same for the two plugins. In Appendix A follows a more in-depth review of the plugins' functionality, and the feedback that was provided to the subjects of each group. Appendix A also provides a summary for all Gamification elements which were implemented in the plugin from the selection and discussion in 2.3.4.

The plugins distinguish between an actual found defect and a regular failure of a test case in the MyLinkedList class by using the original LinkedList class and run the subject's test suite against this class after the run against MyLinkedList. A regular failure can occur when the tester for example makes the wrong assumption or a mistake in the test case which causes it to fail. This is conducted to be able to count the actual defect count on the fly. Since the original LinkedList class is considered to be defect free, a defect is identified as a found defect in MyLinkedList class if a test case fails when running against the MyLinkedList but passes when running against the original LinkedList class. This results in the need to run the test suite twice which creates a longer execution time for the plugin. This is a necessary delay which is the same for the Control group's plugin, and hence the same condition applies to the both groups in execution time.

3.2.2 Intrinsic Motivation Inventory Questionnaire

The subjects were asked to complete a survey online after the experiment, which measured the subjects' experience and perception of the experiment, and their experience in using the Gamification elements. The survey was based on the Intrinsic Motivation Inventory, which is a measurement device intended for measurement of subjects' perception and experience of experiments, with focus on intrinsic motivation.

The measurement device was chosen because it measures what the survey in this study aimed to measure; experience and intrinsic motivation in doing the testing task and using the Gamification elements that had been implemented. The Intrinsic Motivation Inventory has also shown to be a stable measurement device, which in past studies has yielded an adequate construct validity and internal consistency[65].

The Intrinsic Motivation Inventory used in the study consisted of several statements, which the respondents were asked to answer, on a 7-point scale, how true each statement was for them. The 7-point scale ranged from one; ”Not true at all”, to seven; ”Very true”. The statements in the survey were chosen to measure three dimensions of the subject's perception of the task in the experiment: Interest/Enjoyment, Perceived Competence, and Pressure/Tension. The interest/enjoyment measured how enjoyable and interesting the subjects perceived the task to be, the perceived competence measured how well the subjects think they performed compared to the other subjects, and the pressure/tension measured how stressful the subjects perceived the task to be. Each statement was asked at least twice, but with a different wording and the scale negated, which aimed to give more accurate answers. The survey was divided into three parts; general information about the respondent, statements about the task in general, and statements about the

Gamification elements. The control group did only get to answer the general questions and the statements about the task, since they did not have any Gamification elements in their experiment session. Some statements for the Gamification elements were added to measure additional dimensions of the subject's perception of each element, such as usefulness and purpose. The questions/statements asked to the Control group and the Gamification group are reproduced in appendix B.

3.3 Operation

The subjects were given the artifacts for the experiment shortly before the introduction, with time to read it before the experiment started. The subjects had no knowledge about the group they had been assigned to.

Each experiment session consisted of 15 minutes introduction, one hour of testing, and 15 minutes of answering the survey. The groups conducted the experiment in separate rooms, to decrease the risk of subjects from separate groups influencing each other. The experiment environment was the same for both groups: a Linux based OS and the same Eclipse version.

When questions arose during the experiment from one of the subjects the question was repeated to all subjects and answered to minimize the risk of giving different answers to different subjects, the question was also noted by the experiment supervisor if the question would arise at a different iteration. This was done to ensure that subjects were given the same information. If the question concerned only one of the groups it was kept from the other group, but if it was a general question it was given to the person supervising the other group to inform the subjects about the question and answer.

The experiment was conducted in five iterations with a different number of subjects in each run. The Gamification group was run with at least four subjects to ensure the leaderboard did not loose its purpose. The Control group had no requirements of the number of subjects in each experiment session, and did therefor consist of two to four subjects at a time. One iteration with a Control group was conducted with one subject in the regular computer hall and two subjects doing the experiment from home. During this iteration, the distributed subjects were in direct contact with the experiment supervisor and the same setup when it came to distribution of instrumentation and way of handling questions were ensured by the experiment supervisor. The two subjects had a similar setup at home as in the computer hall; a Linux based OS, and the same version of Eclipse.

The subjects were not in any way communicating with each other during the testing session, and the subjects were at the same time seated out of range for viewing another subject's monitor. This was done to minimize the threat that subjects either cheated by looking at another subject or communicated directly or indirectly with one and another.

The subjects were after the study asked to fill out a questionnaire, based on the Intrinsic Motivation Inventory (IMI); a multidimensional measurement device which intended to measure the subjects subjective experience with the task they performed.

All data generated during the experiment were collected after the experiment for both groups. This data included the source files, log files, test reports, and coverage reports. The data were after the experiment reviewed and prepared for the analysis as described in Section 3.4.

3.4 Gathering of Data and Reviews

In the following sections are the methodologies described, for how the collected data was handled and reviewed to ensure that the calculations were made on the correct underlying data.

3.4.1 Defect Detection Review

Before the analysis of the number of defects detected by each subject could be done, a review of the test cases was conducted. The reason for conducting the first step was to locate potential duplicates regarding two test cases finding the same defects. In theory, a subject could write two test cases which found the exact same defect and still be rewarded with more points for finding more defects. This means that the subject by writing two test cases that locates the same defect, and hence is rewarded with points for two defects, actually only has found one defect. The review was necessary to retain the correct number of defects detected for each subject.

When reviewing the source files for these potential duplicates, total number of found defects were simultaneously updated to hold the correct number to still be valid for analysis. The defect count was updated in one of the following ways:

- If a specific defect had been found and counted by two different test cases, the number of defects was decreased by one. Afterwards, the source file was reviewed for any other duplicates, and the this step was repetead if any was found. The actual number of found defects was updated to hold the correct value after the review.
- If there were no test cases which located the same defect in the source file, the total number of defects found were left unchanged.

3.4.2 Requirement Coverage Review

Before the connecting of tests to requirements could start, each test case had to be accepted as correct in the sense that it actually tested the intended behavior with the right assumption. If tested with the wrong assumption it would lead to a failure when running the test suite against the original LinkedList class which had no seeded faults. If a failure occurred in any of the test cases when running the test suite against the original LinkedList, it was checked to verify the existence of a fault from the tester. When a fault in a test case had been found, which resulted in a failure or error when executing the test case, this was temporarily removed from the test suite for the Requirement coverage analysis. Test cases with the wrong assumptions or other failures/errors made by the tester, that caused either an error or a failure in the original LinkedList class, are test cases consisting of any of the following characteristics:

- Fails due to wrong assumption from the tester:

  -
    ```
    public void testExample() {
        list.add(1); list.add(2); list.add(3);
        assertEquals(list.getFirst(), 3);
    }
    ```

    wrong assumption of what to expect of getFirst().

  -
    ```
    public void testExample() {
        MyLinkedList<Integer> list = new MyLinkedList<Integer>();
        exception.expect(IndexOutOfBoundsException.class);
        list.getFirst();
    }
    ```

    wrong assumption of what exception getFirst() throws in case of an empty list.

- Error due to uncaught/unexpected exception:

  -
    ```
    public void testExample() {
        list.add(1); list.add(2); list.add(3); list.add(1);
        assertEquals(list.remove(4), false);
    }
    ```

    The tester tries to remove the fourth index but it is out of bounds for the list. This results in an IndexOutOfBoundsException thrown, but the tester has not expected this or caught it.

Mistakes from the testers that do not cause an unintended failure or error are left in the source file and connected to the actual tested requirement, even if the tester intended to test another requirement and this test case is a duplicate of an existing test case which has already been connected to a requirement. Examples of these are:

- Duplicate test cases, unintended or intended:

  ```
  public void testGetLast() {
      list.add(1); list.add(2); list.add(3);
      assertTrue(list.getFirst() == 1);
  }
  ```

  It is considered a duplicate if the tester has tested the getFirst() function in testGetFirst() and copied the test case, but have forgotten to change the actual call to getLast() instead of getFirst() and what to expect. There is nothing wrong with the assumption or the test case itself.

In the requirement specification there are requirements in the following format: ”Intended behavior1”, intended behavior2”. This indicates that there are two criteria that needs to be fulfilled for the requirement to be considered tested, an example of such a requirement is the first requirement for the E getFirst() method:

- FR-1: ”Returns, but does not remove, the first element of the list.” Here it is stated that the element should be returned by the method call and still be in the correct position in the list after the call. For this requirement to be considered tested, both of these criteria need to be verified.

Some requirements are divided into several sub-requirements; an example where this format is used is for the method boolean contains(Object o):

- FR-13: ”Returns true if the list contains the specified element”
- FR-14: ”Returns false if the list does not contain the specified element”

These requirements could have been written as one of the following format: ”Returns true if the list contains the specified element, or returns false if the specified element is not present in the list”. For this experiment there was a mix of these formats, with a larger portion of the format where the requirements are split up to only hold one intended behavior.

For a given requirement to be considered tested it had to be covered by at least one test case, which tested all stated behaviors for that requirement. This could either be done in one test case, or be split up between two or more test cases. Below follows an example of a test case which does not test both behaviors of the FR-1 requirement and one that test both of the two intended behaviors:

-
  ```
  public void testGetFirst() {
    list.add(1); list.add(2); list.add(3);
    assertTrue(list.getFirst() == 1);
    }
  ```

  This test case verifies that the element returned by the getFirst() method is indeed the first element, but does not consider the second intended behavior that the list should be unchanged after the call.

