Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

TOWARD GAMIFICATION IN SOFTWARE ENGINEERING PRACTICE

WEI REN

Supervisor: STEPHEN BARRETT

SCHOOL OF COMPUTER SCIENCE AND STATISTICS

The thesis is submitted to Trinity College Dublin in fulfilment of the requirements for the degree of Doctor of Philosophy in computer science

March 2024

Declaration

I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and it is entirely my own work except where it explicitly acknowledges the unpublished and/or published work of others.

I agree to deposit this thesis in the University's open access institutional repository or allow the library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

I consent to the examiner retaining a copy of the thesis beyond the examining period, should they so wish (EU GDPR May 2018).

Publications from this Doctoral Thesis

- Ren, W., Barrett, S. (2019, June). Toward Gamification in Software Engineering, *The Irish Conference on Game Based Learning.*

- Ren, W., Barrett, S., & Das, S. (2020, January). Toward gamification to software engineering and contribution of software engineer. In Proceedings of *the 2020 4th International Conference on Management Engineering, Software Engineering and Service Sciences* (pp. 1-5). https://doi.org/10.1145/3380625.3380628

- Ren, W., & Barrett, S. Test-driven development, engagement in activity, and maintainability: An empirical study. IET Software. https://doi.org/10.1049/sfw2.12135

- Ren, W. (2023, December) Gamification in Test-Driven Development Practice. In Proceedings of *the 31st ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE) Workshop: Gamify.* https://doi.org/10.1145/3617553.3617889

- Ren, W., Barrett, S. An Empirical Investigation on the Benefits of Gamification in Communication within Development Teams, *Computer Applications in Engineering Education*. https://doi.org/10.1002/cae.22675

Table of Contents

List of Tables

List of Figures

Abstract

This thesis seeks to expand the explanation and examine the application of gamification strategies in software engineering practice, specifically in the education setting. I hypothesize that gamification strategies, deployed in the situated learning experience, can help students develop and maintain professional practice more effectively.

To verify my hypothesis, first, I identify effective strategies for applying gamification to software engineering practice accompanied by a theoretical framework through desk analysis, and through an empirical study to examine whether gamification is feasible for software engineering practice. Then, we conducted a series of experiments to examine the effectiveness of gamification in software engineering practice. This work focuses on one representative software engineering practice: test-driven development practice (TDD), which are hard to develop and maintain for students and novice developers.

Test-driven development (TDD), which has received considerable attention in recent years, is an example of key software development practice, and past literature suggests that TDD is strongly associated with high-performing engineering practices. First, I conducted an observational study to show that TDD can be applied gamification. Then, I have experimented with treatment and control groups to show that I can improve students' TDD practice efficiency using gamification, and distinguish the impact of different gamification strategies. Furthermore, I have developed evidence that gamification effect retain after intervention ceasing.

Our research argues that gamification is a valuable tool for promoting the development and maintenance of software engineering practices among students.

Acknowledgments

I would like to express my sincere gratitude to everyone who has contributed to the completion of this Ph.D. thesis.

First and foremost, I would like to thank my supervisor, Professor Stephen Barrett, for his guidance and support throughout my research. Despite the challenges I faced during my research work, Professor Barrett was always available to provide me with constructive feedback and advice, which has been instrumental in shaping my research work. I am grateful for his continuous encouragement and patience.

I would also like to thank my confirmation committees, Professor David Gregg and Professor Stefan Weber, and internal and external examiners, Professor Ann Devitt and Dr. Luca Longo, for their insightful feedback, valuable suggestions, and helpful comments. Their guidance has been crucial to the development of my research, and I am thankful for their time and effort. I would like to extend my thanks to the faculty members and administrative staff at Trinity for their support throughout my program.

I am also grateful to my partner, Dr. Yang Zhao, for her constant support and encouragement during my academic journey. She has been my biggest cheerleader, helping me with both academic and my life. Her unwavering support has been invaluable, and I could not have done this without her.

Finally, I want to thank my parents for their unconditional love and support throughout my academic career. They have always been my pillars of strength, motivating me to strive for excellence and never giving up on my dreams.

I am honored to have had the support of so many individuals throughout my doctoral program, and I thank you all from the bottom of my heart.

Chapter 1 Introduction

This thesis aims to expand the explanation and examine the application of gamification strategies in software engineering practice, specifically, in education settings. Gamification is a technique that incorporates game design elements into non-gaming contexts to enhance user performance. I propose that by using gamification strategies into learning experiences, students can develop and maintain professional practices more effectively. In this thesis, I focus on a notable practice in the field of software engineering: test-driven development (TDD). TDD has gained widespread attention for its association with high-performing engineering practices, but its implementation and maintenance can present difficulties for inexperienced developers and students. By examining the impact of gamification on these practices, I seek to shed light on its potential as a tool for enhancing software engineering education and professional development.

Chapter 2 provides an overview and a brief introduction to the relevant literature and hypothesis development. Chapter 3 introduces a theoretical framework for applying gamification to software engineering practices. Chapter 4 introduces the ethic preparation and studies setting. Chapter 5 introduces the methodology, including variables' definition and gamification design. Chapter 6 shows the results of both observational study and experimental studies. Chapter 7 discusses of each study and overarching threat to validity. Finally, this thesis concludes by summarizing the key contribution and limitation in chapter 8.

1.1 Overview and Motivation

The financial and societal impact of the Irish software industry cannot be understated, with a market research report revealing that the cost of software development activities accounted for approximately 31% of the industry's 64.4 billion Euro revenue in 2023.[1] As a result, any improvement in software engineering practices could lead to significant cost savings. For instance, even a 1% increase in efficiency could yield annual savings of around 200 million Euros. Given this context, exploring ways to improve software quality is of paramount importance, making the investigation of gamification as one of the means a crucial area of inquiry. By leveraging gamification, software engineering students and professionals may be better equipped to adopt and maintain practices, leading to improved software quality and potentially substantial financial benefits for the industry.

Software engineering practices include but not limited to test-driven development (TDD) (Sommerville, 2015; Voas and Agresti, 2004). Prior research suggests that high-quality TDD practice is related with improved software quality, particularly in terms of maintainability (see §2.3). Key TDD behaviors, such as writing test sequences, iterative development, and unit testing, as well as sustained engagement in development activities are crucial to achieving these benefits (see §2). Despite the potential advantages of using TDD, novice developers and students may face challenges when attempting to adopt and maintain this approach, resulting in lower TDD usage rates among these groups (see §2.5). Therefore, I seek to examine that the relationship between gamification and TDD behavior in this thesis.