-
  ```
    public void testGetFirst() {
        list.add(1); list.add(2); list.add(3);
        assertTrue(list.getFirst() == 1);
        assertTrue(list.get(0) == 1);
        assertTrue(list.size() == 3);
    }
  ```

  The test case checks both the intended behaviours by first verifying that the return of the call is the first element, and verifies after the call that the first element is still the same and that the size of the list is preserved.

From reviewing the data associated with the usage of the Trace to Test Linkage tool we see that there were eight subjects from the Gamification group who used this tool to connect at least one test to a requirement, this results in a 57% usage of this tool among the subjects in the Gamification group.

3.4.3 Test Coverage Review

When reviewing the test coverage data, all invalid test cases were removed from the source files, this was the same filtering of test cases that were done before the requirement analysis; the criterion can be seen in Section 3.4.2.

For the test coverage analysis it was desired to focus on the valid test cases to gain a more actual indication of the test coverage level between the groups, and disregarding the invalid ones ensured that the analysis would provide a more correct analysis between the groups. For the same reason as the requirement coverage analysis, invalid test cases were not considered to fill their purpose of testing a functionality of the class, and hence should not contribute to any increased number in testing coverage for the class.

3.4.4 Review of Invalid Test Cases

The criterion stated for a test case to be classified as invalid and hence included in the invalid test case count is similar to the set discussed in section 3.4.2. For the sake of the Requirement Coverage analysis and Test Coverage analysis, these test cases were removed, but this analysis focused on the invalid ones. If a test case contained two of the criteria, it was still only counted as one invalid test case. The review was conducted in the same manner as the previous analyses, by reviewing each individual test case and identifying the invalid ones. In addition to wrong assumptions/mistakes the case with unfinished test cases are also included in this count:

- Unfinished test case or unimplemented
  - If the test case consists only of a variation of assertTrue/assertFalse(true/ false); Intentionally or unintentionally the subject has not finished or started the test case.

4

Results and Analysis

The results of the study aimed to specifically underlie the analysis and discussion of the research questions. However, more variables were measured during the experiment, which allows for further analysis, and thus gives a better picture of the effects that Gamification might have in software testing. The results can be categorized into two categories; results from the testing session, and the results from the survey.

4.1 Distributions of Data

In order to identify the statistical methods to use and the assumptions to make about the underlying data, we produced histograms for each variable measured. The data variables measured in the experiment, and which we use in the calculations of upcoming sections, are plotted in Appendix C. The data points are generally spread out in the histograms, but with two or more modes for each variable. A Shapiro-Wilks test was performed for each of the variables in an attempt to conclude if the data was not normally distributed. The Shapiro-Wilks tests did however not show statistical significance for any of the variables under a 95% significance level. Nevertheless, the Shapiro-Wilks tests resulted in a p-value around 0.10 for each variable, which made us reach the conclusion to not assume the data to be normally distributed. The same histograms and Shapiro-Wilks tests were done for the variables from the survey, and they were also assumed to not be normally distributed. Furthermore, all observations was assumed to be independent from each other, and all data values was assumed to be ordinally scaled.

Since the data values were assumed to be independent from each other, ordinally scaled, but not normally distributed, a student t-test was not selected to calculate statistical significance. Instead Mann-Whitney-Wilcoxon tests were performed to calculate statistical significance for each of the two-sample variables, and Wilcoxon signed rank test was used to calculate significance for each of the one-sample variables.

4.2 Detection of Defects

	Control Group	Gamification Group
Mean ($\bar{x}$)	5.70	8.57
Standard deviation (σ)	1.95	2.31
p-value	0.017	

Table 4.1: Number of found defects for the two groups.

The main goal for the subjects in the study was to detect as many defects in the system under test as possible during the given time frame. The number of detected defects for each subject was counted after the reviews described in section 3.4.1.

The average number of detected defects for the Control group was $\bar{x} = 5.70$, with a standard deviation of $\sigma = 1.95$, while the average number of detected defects for the Gamification group was $\bar{x} = 8.57$ with a standard deviation of $\sigma = 2.31$. The numbers can be seen in Table 4.1. The different number of detected defects for both groups was statistically significant under a two-sided Mann-Whitney-Wilcoxon nonparametric test, which resulted in $p = 0.017$. The number of detected defects are further illustrated in Figure 4.1.

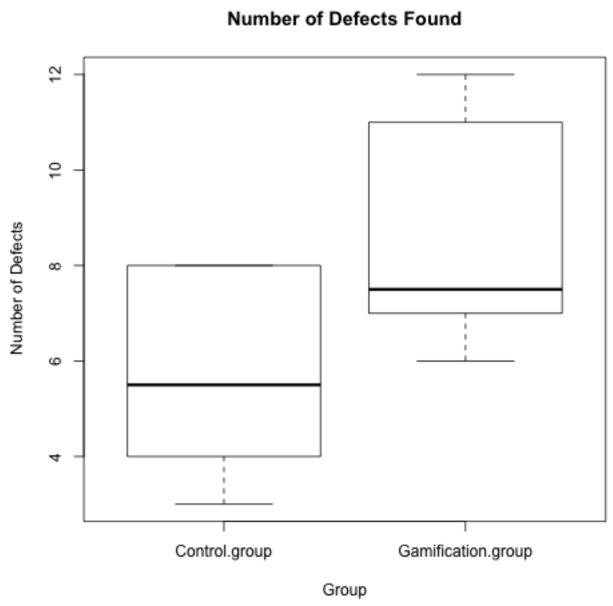


Figure 4.1: The number of detected defects for the subjects of both groups

4.3 Requirements Coverage

The requirement coverage was calculated after the reviews of the test suites described in Section 3.4.2. The results for the requirement coverage are summarized in Table 4.2.

The average percentage of requirements covered by at least one test for the Control group was $\bar{x} = 21.47\%$ with a standard deviation of $\sigma = 8.68\%$. The percentage of requirements covered by at least one test for the Gamification group was $\bar{x} = 36.22\%$ with a standard deviation of $\sigma = 13.34\%$. This resulted in an absolute difference of 14.75%, which under a two sided Mann-Whitney-Wilcoxon test gave a p-value of $p = 0.005$, which proves statistically significance. The results are further illustrated in Figure 4.2.

	Control Group	Gamification Group
Mean ($\bar{x}$)	21.47%	36.22%
Standard deviation (σ)	8.68	13.34
p-value	0.005	

Table 4.2: Requirements Coverage for the two groups

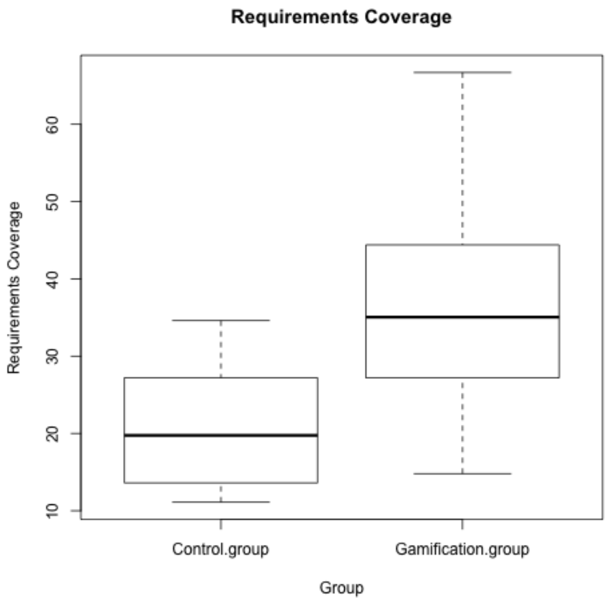


Figure 4.2: The requirements coverage for the subjects of both groups

4.4 Testing Coverage

When quality of test suites is evaluated it is common to analyze the testing coverage. As part of the research into whether the testing quality differed between the groups, an analysis of the testing coverage was performed to function as an aid in the upcoming discussion. The analysis was performed automatically during the experiment, but also after the experiment to exclude invalid test cases from the testing coverage as presented in Section 3.4.3. The types of test coverage measured were instruction coverage and branch coverage.

The instruction coverage for the Control group was in average $\bar{x} = 38.70\%$ with a standard deviation of $\sigma = 12.75\%$, while the Gamification group reached an average of $\bar{x} = 46.79\%$ with a standard deviation of $\sigma = 9.62\%$. The summary of the Instruction coverage for the two groups can be seen in Table 4.3. Furthermore, the mean branch coverage was $\bar{x} = 29.70\%$ for the Control group with a standard deviation of $\sigma = 10.63\%$, and the average branch coverage for the Gamification group was $\bar{x} = 36.71\%$ with a standard deviation of $\sigma = 8.15\%$, these values are summarized in Table 4.4. The results indicate that the Gamification group both reached a higher instruction coverage as well as branch coverage, which is also illustrated in figure 4.3 and 4.4. However, a two sided Mann-Whitney-Wilcoxon test shows $p = 0.113$ for instruction coverage and $p = 0.099$ for branch coverage, hence the results is not statistically significanct.

	Control Group	Gamification Group
Mean ($\bar{x}$)	38.70%	46.79%
Standard deviation (σ)	12.75%	9.62%
p-value	0.113	

Table 4.3: Instruction Coverage for the two groups

	Control Group	Gamification Group
Mean ($\bar{x}$)	29.70%	36.71%
Standard deviation (σ)	10.63%	8.15%
p-value	0.099	

Table 4.4: Branch Coverage for the two groups

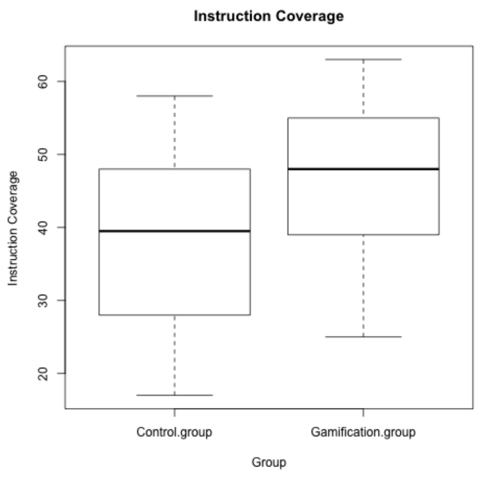


Figure 4.3: Boxplot for the instruction coverage for each group.

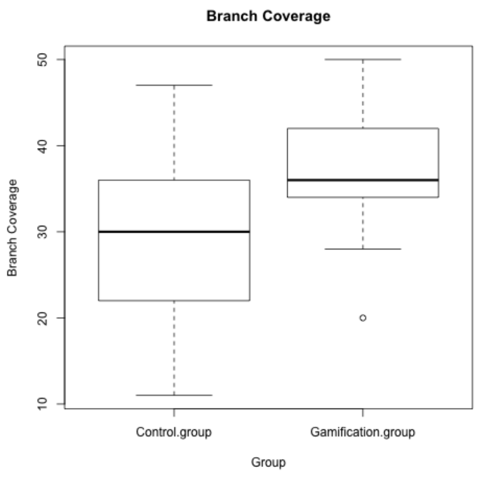


Figure 4.4: Boxplot for the branch coverage for each group.

4.5 Invalid Test Cases

One part of writing a good suite is to write correct test cases that are not invalid. Since the underlying idea behind the study is to research how testing can be improved through Gamification, it is important to assure that factors within Gamification do not lead to testing-code of lower quality and with more invalid test cases. The number of invalid test cases made by the subjects was measured during the experiment, and then reviewed according to Section 3.4.4. The result of the analysis shows that the Control group made an average number of errors of $\bar{x} = 2.80$, with a standard deviation of $\sigma = 2.25$. The Gamification group did an average number of errors of $\bar{x} = 3.86$, with a standard deviation of $\sigma = 4.09$. The result shows that the Gamification group on average made more errors than the Control group. A two sided Mann-Whitney-Wilcoxon test shows however $p = 0.595$, which make the result not statistically significant. The results are summarized in Table 4.5. The Gamification group did also contain an outlier with sixteen errors, which is shown in figure 4.5. The outlier is affecting both the mean and the standard deviation for the Gamification group, and the difference between the groups would be much smaller if it was to be removed.

	Control Group	Gamification Group
Mean ($\bar{x}$)	2.80	3.86
Standard deviation (σ)	2.25	4.09
p-value	0.595	

Table 4.5: Number of Invalid test cases for the two groups

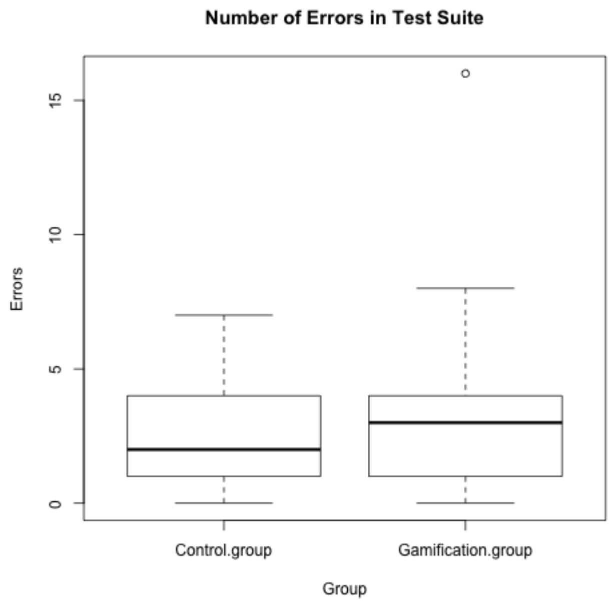


Figure 4.5: The number of invalid test cases for the subjects of both groups

4.6 Results From the Survey

Apart from researching the quality aspect of the test suites that the subjects produced, it has been of importance to investigate the perception of the task performed in the experiment, as well as the perception of the Gamification elements. The perception of the Gamification elements was gathered to get a picture of the properties of each element and how they affected the subjects. It was also measured how the general testing process was perceived by the subjects, and if Gamification in itself could make the task of testing more interesting and motivating. The survey setup and the measured variables is explained in section 3.2.2.

4.6.1 Perception of the Testing Session

The statements about the subjects' perception of the testing task were asked to both groups in the experiment, which allowed comparison between the groups. The variables that were measured were Interest/Enjoyment, Perceived Competence, and Pressure/ Tension; which are summarized in Table 4.6. The statements were based on a 7-points scale, where rate one is the lowest ranking and rate 7 is the highest.