Gamification, which involves the use of game design elements in non-gaming contexts, has been recognized as a valuable approach to optimize activities and improve engagement in software engineering practices (Deterding et al., 2011). While the primary benefits of gamification include behavior change and enhanced engagement recent research has also highlighted its potential to

[1] The IBISWorld Software Development in Ireland - Market Research Report, 2023 edition: https://www.ibisworld.com/ireland/industry-statistics/software-development/3595/

promote a positive learning experience and increase motivation. Moreover, the interest in gamification has grown significantly since 2011, with the software engineering education community showing particular interest (see §2.4).

However, gamification is no silver bullet: inappropriate gamification design can often result in decreased gamification effectiveness, which is one of the most frequently observed negative consequences. For example, the existing designs, such as Octalysis, CEGE, and 6D, are mostly designed for a more general subject and not tailored to software engineering practice (see §2.5). Due to the lack of gamification design for software engineering practice, the first research question is put forward:

RQ1: How to apply gamification on software engineering practice?

Therefore, Chapter 3 presents a gamification framework that is specifically designed for software engineering practice. This framework provides practical guidance for software engineers and developers to incorporate gamification into their practices in a meaningful and effective way.

Furthermore, unclear goals of applying gamification can also have negative consequences, such as decreased motivation and unreasonable activity levels (Moldon et al., 2021). Therefore, in order to clarify whether gamification is feasible on TDD practice, with the goal of improving the efficiency of software engineering practices (in this thesis, I select TDD as a representative practice), and given that the main benefits of gamification are behavior change and increased engagement, so this thesis examines the relationship between the key components of efficient TDD (TDD behavior, engagement in development activities) and maintainability. The second research question raised:

RQ2: Is gamification feasible on TDD practice?

Test-driven development (TDD) is also named as test-first programming, is the practice of writing automatic test cases before production code (Beck, 2003). Production code is a block of code used to satisfy a specified requirement, whereas test code is used to test the corresponding production code.

The adoption of TDD practice is claimed to have an impact on software quality (Bissi et al., 2016), including both internal and external quality (B. W. Boehm et al., 1976). While scholar argue that TDD has only a marginal positive effect on external quality (Rafique and Mišić, 2012), others have found a significant increase in internal quality (Bissi et al., 2016), particularly the improved maintainability (Fucci and Turhan, 2014; Munir et al., 2014; Tosun et al., 2018; Williams et al., 2003). Maintainability is considered one of the fundamental characteristics of internal quality(Botella et al., 2004), and it has been a topic of interest for decades. Increase maintainability can lead to a reduction in support costs (Dhillon, 2006).

Previous studies indicate that the benefits of TDD practice stem from its unique characteristics, such as test-first approach (Beck, 2003; Bissi et al., 2016; Tosun et al., 2018; Tosun et al., 2017). However, recent studies point out that the positive impact of TDD might not be solely due to its test-first attributes. Instead, it might be attributed to the increased focus on testing by putting more effort into testing (Fucci et al., 2016; Fucci et al., 2015), and improved coding by encouraging developers to follow fine-grained coding practices (Fucci et al., 2016). In addition, researchers have found that developers who follow TDD are more likely to be engaged in development activities such as coding and testing (Erdogmus et al., 2005; Tosun et al., 2017). This therefore leads to an interesting question:

RQ2a: does following TDD behaviors improve developers' engagement in development activities, such as coding and testing?

The term "engagement" has been defined by a psychologist Schaufeli as focus, involvement, and passion (Wilmar B. Schaufeli, 2013b). Given that prior research suggests that engagement can improve working efficiency and lead to better working outcomes (Markos and Sridevi, 2010), it is natural to question:

RQ2b: whether a higher engagement level in development activities lead to superior software quality?

When it became clear that gamification is feasible on TDD, the next step was to examine the effectiveness of gamification on TDD through experiments. The lack of sufficient research on the

effectiveness of gamification in software engineering practice hinders our ability to fully comprehend its impact on software engineering practices (Monteiro et al., 2021). Current research on gamification in software engineering has primarily focused on project management (Machuca-Villegas and Gasca-Hurtado, 2018) and requirements (Pedreira et al., 2015), with limited attention given to development practices (Manal M Alhammad and Moreno, 2020).

Also, the existing research on gamification effectiveness has primarily focused on simple activities, such as improving documentation quality or naming variables properly, while there is a lack of understanding regarding its potential impact on more complex practices like TDD (see §2.2.3). In addition, the recent studies call for empirical evidence on the effectiveness of gamification strategies (Manal M. Alhammad and Moreno, 2018; Monteiro et al., 2021; Paula Porto et al., 2021).

Taken together, there is limited research on the application of gamification in complex daily development practices, and a lack of empirical evidence. Unlike previous studies that rely on questionnaires, my study aims to examine the relationship between gamification and TDD practices quantitatively using statistical methods. As a result, this thesis seeks to examine the effectiveness of gamification on TDD practice, and identify relevant gamification strategies that can help optimize software engineering processes. And the third research question raised:

RQ3: Can gamification strategies improve TDD efficiency in both group and individual setting?

To provide a more in-depth understanding of gamification, unlike previous research that has mainly focused on the combined effects of gamification strategies without exploring the impact of individual strategy (§2.5), thesis thoroughly investigates and compare the impact of various gamification strategies on software engineering practices. Based on prior research, there exist certain challenges and limitations in the implementation of gamification strategies in software development practices. Points, leaderboards, and feedback are commonly used gamification techniques (Çeker and Özdaml, 2017; Koivisto and Hamari, 2019; Ren et al., 2020). However, these strategies are often analyzed in combination, making it difficult to determine the impact of each individual strategy (Manal M. Alhammad and Moreno, 2018; Denny et al., 2018; Seaborn and Fels, 2015).

RQ4: Does using gamification strategies together better than using individually?

Furthermore, research on the sustainability of gamification effects in software engineering practice is an aspect that has not been fully explored, although other fields have examined the durability of gamification effects after the intervention has been removed (Manal M. Alhammad and Moreno, 2018; Pedreira et al., 2015).

RQ5: Does the gamification effect on TDD persist after withdrawing the intervention?

In summary, the objective of my thesis is to expand the explanation and examine the application of gamification in software engineering practice, specifically in educational setting. The overall structure is shown below, which includes four works.

To achieve the objective, first of all, this thesis has done two preliminary works for preparation, gamification framework and observational study respectively. The gamification framework is a desk analysis, which addresses the question, how to apply gamification on SEP? The current gamification framework is mostly for general contexts, such as education, business, management, etc. rather than specifically for SEP. Given the framework introduced, observational study examines whether it is feasible for SEP? In here, this work focused on test driven development practice (TDD). Then, I conducted two experimental studies that focused on applying gamification on TDD practice in both group setting and individual setting.