The Control group gave their average interest and enjoyment of the testing task a rating of $\bar{x} = 5.3$, while the Gamification group gave it a rate of $\bar{x} = 6.18$, which under a Mann-Whitney-Wilcoxon two sided test is a significant difference, with a p-value of 0.018. Furthermore, the perceived competence for each group was on average rated as $\bar{x} = 3.93$ for the Control group and $\bar{x} = 4.24$ for the Gamification group, which however is not a statistically significant difference under a Mann-Whitney-Wilcoxon test ($p = 0.310$). Moreover, the last variable measured, Pressure/Tension shows an average rating of $\bar{x} = 2.73$ for the Control group and $\bar{x} = 3.90$ for the Gamification group, a result that is not significant under a Mann-Whitney-Wilxocon test ($p = 0.068$). The p-value was however close to show statistically significance under a 95% significance level.

	Interest/Enjoyment	Perceived Competence	Pressure/ Tension
Control group	5.30	3.93	2.73
Gamification group	6.18	4.24	3.90
Difference	0.88	0.31	1.17
p-value	0.018	0.310	0.068

Table 4.6: Perception of the testing-session.

The subjects' interest before and after the experiment was also measured, which would show whether Gamification could increase subjects' interest in performing software testing. The results, which are reproduced in Table 4.7, show that the Control group rated their interest before the experiment as $\bar{x} = 4.90$, which after the experiment had decreased to $\bar{x} = 4.60$, which however is not statistically significant, $p = 0.407$. The Gamification group on the contrary rated their interest before the experiment as $\bar{x} = 4.64$, which after the experiment had increased to $\bar{x} = 5.29$, this is however not statistically significant either, $p = 0.531$.

	Interest Before Experiment	Interest After Experiment	Difference	p-value
Control group	4.90	4.60	-0.30	0.407
Gamification group	4.64	5.29	0.65	0.531

Table 4.7: Interest in doing the task before and after the experiment.

4.6.2 Perception of the Gamification Elements

The perception of the Gamification elements was measured to get a better picture of how each element was received by the subjects, and which effects each element had on the testing process. By measuring the perception of the game elements, it might be possible in the future to design gamified processes in software engineering that are custom made for the projects where they are implemented. The statements were only asked to the

Gamification group, since they were the only subjects that had used the Gamification elements. The measured perception was for the leaderboard, the badges, the Trace to Test Linkage tool and the storyline (scenario). The rating of each statement ranged from one to seven, and the statistical tests measured whether the variables was statistically different from a mean of four.

Leaderboard

The perception of the leaderboard was measured through four variables: usefulness, enjoyment/interest, perceived competence, and pressure/tension.

The results from the questionnaire are reproduced in Table 4.8. The usefulness of the leaderboard was ranked as $\bar{x} = 4.5$ with standard deviation $\sigma = 1.06$. A one-sample Wilcoxon signed rank test was calculated, which resulted in a p-value of $p = 0.006$, which is statistically significant.

The perceived enjoyment/interest for the leaderboard was on average rated as $\bar{x} = 6.32$ with a standard deviation of $\sigma = 0.70$, which also is statistically significant with a p-value of $p = 0.001$. Furthermore, the perceived competence was rated as $\bar{x} = 4.32$, with a standard deviation of $\sigma = 1.82$, which shows that the rating varied much between each subject. The large standard deviation, and the rating close to the middle, might be explained by the fact that 50% of the subjects at any point in time during the experiment was positioned on the upper half of the leaderboard, and therefore perceived their competence as good, compared to the subjects positioned below them. A similar logic applies to the subjects positioned in the lower half of the leaderboard, which perceived their competence as lower because they are among the 50% with the lowest number of points. Nevertheless, the perceived competence is not significantly different from the middle value four, $p = 0.582$.

The pressure/tension perceived for the leaderboard was rated as $\bar{x} = 5.25$ on average, with a standard deviation of $\sigma = 1.49$. The high standard deviation does also indicate a big difference in rating among the subjects, which might replicate the fact that some subjects are more easily stressed, or that some subjects are more competitive in nature. Moreover, the pressure/tension perceived from the leaderboard is statistically significant with $p = 0.020$ calculated through a one-sample Wilcoxon signed rank test.

	Usefulness	Enjoyment/ Interest	Perceived Competence	Pressure/ Tension
Mean ($\bar{x}$)	5.21	6.32	4.32	5.25
SD (σ)	1.06	0.70	1.82	1.49
p-value	0.006	0.001	0.582	0.020

Table 4.8: Perception of the leaderboard.

Badges

The perception of the badges was measured in regard to usefulness, enjoyment/interest, perceived competence, and pressure/tension. The available badges and levels of these badges are introduced in Appendix A, Section A.2.2. The usefulness was in average rated as $\bar{x} = 3.29$, with a standard deviation of $\sigma = 1.48$; a high standard deviation that indicates that the badges' perceived usefulness differed between the subjects. The rating of usefulness is not statistically different from four ($p = 0.128$), but it does however indicate that the subjects did not find the badges as useful as expected. It might also differ in what the subjects perceived as useful and how they used the badges. The perceived enjoyment/interest of the badges was on the contrary rated as $\bar{x} = 4.77$ with a standard deviation of $\sigma = 1.26$, which might reflect the fact that people differ in what they found interesting, fun and motivating. The enjoyment/interest is not statistically significant ($p = 0.057$), but it is however, an indication that the subjects perceived a positive enjoyment/interest of using the badges. The perceived competence for the badges was rated as $\bar{x} = 4.54$ on average with a standard deviation of $\sigma = 0.81$. The perceived competence is not statistically significant ($p = 0.057$), but a p-value that nearly shows significance gives an indication that the badges increased the perceived competence for the subjects. The perceived pressure/tension was however significantly different from four ($p = 0.032$), with an average of $\bar{x} = 2.68$ and standard deviation of $\sigma = 1.76$.

	Usefulness	Enjoyment/ Interest	Perceived Competence	Pressure/ Tension
Mean ($\bar{x}$)	3.29	4.77	4.54	2.68
SD (σ)	1.48	1.26	0.81	1.76
p-value	0.128	0.063	0.057	0.032

Table 4.9: Perception of the badges.

Trace to Test Linkage Tool

The average perceived usefulness of Trace to Test Linkage was $\bar{x} = 4.50$ with a standard deviation of $\sigma = 1.94$. The large standard deviation might reflect the fact that some people used it more than others, but also that the tool did not give any points and some subjects might therefore have skipped using the tool. The perceived usefulness was not statistically significant ($p = 0.358$). The perceived enjoyment/interest was given an average rating of $\bar{x} = 3.64$ with a standard deviation of $\sigma = 1.27$, which was not significantly different from four ($p = 0.344$). The perceived competence was given an average rating of $\bar{x} = 4.11$ and standard deviation of $\sigma = 1.00$, which through a one sample Wilcoxon signed rank test was shown to not be statistically significant

(p = 0.808). Finally, the perceived pressure/tension was rated low with an average of $\bar{x} = 2.96$ and standard deviation of $\sigma = 1.64$. The perceived pressure/tension was not statistically significant (p = 0.0558), but it showed an indication of being negatively different from four.

	Usefulness	Enjoyment/ Interest	Perceived Competence	Pressure/ Tension
Mean ($\bar{x}$)	4.50	3.63	4.11	2.96
SD (σ)	1.94	1.27	1.00	1.64
p-value	0.358	0.344	0.808	0.0558

Table 4.10: Perception of the Trace to Test Linkage tool.

Fictional Scenario

	Purpose	Enjoyment/Interest
Mean ($\bar{x}$)	5.46	4.89
SD (σ)	1.20	1.25
p-value	0.011	0.037

Table 4.11: Perception of the story, scenario, shown in the start of the testing session.

The Story/Fictional scenario, was given to provide the subjects with purpose and background for the task. The provided Story/Fictional scenario is explained and available in Appendix A, Section A.2.1. The perception of the story was therefore measured accordingly. The variables that were measured were purpose and enjoyment/interest. The purpose was on average rated as $\bar{x} = 5.46$ with a standard deviation of $\sigma = 1.20$. The perceived purpose was statistically significant with $p = 0.011$, calculated through a Wilcoxon signed rank test. The enjoyment/interest was given an average rating of $\bar{x} = 4.89$ and a standard deviation of $\sigma = 1.25$, which through a Wilcoxon signed rank test was calculated to be statistically significant with $p = 0.037$.

5 Discussion

The following sections will answer the research questions stated in Section 1.3, and discuss the results presented in Chapter 4. Section 5.2 will further discuss potential problems with applying Gamification into testing, and whether any results presented in this study support those claims.

5.1 Research questions

Number of defects detected

RQ1: Is there a difference in the number of defects detected by testers when performing unit testing with Gamification applied, than without Gamification applied, in a given time frame?

The experiment carried out in the study resulted in a significantly higher number of defects found by subjects in the Gamification-group than subjects from the Control-group. The null hypothesis is thus rejected and it can, under the conditions and assumptions of the experiment, be concluded that there is a difference in the number of defects found when Gamification is applied to the testing process.

Even if the increase in testing coverage for the Gamification group in regard to instruction and branch coverage are not statistically significant, we recognize the indication of increased testing coverage for the Gamification group in comparison with the Control group. If the difference in instruction and branch coverage would be statistically significant, it could be seen as a contributory factor to the difference in number of found defects.

A correlation between isolated Gamification elements and the number of detected defects was not researched as a part of the study, hence, it could only be concluded that the Gamification element implemented for this study as a group increased the number of defect found, and not how the use of each element contributed to a higher number of found defects. Yet, the results and analysis from the survey show that factors related to interest and enjoyment of the task were significantly higher for the Gamification group than the Control group. This can be seen as the Gamification group being more interested during the task, which might imply that this group was more motivated during the testing session. The motivational factor has been identified as a key factor in Chapter 2, and if the group were more motivated, it might explain the increased number of detected defects.

The subjects' perception of the leaderboard showed a positive perception in both interest/enjoyment, as well as usefulness. These findings were statistically significant within the Gamification group and might therefor lead to an increase in the motivation for the task. Moreover, the leaderboard was perceived by the subjects as being stressful. This fact can increase the pressure among the subjects and lead to more invalid test cases produced by this group in regard to the Control group. However, no statistically significant difference in the number of invalid test cases was found for the groups.