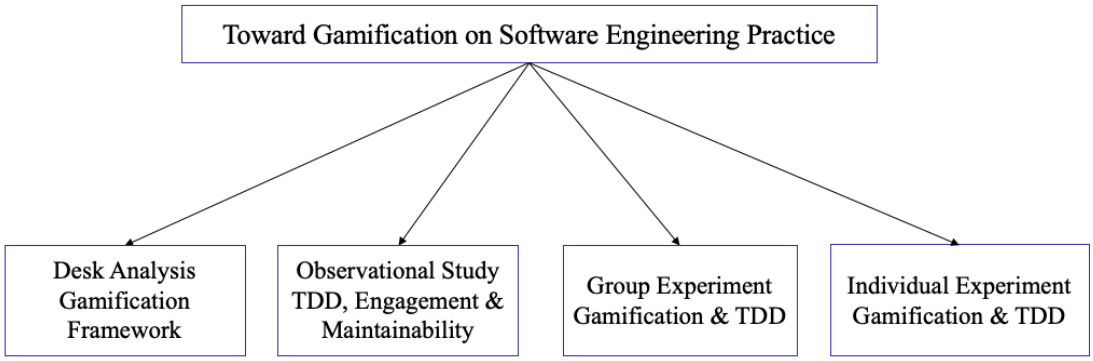


Figure 1.1 Overall Introduction

1.2 Summary of Research Questions in Bullet Points

Research Questions
RQ1: How to apply gamification in software engineering practice?
RQ2: Is gamification feasible on TDD practice?
RQ2a: Does following TDD behaviors improve engagement in development activities? RQ2b: Does higher engagement level lead to a better maintainability?
RQ3: Can gamification strategies improve TDD efficiency in both group and individual setting?
RQ3a: Does gamification promote the following of TDD style? RQ3b: Does gamification improve engagement level in development activities? RQ3c: Does gamification improve maintainability?
RQ4: Does using gamification strategies together better than using individually?
RQ5: Does the gamification effect on TDD persist after withdrawing the intervention?

1.3 Research Design

Following the principles of empirical software engineering research , I use observational study and experimental approach in this thesis, which involves introducing an intervention and observing its effects while controlling for certain factors, making it the most effective method for testing hypotheses (Singleton et al., 1999; Wohlin et al., 2012).

To answer the RQ2, using a data-driven method, I investigated the relationship between Test-Driven Development (TDD), engagement in development activities, and maintainability among 237 third year undergraduate students at a university. Ordinary least squares (OLS) regressions were employed, and the results supported my hypothesis.

To answer RQ3 to RQ5, I conducted two experiments.

The group experiment was carried out with two development teams, consisting of 6 and 20 students respectively, enrolled in the same software development module (Software Design and Implementation) from academic years 2020 and 2021. The teams were tasked with developing a complex Android application, and the experimental period was 45 days. The study followed a Pretest-Posttest Control Group Design. To assess the efficacy of gamification, an empirical analysis was conducted using graphical representations and ordinary least squares (OLS) regression. This was done in order to gain a deeper understanding of the impact of gamification on the development process.

The individual experiment was conducted over the course of a month with 162 final-year undergraduate software engineering students, with the aim of gathering empirical evidence to address the research questions posed in this study. The experiment was conducted within the context of a large third-year software engineering module. The students are from academic years 2021 and 2022, and the experiment lasted for 44 days per academic year.

Table 1.1 presents a brief overview of the methodology and the data analysis employed in this thesis.

Research Method	Data Analysis Method	Data Sources
Observational Study	OLS regression; Bivariate Analysis	

Group Experiment	Graph analytics; Bivariate Analysis; OLS regression	Github
Individual Experiment	Graph analytics; OLS regression	

Table 1.1: Overview of methodology employed

1.4 Key Findings

First, this thesis presents a comprehensive framework for the implementation of gamification design, which makes gamification strategies actionable on software engineering practice. It also introduces a gamification configurable method to assist researchers in selecting appropriate gamification elements based on the intended outcome.

Second, the thesis provides empirical results that gamification is feasible on TDD practice. The two main areas where gamification has an impact, behavior and engagement, are critically related to the effectiveness of TDD practices. More specifically, this thesis finds that following TDD behaviors increase engagement levels in development activities, and engagement level has significant and positive performance consequences for software maintainability. Specifically, the positive impact on maintainability is especially pronounced for engagement in the testing phase versus the coding phase. Additionally, engaging in creating new test cases offers greater benefits on maintainability compared to engaging in maintaining existing test cases. My results remain robust when controlling for various factors such as readability, different tasks, size, and programming language.

Third, this work shows that the implementation of gamification produces favourable results for TDD practices in both group and individual setting. Specifically, the utilization of gamification strategies motivates team members to adhere to TDD behaviors, leading to heightened engagement in development activities, and improving software maintainability.

Fourth, this thesis further analyzes the effects of different gamification strategies on TDD efficiency. It finds that combining multiple gamification strategies is more effective than using them individually. Last, the impact of gamification remains noticeable even after the intervention is removed, indicating lasting efficacy.

Overall, the thesis highlights the potential of gamification as a valuable and practical tool in software engineering, improving efficiency and quality of software development practice, specifically TDD.

Chapter 2: State of the Art

The primary objective of an effective software engineering is to develop and maintain high-performing professional practices (B. W. Boehm, 1976; Sommerville, 2015; Voas and Agresti, 2004), which requires addressing technical challenges (Lenberg et al., 2015; Matturro et al., 2019; Weinberg, 1971). Navigating technical challenges involves overcoming the difficulty of achieving high-quality software development, a process deeply rooted in practical activities within the field of software engineering (Voas and Agresti, 2004). Test-driven development (TDD) as a prime example, represents a crucial and fundamental aspect of the development process (Beck, 2000; Nanthaamornphong and Carver, 2017; Roman and Mnich, 2021; Treude and Storey, 2010; Velmourougan et al., 2014; Zielinski and Szmuc, 2005). Undoubtedly, the significance of TDD cannot be overstated, given its critical role in improving software quality (Papis et al., 2020). By exploring this fundamental aspect of engineering practices, this study seeks to contribute to the current body of knowledge in software engineering and deepen understanding regarding the factors, behaviors and engagement, that form the basis for the creation of high-quality software. Then, this Chapter provides a comprehensive literature review on gamification strategies, including an overview of gamification concepts, its advantages, and its application in software engineering. This Chapter also discusses the existing gaps in the literature on TDD practice, and current application of gamification strategies in the field of software engineering.

2.1 Test-Driven Development Behaviors

In the initial phases of software development, researchers endeavored to systematize the "test-design thinking" development method by adopting various techniques and tools (Bertolino, 2007). An illustration of this is the introduction of the iterative development method based on test-design thinking during NASA's Mercury project in the early 1960s (B. W. Boehm, 1976). Markedly, in 2003, Beck introduced test-driven development (TDD), also known as test-first programming, as a representative software development practice to improve software quality (Beck, 2003; C. Chen et al., 2017; Fucci and Turhan, 2014; Tosun et al., 2018; Tosun et al., 2017).

Since its inception, TDD has gained considerable attention in both academia and software development industry, establishing itself as one of the most widely used agile practices (Bissi et al., 2016; A. Santos et al., 2021; Tosun et al., 2018). In Beck's seminal work, TDD is defined as "write new product code only when a test case fails and refactor product code whenever you see fit" (Beck, 2003), visually depicted in Figure 2-1. A key characteristics of TDD is its emphasis on small and rapid iterations, with each unit designed to be as small as possible and compiled by testable software components (Beck, 2003). Specifically, TDD operates as a cyclic development technique, with each cycle ideally coinciding with the implementation of a small feature (Fucci et al., 2016).

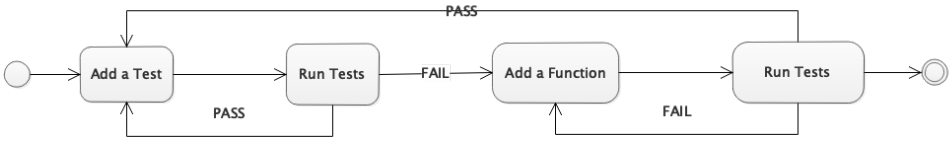


Figure 2-1 Test-driven development

The existing literature indicates a strong correlation between the adoption of the Test-Driven Development (TDD) practice and significant improvements in software quality. Specifically, 76% of prior studies report an enhancement in internal software quality, while 88% of them observe a substantial increase in external software quality (Bissi et al., 2016), with a particular emphasis on maintainability (Dogša and Batič, 2011; Khanam and Ahsan, 2017; Mäkinen and Münch, 2014; Tosun et al., 2018). In addition, TDD's impact on code quality surpasses that of traditional

development approaches, resulting in a twofold improvement, coupled with a notable reduction of approximately 40% in code defects (Bissi et al., 2016; Williams et al., 2003). TDD practice has also been found to promote productivity in academic environments, measured as the line of source code per hour (Bissi et al., 2016).

The underlying mechanisms propelling the benefits of TDD practice can be traced to two key behaviors: unit testing and fast iteration (Beck, 2003; Bissi et al., 2016; Borle et al., 2018; Crispin, 2006; Fucci et al., 2016; A. Santos et al., 2021; Tosun et al., 2018; Tosun et al., 2017),. Concurrently, studies suggest that increasing engagement levels during development activities may further amplify the advantages of TDD practice (Buchan et al., 2011; Fucci et al., 2016; Papis et al., 2020). Empirical evidence indicates that TDD contributes to increased developer engagement and motivation, fostering a greater sense of accomplishment and sustained focus throughout the development process (Buchan et al., 2011; Fucci et al., 2016). These findings underscore the potential for actively fostering engagement levels within TDD practices to enhance its effectiveness in improving software quality and development outcomes. More details on the impact of engagement level on TDD benefits are discussed in the following section.

2.1.1 Unit Testing

Unit testing stands as a critical component within Test-Driven Development (TDD) and the Extreme Programming (XP) methodology. It involves testing the functional requirements and properties of single units within the system, such as classes or methods (Runeson, 2006).

Within the expansive realm of software testing, my focus in this study is specifically on unit testing, a practice found to be effective in workplace settings. For example, research conducted at Microsoft revealed a 21% reduction in the number of faults discovered by software quality assurance and customer-reported problems through the implementation of unit testing. However, it is important to acknowledge concerns raised by researchers about the potential impact of unit testing on productivity, as it may result in a 30% increase in development time (Shull et al., 2002; Williams et al., 2009).

Despite the productivity concerns, unit testing retains its status as a critical component of TDD and XP practices, leading to the improved software quality and reliability. One advantage of unit testing is its ability to pinpoint issues in the codebase at an early stage, preventing them from evolving into significant bugs in the software. This early detection not only facilitates timely issue resolution but also leads to significant cost savings, as fixing a defect post-release is often more expensive than resolving it during development (B. W. Boehm, 1984; Shull et al., 2002).

Moreover, unit testing is believed to increase engagement and motivation among developers. It provides swift feedback, enhancing the sense of accomplishment and maintaining focus throughout the development process (Buchan et al., 2011; Fucci et al., 2016; Papis et al., 2020).

In conclusion, despite concerns regarding its potential impact on productivity, unit testing remains a critical component of TDD and XP practices, offering improvements in software quality and reliability. Additionally, unit testing can also increase engagement and motivation among developers, thereby potentially enhancing the overall benefits of TDD practice.

2.1.2 Fast Iteration

Test Driven Development (TDD) is a software development process that emphasizes the importance of rapid iterations. In TDD, the process involves initially writing a failing test, followed by writing the minimal code necessary to pass the test (Beck, 2003). This approach encourages developers to write code in quick cycles, aiming for frequent, small improvements to the software.

The rapid iteration cycle provides several advantages. Firstly, it is instrumental in facilitating early defect detection and ensuring code quality. Second, the ability of rapid iterations to focus on a specific aspect of the software (such as a single method or class), combined with an increased number of development cycles, can help developers to better understand the structure of the code (Fucci et al., 2016; Tosun et al., 2018). This aligns with TDD's positive impact on both internal and external software quality (Bertolino, 2007; Chávez et al., 2017; Fucci et al., 2016; Tosun et al., 2017).

2.2 Engagement level in Development Activities

2.2.1 Background of Engagement

Engagement is a psychological construct that refers to the extent to which individuals are fully involved and enthusiastic about their work (Wilmar B. Schaufeli, 2013a). According to (Bakker and Demerouti, 2008), engagement has been found to positively impact work performance, such as creativity, productivity, and willingness to undertake additional tasks. In the software engineering domain, engagement was initially introduced as a means of improving communication and teamwork among software development teams (Crowston and Howison, 2005; Ehrlich and Cataldo, 2012; Shihab et al., 2009).