The provided storyline for the Gamification group was perceived to lead to a higher degree of interest/enjoyment as well as purpose among the subjects.

Percentage of requirements covered by test cases

RQ2: Is there a difference in the percentage of requirements covered by tests when performing unit testing with Gamification applied, than without Gamification applied, in a given time frame?

From the results of the Trace to Test Linkage analysis for the two groups we see that the Gamification group covered significantly more requirements than the Control group, as seen in Section 4.3. The null hypothesis is therefore rejected, and we can conclude that a higher number of requirements were covered by the Gamification group under the conditions and assumptions of the experiment.

For the same reasons as discussed in Section 5.1, we cannot conclude that a specific Gamification element had a statistically significance on the percentage of requirements covered by at least one test. We did not find any statistically significant result, in neither the perceived interest/enjoyment, usefulness, perceived competence, nor pressure/tension. In Section 3.4.2 it was stated that 57% of the subjects in the Gamification group used the tool. This was a lower usage than we had expected, and therefore we believe that this tool had little effect on the percentage of requirements covered by the Gamification group. By this assumption we can only conclude that we believe that the general use of Gamification increased the subjects' motivation and interest in the task, which resulted in more test cases produced, and hence more requirements covered.

5.2 Usage of the Plugin

We cannot conclude that the subjects in the Gamification group did not use the plugin as a game where their sole purpose was to collect as many points as possible to increase their position in the leaderboard. However, if this would be the case, there might have been a statistical significance regarding the number of invalid test cases (as a consequence of more mistakes done when the subjects' stress level increased), as well as indications of subjects trying to cheat. Cheating could have been conducted by duplicating test cases which found defects, or writing empty test cases to increase the number of test cases. Since we saw no evidence of this behaviour in the reviews of the test suites, we cannot state that this is an actual concern for this experiment. However, given the short time period of testing we assume that there were subjects stressing to finishing as many requirements as possible to increase their score, even if this did not lead to an increase in the number of invalid test cases. An implementation of Gamification for software testing in industry that only focuses on external rewards might increase the subjects desire to play the game, rather than writing a good test suite. This is why it is desirable to focus more on internal motivation and *Meaningful Gamification*, as discussed in Section 2.3.2, when applying Gamification in an industry context.

6

Validity Threats

The following sections will present and discuss the validity threats identified for this study. The discussed validity threats are a selection from [66], which is based on Wohlin's et al. discussion of validity threats [62].

6.1 Conclusion Validity

Our experiment includes humans as subjects. There is a possibility that these subjects possesses a different set of individual knowledge, experience, and skill in the area. These factors may affect the results of the conducted experiment. We believe that we, through our random sampling, minimized risk of these factors affecting the conclusion validity.

We are encouraging researchers to conduct similar replications of our experiment, but with larger sample sizes, and with a higher percentage of practitioners from industry.

6.2 Internal Validity

All subjects had to complete a survey, providing us with information of previous knowledge in software testing, and programming in Java, before they were selected and allowed to participate in the experiment. We did not verify the answers from the potential subjects, but relied on their truthfulness when defining their previous knowledge. However, since all subjects were signing up voluntarily, and was not rewarded for their participation, we believe that we got a sample of motivated and unbiased subjects for this experiment. If the experiment would be conducted in a course with mandatory participation you would most likely be conducting the experiment with a certain amount of subjects completely lacking motivation for the task, the same applies for rewarding subjects with external rewards.

The provided time for the testing session was one hour, although we would have preferred a longer time frame for the testing session, we believe that a longer experiment would have decreased the motivation to participate in the experiment. This is why we are encouraging researchers to look closer into conducting case studies in industry to see how our results compare to results gathered during a complete project at a company. It can also be valuable to investigate if the testing session can be extended by the Gamification group, in the sense that their willingness to test the system more thoroughly increases. Which might imply a desire to conduct the testing session for a longer time than the Control group, this is not considered in this thesis and is something that we would like to see researcher investigate further.

In our case we also have the possible issue with subjects playing a game during the testing session in the Gamification group, solely focusing on collecting points and missing the actual task of locating defects. We do however believe that we would see a statistically significance in the increase of the number of invalid test cases if this was the case for our study, which was not observed. We encourage researchers to look closer into these aspects when conducting similar studies.

6.3 Construct Validity

The first factor to the construct validity is seeding of faults. There are arguments saying that fault seeding may be done biased by the researchers since it is often done using unrealistic faults [67]. We believe that we minimized this risk by using our algorithm for the location to seed a fault in the class, described in Section 3.2, as well as the use of Table 3.3 to select a fault to introduce. We cannot say that we have fully mitigated this threat, but we have minimized our own bias in selecting places to seed faults, and what fault to seed.

We measured the effectiveness of the testing techniques, with and without Gamification applied, based on the number of detected defects and the percentage of requirements covered by tests. There might have been other interesting aspects to study in the test suits produced by the subjects. But the focus for this study, however, was put on the number of found defects and the requirement coverage. Studying the quality of the test suites or the efficiency of the two groups may lead to new findings about the impact of Gamification. This is yet another aspect that we encourage researchers to investigate.

6.4 External Validity

In our sample for this experiment we had twenty-four subjects, seven of which were from industry. This gives a percentage of 29% subjects from industry. It should be stated that the average working experience for the group of industry people was <2 years. We see this as a threat to the validity about generalizing the findings, under the stated conditions in this experiment, to industrial practices. We do however believe that the idea of Gamification combined with testing can be generalized to the industry, but not under the proposed environment in our experiment. The environment requires a testing object where the defects can be identified when running a test suite. We do not propose that researchers try to implement our plugin into industry, but rather try to introduce Gamification in an organisational context.

We believe that the specific time frame in this experiment for the subjects to conduct the testing session makes any real generalizations to industry hard in the given context. In a real project, a live leaderboard, as the one provided in the experiment would not make sense, but a long going implementation of something equal which provides a summary over the whole project time may be possible to provide.

7

Conclusion and Future Work

This study has shown that Gamification can have a positive impact on the number of defects detected and the percentage of requirements covered by tests in software testing. The conducted experiment for this study shows that Gamification made the testing phase more fun and interesting, but also increased the pressure on the testers. The study has not concluded which Gamification element(s) that might have increased the number of detected defects and the percentage of requirements covered by tests, but we believe that the increase in interest/enjoyment had a major impact. The quantitative data that were gathered from the subjects of the study does however conclude, with statistical significance, that the leaderboard was useful, and promoted enjoyment/interest, but also increased the pressure/tension felt by the subjects. The badges seemed to positively promote enjoyment/interest and perceived competence, but the data were not statistically significant. Moreover, the study found that the badge system was not significantly perceived as stressful by the subjects. Furthermore, the Trace to Test Linkage tool was not perceived as useful nor fun, and it did not lead to any difference in the perceived competence by the subjects even if there were a statistical significance in the percentage of requirements covered by tests, which implies that the Gamification itself might have increased this. Finally, the storyline shown to the subjects before the experiment started was found to statistically increase the purpose as well as the enjoyment in the task. These findings indicates that even if it cannot be stated which Gamification element(s) that were the main reason for the increased effectiveness, it can be stated that when the interest and motivation increased, and a purpose was provided to the Gamification group, this group performed better under the stated conditions. In summary we conclude that Gamification can have positive effects in software testing, but that all Gamification elements might not work, and some elements might need to be tailored to the specific organization where it will be used. There is also a risk that Gamification can have side effects which are not taken into consideration when the Gamification environment is designed. This last remark is the reason we encourage researchers to conduct more research in this area.

7.1 Future Work

As a result of this study introducing a new concept in software testing, Gamification, we see the potential for a broad variety of research in this area.

First, we encourage researchers to conduct more controlled experiments similar to our experiment, where Gamification is introduced into the testing technique selected against a control group. This experiment has focused on unit testing but from our gathered knowledge from this thesis we do not recognize any specific testing technique where implementing Gamification would not be possible or lose its purpose.

We would also like to see researchers moving the experiments out of the controlled environments and conduct case studies at companies to study the effect of Gamification in organizations.