Previous studies have suggested that engagement plays a crucial role in enhancing work performance and productivity (Markos and Sridevi, 2010). In the field of software engineering, there have been a few studies that have examined the impact of engagement on team performance, with a particular focus on communication among software development teams (Crowston and Howison, 2005; Ehrlich and Cataldo, 2012; Shihab et al., 2009). More recently, researchers have begun to explore engagement in other aspects of software engineering, such as team collaboration (Akpolat and Slany, 2014), requirement documents (Tizard et al., 2022), and software architecture (Keuler et al., 2012).

Despite previous research efforts, the role of engagement in development activities, such as coding and testing, remains a significant gap in the literature. (J.-C. Chen and Huang, 2009; Fucci et al., 2016; Myers et al., 2011; Rosen et al., 2015). While engagement has been found to positively impact work performance and productivity in software engineering, there has been limited investigation into the impact of engagement on development activities. This represents an important area for future research, as engagement in these activities can significantly affect the quality and efficiency of software development. Furthermore, disengagement in development activities has been shown to have a negative impact on individual performance (Qiu et al., 2019).

With respect to Test-Driven Development (TDD), previous research has reported that the benefits of TDD are not solely attributable to the rapid iteration process, but also to the increased focus on the development activity (Fucci et al., 2016). Tosun et al. (2018) further demonstrated that the benefits of TDD practice are driven by developers being more likely to participate and engage in testing activities. Higher engagement in the development activity has been identified as a factor that leads to high-performing TDD practice (Beck, 2000; Bissi et al., 2016).

2.2.2 Measurement of Engagement

Measuring engagement poses a considerable challenge in this study due to its inherent lack of direct observability. To address this issue, psychologists have proposed using a set of questionnaires, Utrecht Work Engagement Scale (UWES), to measure engagement in working environments (Breevaart et al., 2012; W. B. Schaufeli and Bakker, 2003).

In recent years, researchers in computer science education have attempted to quantitatively measure engagement in the classroom from three perspectives: behavioral, cognitive, and emotional engagement (Fredricks et al., 2004).[3] Among these, behavioral engagement is considered easier to capture, given that behavioral patterns can be defined, observed, and interpreted (Liu et al., 2014). The measurement of engagement is crucial in educational research as behavioral engagement plays a crucial role in students' participation in learning and leads to positive academic outcomes (Fredricks et al., 2004; Qahri-Saremi and Turel, 2016).

Various attempts have been made to measure engagement through behavioral engagement, including quantifying completed activities and time allocation to tasks (Riemer and Schrader, 2016, Orji et al. 2021). Engagement metrics derived from completed activities encompass factors such as the number of video views and bullet chats (He et al., 2022), the number of clicks on links to learning resources (Namara et al., 2022; Orji et al., 2021) and the number of participation tasks completed. Similarly, engagement metrics derived from time allocation include investigating code review time

[3] Behavioral engagement refers to participants' activities, cognitive engagement refers to students' affective reactions in the classroom, and emotional engagement refers to a desire to go beyond requirements.

length (Rauf et al. 2022, Lane and Harris 2015), measuring the duration of time spent at school (Qahri-Saremi and Turel, 2016), and monitoring time spent on open-source GitHub projects (Qiu et al., 2019). Please see section 5.2 for a discussion of the measures of engagement used in this study.

2.3 Software Quality

Software quality is a critical aspect that reflects the effectiveness of Test-Driven Development (TDD) practice. The definition of software quality has been a subject of study for several decades, with various researchers emphasizing different aspects. For instance, B. W. Boehm et al. (1976) defined software quality as including portability, as-is utility, and maintainability. In recent years, the International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) updated the definition of software quality to include functionality, reliability, usability, efficiency, probability, and maintainability, as per ISO/IEC 25010:2011 (Estdale and Georgiadou, 2018).

The attribute of maintainability is of utmost importance in software quality and has gained a lot of attention in recent years. Maintainability refers to the ease with which a software system can be modified, and it is one of the fundamental aspects of software quality (Alsolai and Roper, 2020). In this regard, software maintainability is linked to the maintenance process, which constitutes the majority of the cost of the software development lifecycle (Zelkowitz, 1978). Research has indicated that highly maintainable software could reduce approximately 75% of the cost in most systems' life cycles (C. Chen et al., 2017). Moreover, during the development life cycle, 60% to 70% of the consumption is for the maintenance of the software, including resources, time, money, and effort consumption (Razina and Janzen, 2007). Consequently, it is vital to improve software maintainability to effectively manage the cost.

Maintainability is also a key aspect that reflects the performance of TDD practice, as indicated by several previous studies (Bissi et al., 2016; Borle et al., 2018; Buchan et al., 2011; C. Chen et al., 2017; Razina and Janzen, 2007; A. Santos et al., 2021; Tosun et al., 2018; Williams et al., 2003). Therefore, maintainability is used in this thesis to represent the performance of TDD practice.

2.4 Gamification

Providing effective training for future software engineers is a crucial challenge in software engineering education. To tackle this challenge, researchers have devoted their efforts towards improving software engineering teaching methods and techniques. This improvement involves enhancing the learning process and finding the most efficient approach for providing software engineering students with the necessary knowledge and skills. To achieve this, the software engineering community has examined various innovative pedagogical strategies. Of these strategies, gamification stands out as the most representative, as indicated by the extensive research conducted by Connolly et al. (2007); Thomas and Berkling (2013).

2.4.1 Background and Definition

Gamification, initially introduced as a digital marketing strategy, emerged in the business and marketing domain in 2008 as a means to enhance customer engagement. However, it did not gain widespread popularity until the mid-2010s (Deterding et al., 2011). Over time, gamification has been successfully extended and adopted in various domains, including health, business, and education due to its efficacy in enhancing motivation and engagement (Barata et al., 2014; Kapp, 2012; Latulipe et al., 2015; Monterrat et al., 2014; Muntean, 2011).

In recent years, gamification has emerged as a highly significant and widely utilized teaching technology in education (Johnson et al., 2014). The academic community's interest in gamification has grown steadily, with a significant amount of literature published on the topic, providing educational solutions for the new generation of students (Bíró, 2014).

Gamification has now been applied in the context of software engineering, making significant contributions to this field. The definition of gamification, as proposed by Deterding et al. (2011), in the area of computer science, is the use of game design elements in non-game contexts. In other words, gamification utilizes game elements and mechanics to alter behavior and enhance people's motivation and engagement in their tasks (Deterding et al., 2011; Garcia et al., 2017).

The increasing interest in gamification among the academic community has led to a wealth of literature exploring its potential uses and benefits. In the context of software engineering, gamification has opened up new opportunities to improve the learning experience and training of future software engineers.

2.4.2 Gamification Advantage

Rodrigues et al. (2018) conducted a systematic mapping study on gamification, which concludes that gamification is a growing topic since 2011 and that the technique is a new trend used to engage students in software engineering education (Ouhbi and Pombo, 2020). Monteiro et al. (2021) argue that gamification is an interesting technique to provide a positive impact on software engineering, including changing people's behavior, improving engagement (Paula Porto et al., 2021), collaboration (Barreto and França, 2021), and participation (Monteiro et al., 2021) in industrial contexts.

Furthermore, the application of gamification in software engineering goes beyond motivation and involvement. Dubois and Tamburrelli (2013) argued that gamification applied in software development practice has several advantages due to its mechanisms. It is expected to improve the results in software engineering tasks, both in terms of product quality and project performance (García et al., 2017), such as encouraging simple good programming practices (Singer and Schneider, 2012), identification and fault removal (Fraser, 2017), and improving the performance of processes (Dorling and McCaffery, 2012).

Recent reviews of studies in gamification show positive effects on behavioral outcomes (Ilhan et al., 2022; Saleem et al., 2021). For example, Prause et al.'s investigation into the impact of adding a point system to coding conventions is an early example of work in gamification and behaviors (Prause and Jarke, 2015). They found that introducing gamification interventions can effectively improve adherence to coding conventions for students.

More recent studies have shown that engagement in class can increase significantly following the introduction of common game elements such as points and leaderboards (de Almeida Souza et al.,

2017; Hsieh and Yang, 2020; Ivanova et al., 2019). Some studies investigate the sustainability of gamification in the education area (Xiuhan Li and Chu, 2021; Sanchez et al., 2020; Suh et al., 2017; Tahmasbi and Fuchsberger, 2018). Therefore, in software engineering, researchers are aware of the potential benefits of gamification, such as gamification rewards developers for their activities and makes work more enjoyable (Garcia et al., 2017). Thus, unpleasant tasks such as writing unit tests and performing maintenance may have a positive impact on the development team because of gamification mechanisms.

2.5 Hypotheses Development

There is a relatively small body of literature on the use of gamification in software engineering (Monteiro et al., 2021). Despite its growing popularity, the theoretical framework for gamification in software engineering remains largely limited (Kasurinen and Knutas, 2018; Monteiro et al., 2021). The previous literature review identified the lack of a sound methodological approach for the application of gamification, which hinders the replicability of gamification proposals in different organizations or scenarios (García et al., 2017). The current gamification frameworks, such as Octalysis (Chou, 2019), GOAL (García et al., 2017), social gamification framework (Simões et al., 2013), CEGE (Zichermann and Cunningham, 2011), 6D (Werbach and Hunter, 2020), are not specifically designed for software development, especially for day-to-day development. Additionally, inappropriate gamification design has been shown to result in performance loss among software engineers (Toda et al., 2018). Therefore, the first research question is raised:

RQ1: *How to apply gamification in software engineering practice?*

In addition, the current state of research on is still insufficient to show that gamification is feasible for Test-Driven Development (TDD) practice. So, the first research question is naturally proposed:

RQ2: *Is gamification feasible on TDD practice?*

As gamification main benefits are changing behaviors and increasing engagement level, it is necessary to examine whether behavior and engagement are related to the efficiency of TDD. In a recent paper, Tosun et al. (2018) reported that adhering to one of the key attributes of TDD, testing first, leads to a higher efficiency in unit tests. However, previous work argues that this improvement might be due to a higher level of engagement in testing and coding, rather than to the test-first approach itself (Fucci et al., 2016; Fucci et al., 2015). Furthermore, previous study claims that students who follow TDD exhibit better task focus (Erdogmus et al., 2005). As such, my first sub research question is generated,

RQ2a: *Does following TDD behaviors improve engagement in development activities?*

Engagement has been shown to improve working efficiency and yield better working outcome in general (Markos and Sridevi, 2010). However, there is limited study focuses on engagement level in development activities such as coding and testing. I therefore ask a question:

RQ2b: *Does higher engagement level lead to a better maintainability?*

Next, it needs to be examined whether it is necessary to apply gamification to TDD. In recent years, there have been some attempts to apply gamification to software engineering activities, but such work has primarily focused on very simple and lower-order activities. For example, studies have explored the use of gamification to encourage young students to name variables correctly (Prause and Jarke, 2015), shorten coding time (Tsunoda and Yumoto, 2018), improve software documentation (Sukale and Pfaff, 2014), and communication within web design (Hsieh and Yang, 2020). However, while gamification has been used to motivate complex behavior and higher-order activities in other areas, such as business, there has been a lack of research on the application of gamification techniques in students to develop and maintain complex professional software engineering practices, such as TDD (Trinidad et al., 2021). Moreover, the empirical evidence of the effectiveness of gamification in software engineering is largely limited (de Almeida Souza et al., 2017; Monteiro et al., 2021).

Furthermore, TDD poses significant challenges to effective implementation (Hammond and Umphress, 2012). While TDD has been successfully extended by high-level software engineers (Tosun et al., 2018), it is difficult for students and novice developers to adopt and sustain (Choma et al., 2018; Garousi et al., 2020; Hammond and Umphress, 2012; Muller and Tichy, 2001; Persson and Isberg, 2019; Rocha et al., 2021). For instance, previous researchers have investigated various agile methodologies, including TDD, in university courses and discovered that TDD was among the most challenging practices to adopt due to students' belief that writing test cases before coding was impractical (Muller and Tichy, 2001). Furthermore, students have difficulty adopting unit testing (Garousi et al., 2020) and changing development behaviors to adhere to TDD (Buffardi and Edwards, 2014; Mugridge, 2003; Persson and Isberg, 2019), among other difficulties. Therefore, the next question arises naturally:

RQ3: *Can gamification strategies improve TDD efficiency in both group and individual setting?*

The utilization of gamification has been shown to enhance the performance of individuals while performing a specific task (Pedreira et al., 2015). Literature suggests that this improvement can be attributed to the alteration of specific behaviors and engagement levels (Monteiro et al., 2021; Paula Porto et al., 2021). Previous studies have also posited that the benefits (software maintainability) derived from the practice of Test-driven Development (TDD) can be attributed to its unique behaviors and heightened engagement in development activities, such as coding and testing (Beck, 2003; C. Chen et al., 2017; Fucci et al., 2016; Madeyski and Biela, 2007; Tosun et al., 2018). Consequently, my objective is to encourage students to adopt and maintain TDD practices by altering their behaviors to align with TDD's unique behaviors and increasing their engagement in development activities. So, these sub research questions are proposed:

RQ3a: *Does gamification promote the following of TDD style?*

RQ3b: *Does gamification improve engagement level in development activities?*

RQ3c: *Does gamification improve maintainability?*

Several common limitations of gamification have been identified in the literature, and researchers are recommended to avoid these issues in future work (Hamari et al., 2014; Monteiro et al., 2021; Paula Porto et al., 2021; Seaborn and Fels, 2015). The first major theme in the literature highlights the need for more rigorous experimental designs to evaluate the effectiveness of gamification in software engineering. Specifically, researchers stress the importance of employing control groups and isolating the effects of individual gamification elements. Studies such as those conducted by Hamari et al. (2014) and Denny et al. (2018) have noted the low number of participants and short duration of many gamification experiments. It is worth noting that gamification research in software engineering education has often utilized quasi-experimental designs, rather than randomly selected groups of students receiving different interventions. As such, it is important for future studies to consider the methodological limitations of existing research and strive for more robust experimental designs to yield more reliable results.