One aspect that our study does not investigate deep enough is, if any, what Gamification element(s) is responsible for the significant increase in the results, or if it is a combination of several elements. Yet, we believe that the increased and immediate feedback that was given to the subjects using the Gamification plugin had positive effects on the results in the experiment. More research is needed in this area to be able to state anything regarding the correlation between Gamification aspects and immediate user feedback.

We do also encourage the industry to study how Gamification can be implemented in their own software engineering practices, not only in software testing. Since we believe that we have shown positive results in our experiment when applying Gamification to testing, we believe that this should be investigated even further in more fields within Software Engineering.

When implementing Gamification in any software engineering practice in industry we strongly encourage the implementers to be careful in regard to the possible demotivating aspects of Gamification. These are the aspects that for example a leaderboard can introduce when practitioners in the bottom section might perceive their competence as inadequate in regard to other practitioners in the team. If for example Gamification is implemented in a testing process, it needs to be implemented in such a way that it rewards all practitioners, even if they are testing different aspects of the system. This should be done by taking into consideration that one section of the system may be ”easier” to test and result in more points than a more complex section of system. Weighing the potential use of a reward system, and promoting practitioners to not only focusing on easy sections of the system, is a vital part of implementing Gamification successfully in a testing process.

Bibliography

[1] G. J. Myers, C. Sandler, T. Badgett, The art of software testing, 3rd Edition, John Wiley & Sons, Hoboken, N.J.

[2] G. Tassey, The economic impacts of inadequate infrastructure for software testing, National Institute of Standards and Technology, RTI Project 7007.

[3] D. Z. Alrmuny, Open problems in software test coverage, Lecture Notes on Software Engineering 2 (1).

[4] A. Bertolino, The (im) maturity level of software testing, ACM SIGSOFT Software Engineering Notes 29 (5) (2004) 1–4.

[5] J. A. Whittaker, What is software testing? and why is it so hard?, Software, IEEE 17 (1) (2000) 70–79.

[6] B. Lawrence, K. Wiegers, C. Ebert, The top risk of requirements engineering, Software, IEEE 18 (6) (2001) 62–63.

[7] S. Deterding, D. Dixon, R. Khaled, L. Nacke, From game design elements to gamefulness: defining gamification, in: Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments, ACM, 2011, pp. 9–15.

[8] C. Eickhoff, C. G. Harris, A. P. de Vries, P. Srinivasan, Quality through flow and immersion: gamifying crowdsourced relevance assessments, in: Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval, ACM, pp. 871–880.

[9] S. Grant, B. Betts, Encouraging user behaviour with achievements: an empirical study (2013).

[10] J. H. Jung, C. Schneider, J. Valacich, Enhancing the motivational affordance of information systems: The effects of real-time performance feedback and goal setting in group collaboration environments, Management Science 56 (4) (2010) 724–742. URL `http://pubsonline.informs.org/doi/abs/10.1287/mnsc.1090.1129`

[11] M. Witt, C. Scheiner, S. Robra-Bissantz, Gamification of online idea competitions: Insights from an explorative case, Informatik schafft Communities (2011) 192.

[12] D. Martin, J. Rooksby, M. Rouncefield, I. Sommerville, 'good' organisational reasons for 'bad' software testing: An ethnographic study of testing in a small software company, in: Software Engineering, 2007. ICSE 2007. 29th International Conference on, 2007, pp. 602–611.

[13] H. Shah, M. J. Harrold, Studying human and social aspects of testing in a service-based software company: case study (2010).

[14] T. Kanij, R. Merkel, J. Grundy, An empirical study of the effects of personality on software testing, in: Software Engineering Education and Training (CSEE&T), 2013 IEEE 26th Conference on, 2013, pp. 239–248.

[15] N. Juristo, A. M. Moreno, S. Vegas, Reviewing 25 years of testing technique experiments, Empirical Software Engineering 9 (1-2) (2004) 7–44.

[16] A. Bertolino, Software testing research: Achievements, challenges, dreams (2007).

[17] T. Vos, B. Marın, I. Panach, A. Baars, C. Ayala, X. Franch, Evaluating software testing techniques and tools, Proceedings of JISBD 2011 (2013) 531–536.

[18] S. Eldh, H. Hansson, S. Punnekkat, A. Pettersson, D. Sundmark, A framework for comparing efficiency, effectiveness and applicability of software testing techniques, in: Testing: Academic and Industrial Conference - Practice And Research Techniques, 2006. TAIC PART 2006. Proceedings, 2006, pp. 159–170.

[19] T. F. Hammer, L. L. Huffman, L. H. Rosenberg, W. Wilson, L. Hyatt, Doing requirements right the first time, CROSSTALK The Journal of Defense Software Engineering (1998) 20–25.

[20] A. Marczewski, Gamification: a simple introduction, Andrzej Marczewski, 2013.

[21] Google, Gamification at google trends, accessed: 2014-02-11 (2014).
URL `http://www.google.com/trends/explore#q=gamification`

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

[23] K. Huotari, J. Hamari, Defining gamification: a service marketing perspective, in: Proceeding of the 16th International Academic MindTrek Conference, ACM, 2012, pp. 17–22.

[24] G. Hertel, S. Niedner, S. Herrmann, Motivation of software developers in open source projects: an internet-based survey of contributors to the linux kernel, Research policy 32 (7) (2003) 1159–1177.

[25] J. Moreno, Digital competition game to improve programming skills., Educational Technology & Society 15 (3) (2012) 288–297.

[26] Gamification by design; implementing game mechanics in web and mobile apps, Vol. 26, 2011, copyright - Copyright Book News, Inc. Dec 2011; Last updated - 2013-07-02.
URL `http://search.proquest.com/docview/906481809?accountid=10041`

[27] A. Anderson, D. Huttenlocher, J. Kleinberg, J. Leskovec, Steering user behavior with badges, in: Proceedings of the 22nd international conference on World Wide Web, International World Wide Web Conferences Steering Committee, 2013, pp. 95–106.

[28] J. Hamari, J. Koivisto, H. Sarsa, Does gamification work?—a literature review of empirical studies on gamification, in: Proceedings of the 47th Hawaii International Conference on System Sciences. HICSS, 2014.

[29] S. Deterding, Situated motivational affordances of game elements: A conceptual model, in: Gamification: Using Game Design Elements in Non-Gaming Contexts, a workshop at CHI, 2011.

[30] E. B. Passos, D. B. Medeiros, P. A. Neto, E. W. Clua, Turning real-world software development into a game, in: Games and Digital Entertainment (SBGAMES), 2011 Brazilian Symposium on, IEEE, 2011, pp. 260–269.

[31] D. J. Dubois, G. Tamburrelli, Understanding gamification mechanisms for software development, in: Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering, ACM, pp. 659–662.

[32] S. Sheth, J. Bell, G. Kaiser, Halo (highly addictive, socially optimized) software engineering (2011).

[33] M. Csikszentmihalyi, Beyond boredom and anxiety, Jossey-Bass, 2000.

[34] J. P. Charlton, I. D. Danforth, Distinguishing addiction and high engagement in the context of online game playing, Computers in Human Behavior 23 (3) (2007) 1531–1548.

[35] J. Bell, S. Sheth, G. Kaiser, Secret ninja testing with halo software engineering (2011).

[36] S. Sheth, J. Bell, G. Kaiser, A competitive-collaborative approach for introducing software engineering in a cs2 class, in: Software Engineering Education and Training (CSEE&T), 2013 IEEE 26th Conference on, IEEE, 2013, pp. 41–50.

[37] S. Nicholson, A user-centered theoretical framework for meaningful gamification, Proceedings GLS 8.

[38] J. Bitzer, W. Schrettl, P. J. Schröder, Intrinsic motivation in open source software development, Journal of Comparative Economics 35 (1) (2007) 160–169.

[39] M. R. Lepper, D. Greene, R. E. Nisbett, Undermining children's intrinsic interest with extrinsic reward: A test of the" overjustification" hypothesis., Journal of personality and social psychology 28 (1) (1973) 129.

[40] S. Halan, B. Rossen, J. Cendan, B. Lok, High Score! - Motivation Strategies for User Participation in Virtual Human Development, Vol. 6356 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2010, book section 52, pp. 482–488. URL `http://dx.doi.org/10.1007/978-3-642-15892-6_52`

[41] W. Li, T. Grossman, G. Fitzmaurice, Gamicad: a gamified tutorial system for first time autocad users, in: Proceedings of the 25th annual ACM symposium on User interface software and technology, ACM, 2012, pp. 103–112.

[42] G. Barata, S. Gama, J. Jorge, D. Gonçalves, Improving participation and learning with gamification.

[43] R. Farzan, J. M. DiMicco, D. R. Millen, C. Dugan, W. Geyer, E. A. Brownholtz, Results from deploying a participation incentive mechanism within the enterprise (2008).

[44] R. Farzan, P. Brusilovsky, Encouraging user participation in a course recommender system: An impact on user behavior, Computers in Human Behavior 27 (1) (2011) 276–284.

[45] J. Thom, D. Millen, J. DiMicco, Removing gamification from an enterprise sns (2012).

[46] A. F. Aparicio, F. L. G. Vela, J. L. G. Sánchez, J. L. I. Montes, Analysis and application of gamification, in: Proceedings of the 13th International Conference on Interacción Persona-Ordenador, ACM, 2012, p. 17.

[47] R. M. Ryan, C. S. Rigby, A. Przybylski, The motivational pull of video games: A self-determination theory approach, Motivation and Emotion 30 (4) (2006) 347–363.

[48] R. M. Ryan, E. L. Deci, Self-determination theory and the facilitation of intrinsic motivation, social development, and well-being., American psychologist 55 (1) (2000) 68.

[49] C. P. Niemiec, R. M. Ryan, Autonomy, competence, and relatedness in the classroom applying self-determination theory to educational practice, Theory and Research in Education 7 (2) (2009) 133–144.

[50] C. Mihaly, Flow: the psychology of optimal experience (1990).

[51] F. Groh, Gamification: State of the art definition and utilization, in: Proceedings of the 4th seminar on Research Trends in Media Informatics, 2012, pp. 39–46.

[52] J. Hamari, V. Eranti, Framework for designing and evaluating game achievements, Proc. DiGRA 2011: Think Design Play 115 (2011) 122–134.

[53] J. Cameron, W. D. Pierce, Reinforcement, reward, and intrinsic motivation: A meta-analysis, Review of Educational research 64 (3) (1994) 363–423.

[54] C. I. Muntean, Raising engagement in e-learning through gamification, in: Proc. 6th International Conference on Virtual Learning ICVL, 2011, pp. 323–329.

[55] J. Fernandes, D. Duarte, C. Ribeiro, C. Farinha, J. M. Pereira, M. M. d. Silva, ithink: A game-based approach towards improving collaboration and participation in requirement elicitation, Procedia Computer Science 15 (2012) 66–77.

[56] A. Dorling, F. McCaffery, The gamification of spice, in: Software Process Improvement and Capability Determination, Springer, 2012, pp. 295–301.

[57] J. M. Tauer, J. M. Harackiewicz, Winning isn't everything: Competition, achievement orientation, and intrinsic motivation, Journal of Experimental Social Psychology 35 (3) (1999) 209–238.

[58] U. Jayasinghe, A. Dharmaratne, Game based learning vs. gamification from the higher education students' perspective, in: Teaching, Assessment and Learning for Engineering (TALE), 2013 IEEE International Conference on, IEEE, 2013, pp. 683–688.