Furthermore, while previous studies have examined the impact of commonly used gamification strategies, such as points, leaderboards, and feedback, in the software engineering domain, they have largely focused on the combined effects of these strategies rather than their individual contributions (Manal M. Alhammad and Moreno, 2018; Denny et al., 2018; Seaborn and Fels, 2015). Thus, further research is needed to investigate the specific effects of each gamification element on software engineering outcomes. Therefore, the fourth research question is raised:

RQ4: *Does using gamification strategies together better than using individually?*

Another important aspect of gamification that warrants further exploration is the sustainability of its effects. Previous studies have analyzed whether gamification effects persist after the removal of gamified elements in different areas, such as medical education, mobile application interaction, and civic engagement platforms (Hassan, 2017; Law et al., 2011; Pesare et al., 2016). However, research on the sustainability of gamification effects in software engineering practices remains limited (Kalogiannakis et al., 2021; Xiuhan Li and Chu, 2021; Sanchez et al., 2020; Suh et al., 2017; Tahmasbi and Fuchsberger, 2018). Further investigation in this area could help shed light on the long-term effectiveness of gamification strategies in software engineering and inform the development of sustainable gamification interventions. So, the fifth research question is:

RQ5: *Does the gamification effect on TDD persist after withdrawing the intervention?*

Chapter 3 Gamification Framework

Our objective is to integrate gamification techniques into educational software engineering curricula in order to encourage students to cultivate and sustain professional software engineering practices more efficiently. A significant challenge in incorporating gamification into software engineering practice is the impracticality of implementing explicit gamification strategies that integrate with daily processes. The responsibility of gamification researchers is to clarify the software engineering process, streamlining its presentation to make it executable. In this study, I provide a gamification model for the complex and contemporary software engineering process and synthesize the findings from prior studies to identify which gamification strategies can be implemented and their associated benefits.

3.1 Introduction and Background

The integration of gamification principles into the realm of educational software engineering has garnered significant attention in recent years as a means of improving software development outcomes (as demonstrated in Figure 3.1). Gamification is defined as the application of game design elements in non-gaming contexts (Deterding et al., 2011), and was first introduced in the field of computer science in 2011. Subsequent research has confirmed its effectiveness across various disciplines, including education, online communities, and business, leading to improved psychological and behavioral outcomes for users (Hamari et al., 2014). Given these benefits,

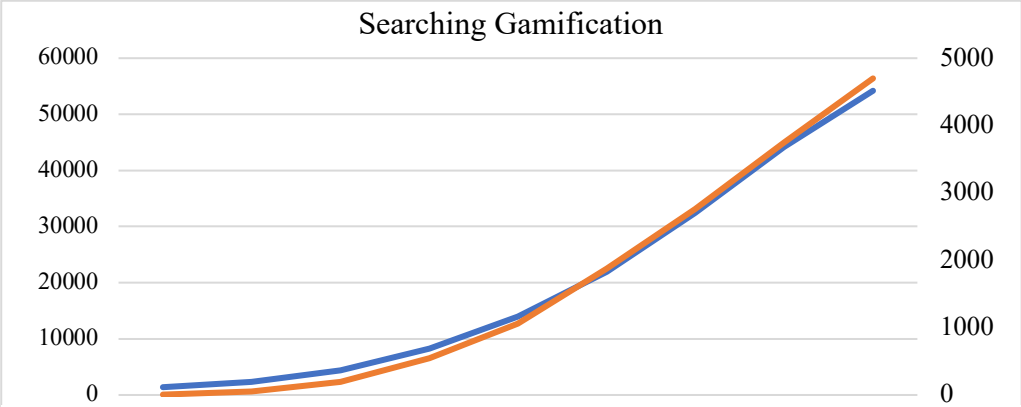


Figure 1.1 Searching Gamification

gamification has been proposed as a cutting-edge technique for enhancing the efficiency of professional practice in the software engineering community (Hamari et al., 2014; Pedreira et al., 2015).

The main objectives of gamification are to bolster users' motivation, engagement, involvement in activities, acquisition of knowledge, and adoption of new technology. The effects of gamification, including these outcomes, have been demonstrated to lead to an improvement in activity or process efficiency (Hamari and Koivisto, 2015; Hamari et al., 2014; Huotari and Hamari, 2017; Pedreira et al., 2015), particularly in effecting behavior change across diverse domains such as education (Denny, 2013; Hakulinen et al., 2013; Hamari et al., 2016; Landers, 2014) and business (Hamari, 2013). To design an effective gamification strategy that realizes these objectives, a range of commonly employed gamification elements have been identified, including points, achievements, rules, challenges, feedback, levels, rankings, among others (Hamari et al., 2014).

The advancement of gamification as a method has led to the development of several frameworks that provide structure to the design and implementation process. Prominent among these are the Mechanics-Dynamics-Aesthetics (MDA) framework, the Mechanics-Dynamics-Components (MDC) framework, and the Octalysis framework (Chou, 2019; Hunicke et al., 2004; Sisomboon et al., 2019). The MDA framework, which originated in the education field, is considered an early framework for gamification, offering a systematic approach by categorizing gamification into mechanics, dynamics, and aesthetics. It provides clear definitions for these components. The MDC framework, specifically aimed at the business sector, defines mechanics as elements that stimulate engagement, dynamics as elements that immerse users into the environment, and components as more specific forms of mechanics or dynamics. The Octalysis framework, on the other hand, is a more general framework, focusing on the use of core behavior drivers to motivate users to complete tasks efficiently through an interactive experience.