[59] M. Ebner, A. Holzinger, Successful implementation of user-centered game based learning in higher education: An example from civil engineering, Computers & education 49 (3) (2007) 873–890.

[60] A.-L. Smith, L. Baker, Getting a clue: creating student detectives and dragon slayers in your library, Reference Services Review 39 (4) (2011) 628–642.

[61] D. R. Flatla, C. Gutwin, L. E. Nacke, S. Bateman, R. L. Mandryk, Calibration games: making calibration tasks enjoyable by adding motivating game elements, in: Proceedings of the 24th annual ACM symposium on User interface software and technology, ACM, 2011, pp. 403–412.

[62] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in software engineering, Springer, 2012.

[63] J. Cohen, Statistical power analysis for the behavioral sciences (rev, Lawrence Erlbaum Associates, Inc, 1977.

[64] L. Madeyski, N. Radyk, Judy - a mutation testing tool for java, Software, IET 4 (1) (2010) 32–42.

[65] N. Tsigilis, A. Theodosiou, Temporal stability of the intrinsic motivation inventory, Perceptual and Motor Skills 97 (1) (2003) 271–280.
URL `http://dx.doi.org/10.2466/pms.2003.97.1.271`

[66] R. Feldt, A. Magazinius, Validity threats in empirical software engineering research-an initial survey., in: SEKE, 2010, pp. 374–379.

[67] L. C. Briand, A critical analysis of empirical research in software testing, in: Empirical Software Engineering and Measurement, 2007. ESEM 2007. First International Symposium on, IEEE, 2007, pp. 1–8.

Appendices

A

Plugin

The features that are present in both plugins are information about the test suite, test coverage information, requirement specification, JUnit API, and upload functionality for the project.

The basic information about the test suite contains the tester's name, number of test cases, number of passed test cases and number of failures/errors in test suite. The failures/errors are identified as either a found defect or an incorrect test case by the plugin.

The test coverage information includes the overall test coverage for the MyLinkedList class, Statement and Branch coverage as well as coverage for each of the methods present in the MyLinkedList class.

The requirement that the MyLinkedList class should adhere to, which are the same as for the original LinkedList class are available from the plugin in a new window with a search function to easily identify a specific requirement.

The JUnit API allows the participant to browse the JUnit's official API[1].

The upload of the source files are used to store the participants' source files on an FTP-server for further analysis after the experiment.

The plugin creates a test report when building the test suite, which later is parsed, to retrieve all the data that are necessary to provide the instant feedback to the participants, and for the post experiment analysis. The second report that is created is a test coverage report which is used to parse the data about the test coverage for the test suite, this information is also collected and provided to the participant. The last report that is created by the plugin is a logfile with a timestamp for each run which contains the combined information of the two reports, as well as information provided by the plugin when parsing the data. This is used to analyze how the participant's test suite evolves during the experiment session.

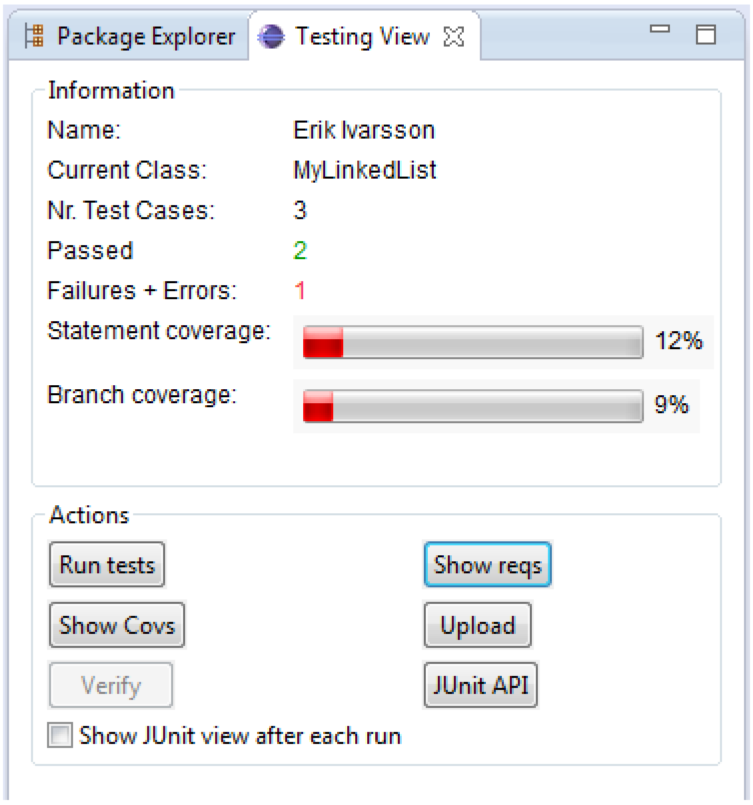
Verifying the failures and errors found in MyLinkedList class as actual defects by

[1]http://junit.sourceforge.net/javadoc/org/junit/Assert.html

running the same test suite against a original class made sense since the fictional scenario states that the MyLinkedList class should adhere to the same requirements as the original LinkedList class. Nonetheless, a failure, or error, caused by a developer making the wrong assumptions for a method under test, is identified if it causes a failure/error in both the execution for MyLinkedList and the original LinkedList class.

A.1 Standard Version

For the Standard version used in the Control group there were only basic information regarding number of test cases, number of failures/errors, number of passed test cases and the testing coverage for the test suite. This data were visualized to the tester to simulate the regular environment with our underlying data gathering and database connection available. The Standard version of the plugin can be seen in Figure A.1.



Name:	Erik Ivarsson
Current Class:	MyLinkedList
Nr. Test Cases:	3
Passed	2
Failures + Errors:	1
Statement coverage:	12%
Branch coverage:	9%

Figure A.1: Overview of the Standard plugin.

A.2 Gamification Version

In addition to the standard information and feedback to the subject discussed in previous sections, the Gamification group's plugin provides a Gamification layer which has been added to the standard plugins user interface. This user interface provides feedback using Gamification elements. The used Gamification elements were the following:

- Fictional scenario about the purpose of the task
- Badges
- Point system
- Leaderboard
- Trace-to-Test Linkage tool
- Instant feedback

In Figure A.3 the complete layout of the plugin is visible.

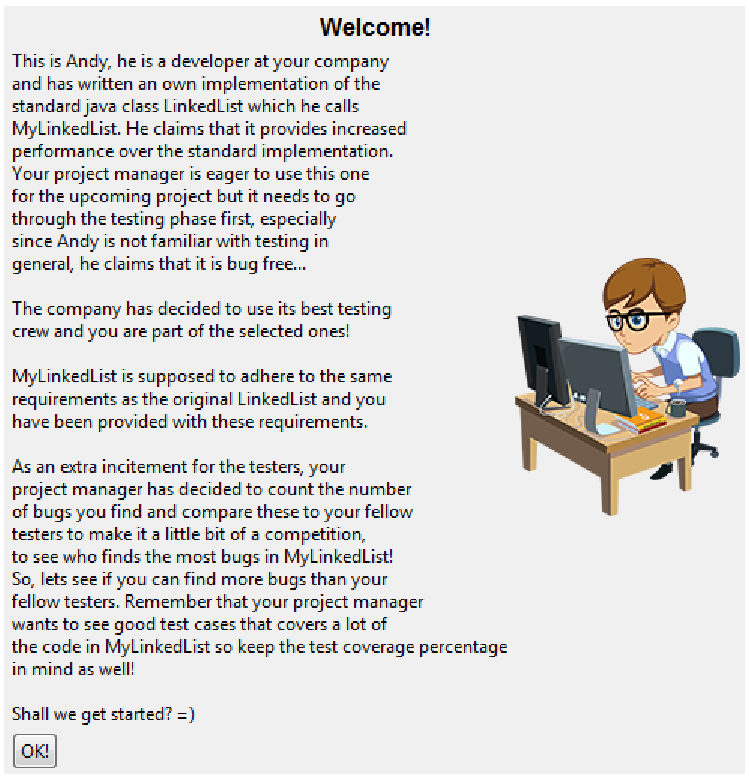
A.2.1 Fictional Scenario

When the software tester started using the plugin, she was presented with a welcome screen that introduced her to a fictional scenario. In the stated scenario the subject was a member of company X's testing crew and was tasked with testing a java class called MyLinkedList, the scenario stated that this was a new implementation of the standard java class LinkedList and that it needed to go through the testing phase before it could be used in the upcoming project. The main purpose of the scenario was to locate as many of the potential defects presented in MyLinkedList. The scenario also stated that the team's project manager had created a small competitive environment by deciding to summarizing the stats from the testers test runs to see if the testers found any defects in the MyLinkedList class and compare these to the other testers in a leaderboard, while simultaneously summarizing information about other aspects of the test runs such as testing coverage and number of test cases. The fictional scenario provided upon the start up is seen in Figure A.2

A.2.2 Badges

The plugin provides the subjects with a selection of collectible badges, these badges are rewarded for achieving a specific predefined milestone during the testing session. The available badges include five visible badges, with a predefined goal, that have to be fulfilled before receiving the badge. It does also exist a hidden badge, where the predefined goal is hidden. The defined visible badges have three levels: bronze, silver and gold. Reaching a new higher level is awarded with a specific amount of points. The hidden badge has the same levels but these levels do not reward the user with any points. The following badges exists in the plugin.

- **TestCaseWriter**: Achieved for writing a certain amount of test cases.
- **TestAddictor**: Achieved for writing test cases that passes.



Welcome!

This is Andy, he is a developer at your company and has written an own implementation of the standard java class LinkedList which he calls MyLinkedList. He claims that it provides increased performance over the standard implementation. Your project manager is eager to use this one for the upcoming project but it needs to go through the testing phase first, especially since Andy is not familiar with testing in general, he claims that it is bug free...

The company has decided to use its best testing crew and you are part of the selected ones!

MyLinkedList is supposed to adhere to the same requirements as the original LinkedList and you have been provided with these requirements.

As an extra incitement for the testers, your project manager has decided to count the number of bugs you find and compare these to your fellow testers to make it a little bit of a competition, to see who finds the most bugs in MyLinkedList! So, lets see if you can find more bugs than your fellow testers. Remember that your project manager wants to see good test cases that covers a lot of the code in MyLinkedList so keep the test coverage percentage in mind as well!

Shall we get started? =)

Figure A.2: The fictional scenario provided upon start up.

- **BugKiller**: Achieved for writing test cases that locate defects.
- **CoverageIncreaser**: Achieved for reaching a specific percentage of Instruction and Branch coverage, average between these two for the whole MyLinkedList.
- **MethodsCoverageIncreaser**: Achieved for reaching over 75 percentage test coverage in a specific number of methods in MyLinkedList.
- **Req. Coverage**: Achieved for connecting a certain amount of test cases to requirements in the Trace to Test Linkage tool (Hidden, until the first milestone is reached).

A.2.3 Point system

The developed point system combined stats about number of found defects, test coverage, number of test cases and the values from the earned badges to calculate the specific score which the leaderboard is based on. The exact algorithm for the point system was not known by the subjects, but the weight about the different variables was explained, and the value for each badge was stated.

A.2.4 Leaderboard

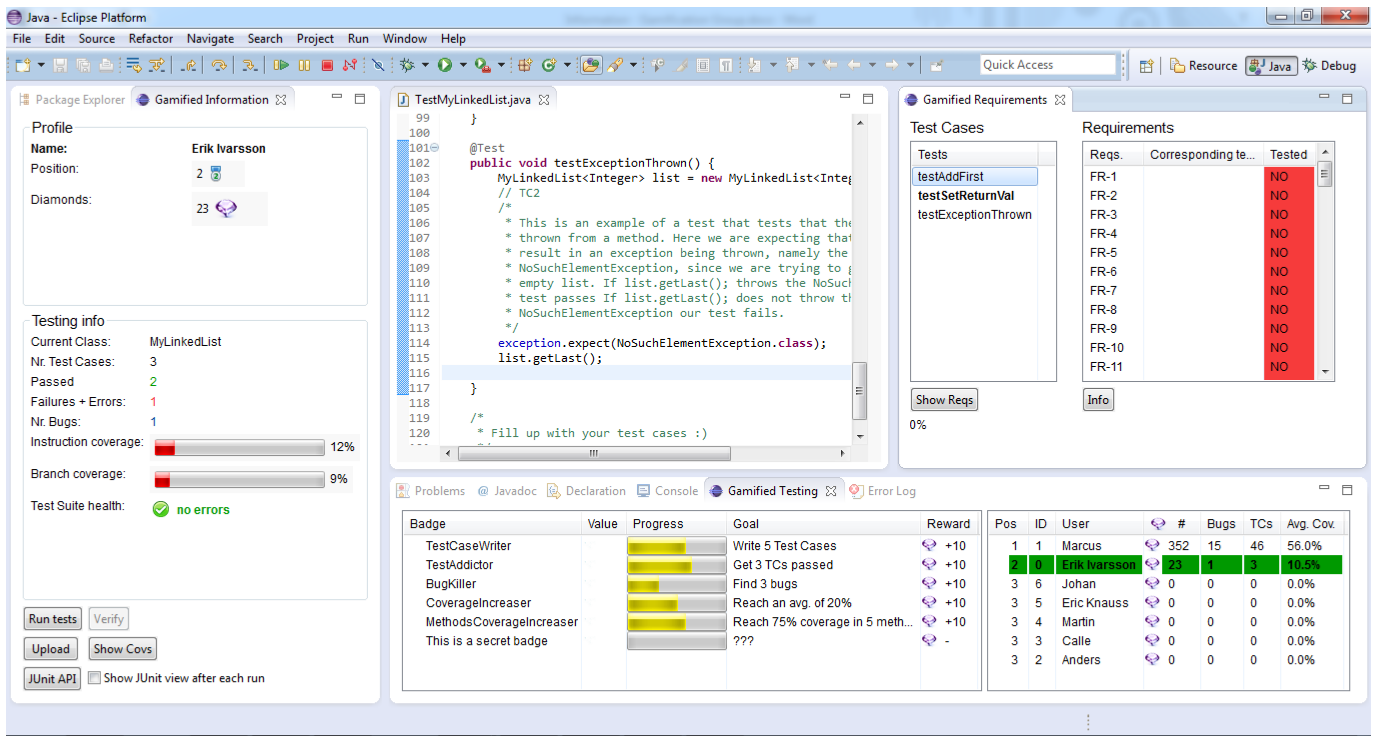
The leaderboard summarized the variables on which the overall point system was calculated from. The leaderboard combined information from the overall score, number of defects, number of test cases, and the average test coverage for each of the subjects using the plugin, this means that there are no data from subjects in the other group.

A.2.5 Trace to Test Linkage Tool

In this helper tool the subject has her test cases which are visible after the run and all of the requirements for the MyLinkedList class. By drag and dropping test cases to a corresponding requirement the subject can easily keep track of her progress in testing the requirement and is presented with a percentage of the number of covered requirements. This connection is completely done by the subject and does not contain any check whether the test case that the subject connects to the specific requirement actually tests the requirement. Using this tool unlocks the hidden badge but does not reward the subject with any points other than it may increase the subject will to increase the number of requirements covered by tests, by setting up own goals for the use of this tool.

A.2.6 Instant Feedback

The instant feedback to the subject updates after each run for the information of the test suite (number of test cases, number of passed test cases, failures, errors, number of defects found and the test coverage) which resulted in instant updates of the badges and point system. The plugin recognizes if the subjects have increased the number of defects found or a new level for a badge was achieved and present this in a new window which summarizes the subjects achievements and alerts the subject that a new milestone was reached. The leaderboard is updating information from the database every five seconds which creates a live leaderboard for the subjects to see their current position in comparison with other subjects from the Gamification group. The Trace-to-Test Linkage tool provides instant feedback about the number of requirements that the tester has marked as tested, and the progress is visualized by a percentage of the requirements marked as tested.



Profile

Name:	Erik Ivarsson
Position:	2
Diamonds:	23

Testing info

Current Class:	MyLinkedList
Nr. Test Cases:	3
Passed	2
Failures + Errors:	1
Nr. Bugs:	1
Instruction coverage:	12%
Branch coverage:	9%
Test Suite health:	no errors

Show JUnit view after each run

```
    }

    @Test
    public void testExceptionThrown() {
        MyLinkedList<Integer> list = new MyLinkedList<Integ
        // TC2
        /*
         * This is an example of a test that tests that the
         * thrown from a method. Here we are expecting that
         * result in an exception being thrown, namely the
         * NoSuchElementException, since we are trying to g
         * empty list. If list.getLast(); throws the NoSuch
         * test passes If list.getLast(); does not throw th
         * NoSuchElementException our test fails.
         */
        exception.expect(NoSuchElementException.class);
        list.getLast();

    }

    /*
     * Fill up with your test cases :)
```

Test Cases

Tests
testAddFirst
testSetReturnVal
testExceptionThrown

0%

Requirements

Reqs.	Corresponding te...	Tested
FR-1		NO
FR-2		NO
FR-3		NO
FR-4		NO
FR-5		NO
FR-6		NO
FR-7		NO
FR-8		NO
FR-9		NO
FR-10		NO
FR-11		NO

Badge	Value	Progress	Goal	Reward
TestCaseWriter			Write 5 Test Cases	+10
TestAddictor			Get 3 TCs passed	+10
BugKiller			Find 3 bugs	+10
CoverageIncreaser			Reach an avg. of 20%	+10
MethodsCoverageIncreaser			Reach 75% coverage in 5 meth...	+10
This is a secret badge			???	-

Pos	ID	User	#	Bugs	TCs	Avg. Cov.
1	1	Marcus	352	15	46	56.0%
2	0	Erik Ivarsson	23	1	3	10.5%
3	6	Johan	0	0	0	0.0%
3	5	Eric Knauss	0	0	0	0.0%
3	4	Martin	0	0	0	0.0%
3	3	Calle	0	0	0	0.0%
3	2	Anders	0	0	0	0.0%

Figure A.3: Complete overview of the Gamification plugin.

B

Questions for the Intrinsic Motivation Inventory

The statements and questions asked to the subjects of the experiment are reproduced below.

B.1 General Information

Thank you for conducting our experiment!

Before you leave we would like you to fill in the following questionnaire. The following pages contain several statements that ought to evaluate your perception and experience in performing the experiment. For each statement, you are asked to indicate on a scale 1 to 7, how true the statement is for you. Answer each statement from the perspective of being a software developer, without thinking too much on each statement.

If you have any questions or need translation, please ask the moderator of the experiment (Marcus or Erik). There are 10 questions in this survey.

B.2 General Questions (both groups)

General questions about you

(Your personal data is handled with care, and will only be used to group your answers on this survey with your results in the experiment. Your name or other personal data won't show up in any reports of any kind.)

1. State the unique ID you was assigned for the experiment. *
2. How many years have you studied Computer Science/Software Engineering or relevant subjects, at university level? * (This question includes all computer related studies at a university or (Sv. högskola). Example of fields: Computer Science, Software Engineering, Information Technology, Systems Engineering etc.)

   Please choose only one of the following:

   (a) 1 Year
   (b) 2 Years
   (c) 3 Years
   (d) 4 Years
   (e) 5 Years
   (f) 6 or more
3. How many years of industry experience do you have? * (With industry experience means the years of experience you have in working with programming/-software in industry.)

   Please choose only one of the following:

   (a) 0 Years
   (b) 1-5
   (c) 6-10
   (d) 11-15
   (e) 16-20
   (f) 21 or more.
4. Do you have experience in testing *

   Please choose only one of the following:

   (a) Yes

(b) No

5. Do you have experience with testing in jUnit? *

   Please choose only one of the following:

   (a) Yes
   (b) No

B.3 Questions About the Task Performed (both groups)

General questions about the experiment

For each of the following statements, please indicate how true it is for you. Please choose the appropriate response for each item:

1 Not true at all	2	3	4	5	6	7 Very true

1. Before I did the experiment, I found the task interesting.
2. I did not feel at all nervous about doing the task.
3. I found the task very interesting.
4. I felt tense while doing the task.
5. I think I did well at this activity compared to other participants.
6. Doing the task was fun.
7. I am NOT satisfied with my performance at this task.
8. I thought the task was boring.
9. I felt pressured while doing the task.
10. After working at this task for awhile, I felt pretty competent.

B.4 Gamification Elements (gamification-group)

B.4.1 The Leaderboard

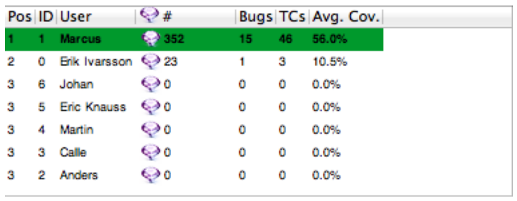
Questions about the leaderboard

Below you find statements regarding your perception of the leaderboard. For each statement, please indicate how true each statement are for you. Please choose the appropriate response for each item:

1 Not true at all	2	3	4	5	6	7 Very true

Note: The ”task” refer to the task of testing during the experiment, and how the leaderbord affected your testing.

The Leaderboard



Pos	ID	User	#	Bugs	TCs	Avg. Cov.
1	1	Marcus	352	15	46	56.0%
2	0	Erik Ivarsson	23	1	3	10.5%
3	6	Johan	0	0	0	0.0%
3	5	Eric Knauss	0	0	0	0.0%
3	4	Martin	0	0	0	0.0%
3	3	Calle	0	0	0	0.0%
3	2	Anders	0	0	0	0.0%

1. The leaderboard didn’t make me stressed.
2. The leaderboard made this task more fun.
3. The leaderboard made the task more boring.
4. The leaderboard made me feel pressured while doing the task.
5. The leaderboard made me feel more competent in doing the task
6. The leaderboard made me feel less skilled in doing this task than other participants.
7. The leaderboard made the task more motivating.
8. I found the leaderboard useful in performing the task.
9. I did not use or look at the leaderboard.
10. I didn’t found the leaderboard to motivate me in performing the task.

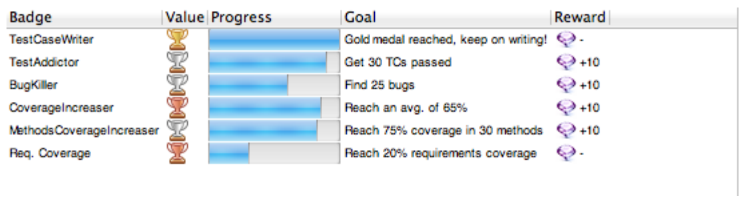
B.4.2 The Badges

Questions about the badges

Below you find statements regarding your perception of the badges. For each statement, please indicate how true each statement are for you. Please choose the appropriate response for each item:

1 Not true at all	2	3	4	5	6	7 Very true

The Badges



Badge	Value	Progress	Goal	Reward
TestCaseWriter	🏆		Gold medal reached, keep on writing!	-
TestAddictor	🏆		Get 30 TCs passed	+10
BugKiller	🏆		Find 25 bugs	+10
CoverageIncreaser	🏆		Reach an avg. of 65%	+10
MethodsCoverageIncreaser	🏆		Reach 75% coverage in 30 methods	+10
Req. Coverage	🏆		Reach 20% requirements coverage	-

Note: The "task" refer to the task of testing during the experiment, and how the badges affected your testing.

1. The badges didn't make me stressed.
2. The badges made this task more fun.
3. The badges made the task more boring.
4. The badges made me feel pressured while doing the task.
5. The badges made me feel more competent in doing the task
6. The badges made me feel less skilled in doing this task than other participants.
7. The badges made the task more motivating.
8. I found the badges useful in performing the task.
9. I did not use or look at the badges.
10. I didn't found the badges to motivate me in performing the task.

B.4.3 The Trace To Test Linkage Tool

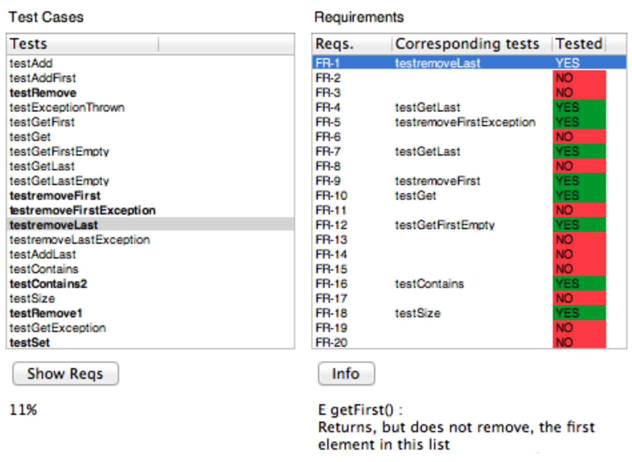
Questions about the Trace To Test Linkage

Below you find statements regarding your perception of the Trace To Test linkage. For each statement, please indicate how true each statement are for you. Please choose the appropriate response for each item:

1 Not true at all	2	3	4	5	6	7 Very true

Note: The "task" refer to the task of testing during the experiment, and how the Trace To Test linkage affected your testing.

The Trace To Test Linkage


Test Cases

Tests
testAdd
testAddFirst
testRemove
testExceptionThrown
testGetFirst
testGet
testGetFirstEmpty
testGetLast
testGetLastEmpty
testremoveFirst
testremoveFirstException
testremoveLast
testremoveLastException
testAddLast
testContains
testContains2
testSize
testRemove1
testGetException
testSet

Requirements

Reqs.	Corresponding tests	Tested
FR-1	testremoveLast	YES
FR-2		NO
FR-3		NO
FR-4	testGetLast	YES
FR-5	testremoveFirstException	YES
FR-6		NO
FR-7	testGetLast	YES
FR-8		NO
FR-9	testremoveFirst	YES
FR-10	testGet	YES
FR-11		NO
FR-12	testGetFirstEmpty	YES
FR-13		NO
FR-14		NO
FR-15		NO
FR-16	testContains	YES
FR-17		NO
FR-18	testSize	YES
FR-19		NO
FR-20		NO

11%

E getFirst() :
Returns, but does not remove, the first element in this list

1. The Trace To Test linkage didn't make me stressed.
2. The Trace To Test linkage made this task more fun.
3. The Trace To Test linkage made the task more boring.
4. The Trace To Test linkage made me feel pressured while doing the task.
5. The Trace To Test linkage made me feel more competent in doing the task
6. The Trace To Test linkage made me feel less skilled in doing this task than other participants.
7. The Trace To Test linkage made the task more motivating.

8. I found the Trace To Test linkage useful in performing the task.
9. I did not use or look at the Trace To Test linkage.
10. I didn't found the Trace To Test linkage to motivate me in performing the task.

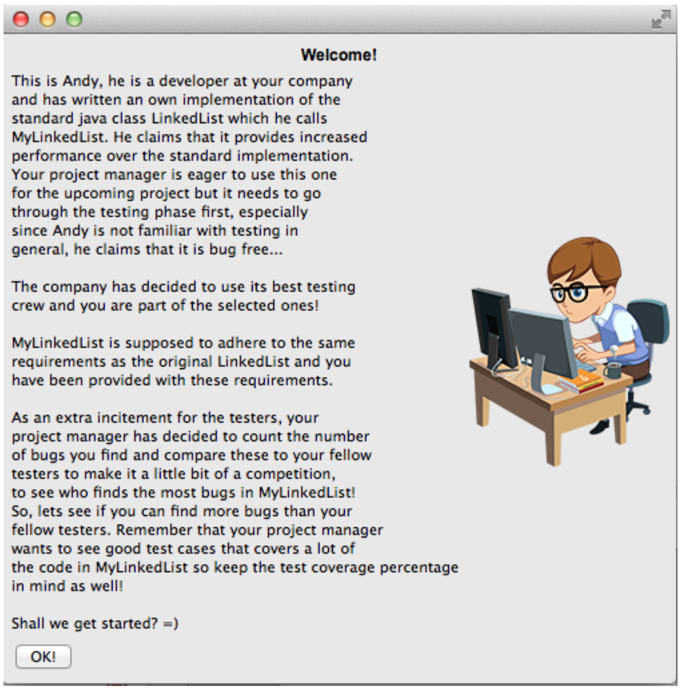
B.4.4 The Welcome Screen

Questions about the welcome scenario

Below you find statements regarding your perception of the welcome scenario, which was shown after you had verified yourself. For each statement, please indicate how true each statement are for you. Please choose the appropriate response for each item:

1 Not true at all	2	3	4	5	6	7 Very true

The Welcome Scenario



Welcome!

This is Andy, he is a developer at your company and has written an own implementation of the standard java class LinkedList which he calls MyLinkedList. He claims that it provides increased performance over the standard implementation. Your project manager is eager to use this one for the upcoming project but it needs to go through the testing phase first, especially since Andy is not familiar with testing in general, he claims that it is bug free...

The company has decided to use its best testing crew and you are part of the selected ones!

MyLinkedList is supposed to adhere to the same requirements as the original LinkedList and you have been provided with these requirements.

As an extra incitement for the testers, your project manager has decided to count the number of bugs you find and compare these to your fellow testers to make it a little bit of a competition, to see who finds the most bugs in MyLinkedList! So, lets see if you can find more bugs than your fellow testers. Remember that your project manager wants to see good test cases that covers a lot of the code in MyLinkedList so keep the test coverage percentage in mind as well!

Shall we get started? =)

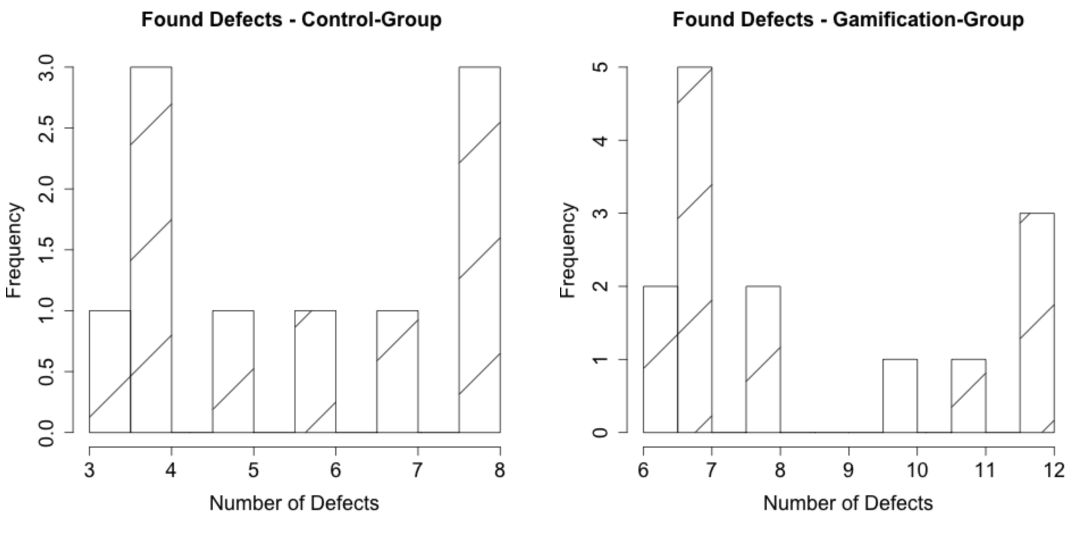
1. The scenario made the experiment more interesting.

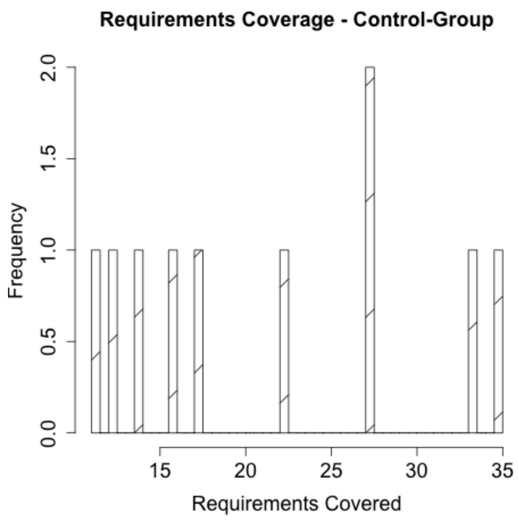
2. The scenario made the experiment LESS interesting.
3. The scenario gave the experiment a purpose.
4. The purpose of the experiment decreased because of the scenario.
5. The scenario increased my motivation for the experiment.
6. The scenario made me LESS motivated for the experiment.

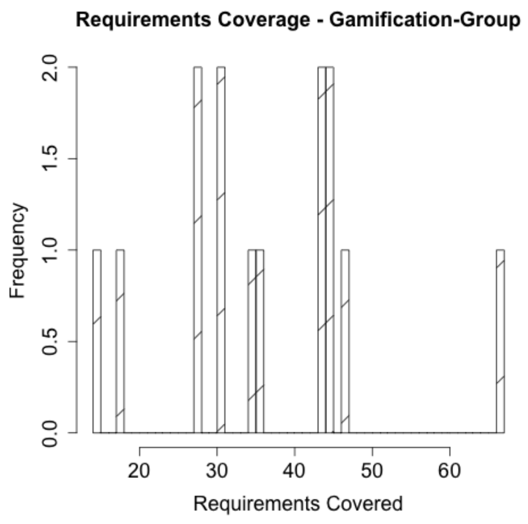
C

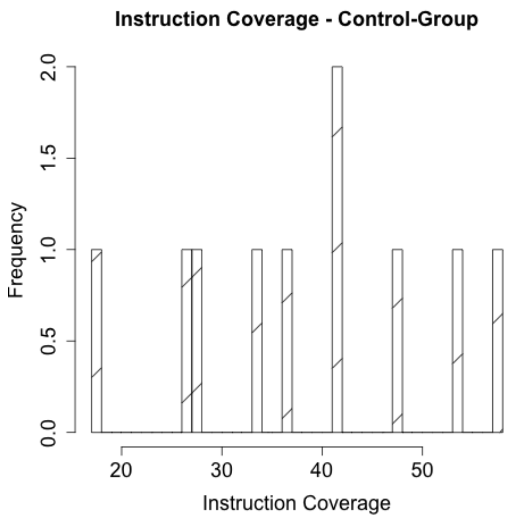
Histograms of Data From the Experiment

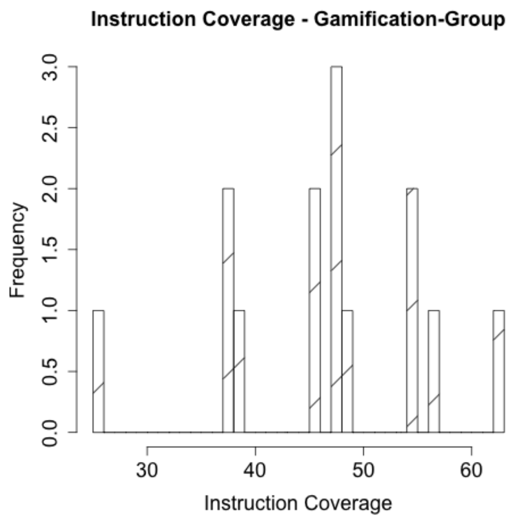
Figure C.1: Histograms for the data from the experiment.

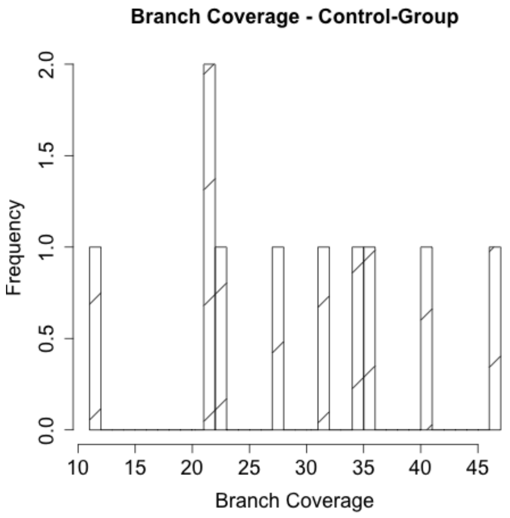


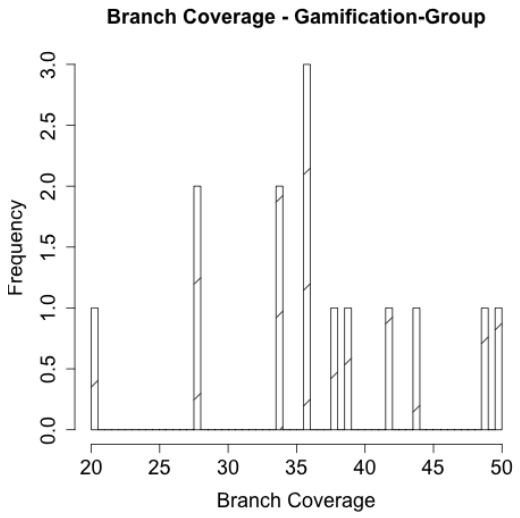

Requirements Coverage - Control-Group
Frequency
0.0
0.5
1.0
1.5
2.0
15
20
25
30
35
Requirements Covered

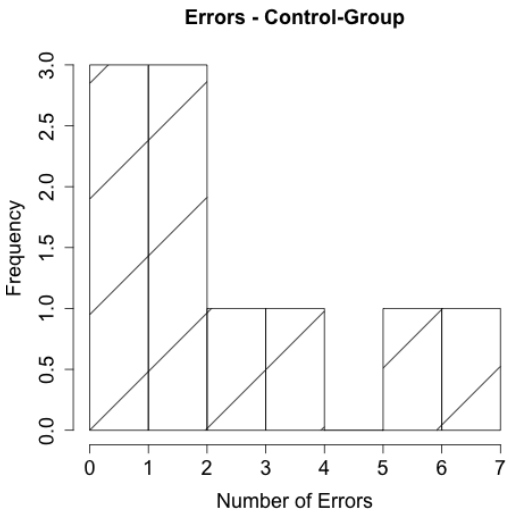

Requirements Coverage - Gamification-Group
Frequency
0.0
0.5
1.0
1.5
2.0
20
30
40
50
60
Requirements Covered

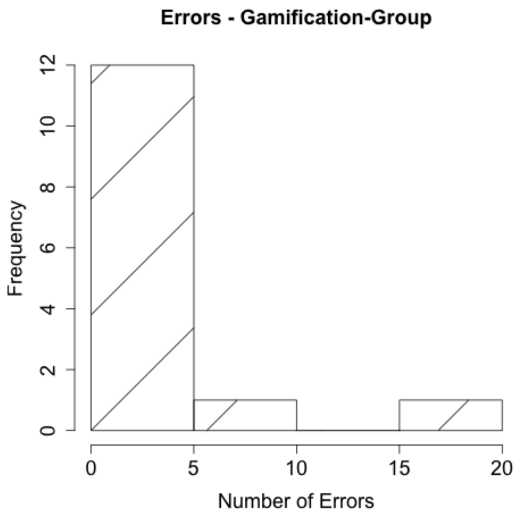

Instruction Coverage - Control-Group
Frequency
0.0
0.5
1.0
1.5
2.0
20
30
40
50
Instruction Coverage


Instruction Coverage - Gamification-Group
Frequency
0.0
0.5
1.0
1.5
2.0
2.5
3.0
30
40
50
60
Instruction Coverage


Branch Coverage - Control-Group
Frequency
Branch Coverage


Branch Coverage - Gamification-Group
Frequency
Branch Coverage


Errors - Control-Group
Frequency
Number of Errors


Errors - Gamification-Group
Frequency
Number of Errors