Several researchers have proposed preliminary gamification frameworks in the field of software engineering, including the G-SPI framework (Herranz et al., 2019), the Gamification Design Framework (Matsubara and Da Silva, 2017), the Software Engineering Gamification Framework (Dal Sasso et al., 2017), and GOAL (Garcia et al., 2017). Recent research in the field of software

engineering has synthesized the current gamification design methods and established a framework for gamified software engineering (Morschheuser et al., 2018). This research utilized a Design Science Research (DSR) approach, where method fragments were collected and assembled into a comprehensive framework. The framework includes the following steps: 1) Analysis of the target audience to determine the feasibility of gamification; 2) Identification of a suitable psychology theory to support the effectiveness of gamification in the activities; 3) Selection of appropriate gamification elements for the activities; 4) Selection of the necessary tools for implementation; 5) Evaluation of results through data collected from questionnaires and expert surveys.

However, despite the progress made in the gamification framework of software engineering, there remain some shortcomings. Firstly, the existing gamification framework is inadequate and lacks practical applicability to the daily software development process in software engineering. Only a limited number of studies offer concrete guidance on gamification design (Prause and Jarke, 2015). Secondly, the correlation between gamification elements and specific impacts is yet to be fully established. Thirdly, much of the research on gamification has focused on qualitative over quantitative analysis, and empirical evidence is often limited (de Almeida Souza et al., 2017; Monteiro et al., 2021). There are only a few studies that present quantitative results of gamification mechanisms in the software engineering domain (Chow and Huang, 2017), and most gamification frameworks or approaches have been developed without a data-driven approach. As the field of gamification continues to expand both theoretically and practically, it is imperative to develop a gamification model for daily development process in software engineering that can accommodate the evolving challenges of gamification.

3.2 Gamification Methodology

In the previous sections, it has been highlighted that the integration of gamification into the software development process presents various challenges, including the need for multidisciplinary knowledge and a lack of a unified approach to gamification design. To address these challenges, this study aims to synthesize the existing literature on gamification and design principles to answer the following research questions:

RQ1: What is the process for building a gamification framework for the day-to-day software development process?

RQ2: What is the approach for selecting gamification elements with the appropriate benefits?

The methodology proposed in this study is aimed at facilitating the implementation of gamification within the software development process. The model provides information regarding the actors involved, potential benefits, the implementation of game elements, and the data involved in the gamification process.

The methodology developed in this study consists of two components: 1) a model for applying gamification to the day-to-day software development process, which includes a method for collecting and analyzing quality data based on users' digital footprints, and 2) a configurable approach for selecting gamification elements with appropriate benefits.

3.2.1 Gamification Model

As previously discussed, the implementation of gamification in the day-to-day software development process necessitates a comprehensive model. This model seeks to provide a systematic approach for the software development process, as well as furnish detailed guidance. The gamification model is comprised of four distinct phases, including Preparation, Platform, Gamification Design, and Development. (Refer to Figure 3.2).

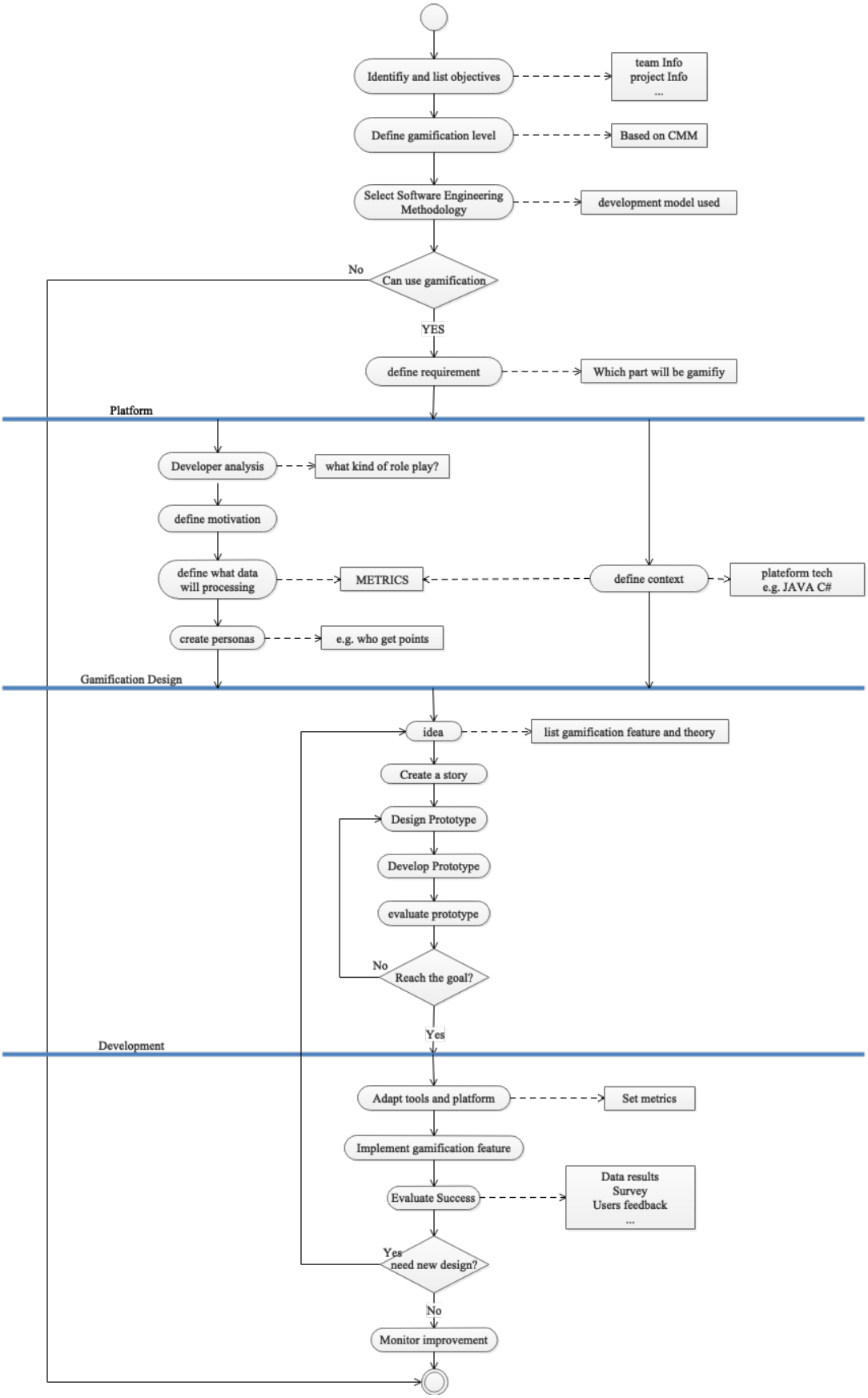


Figure 3.2 Gamification Model

3.2.1.1 Preparation

In this stage of the gamification model, there are five steps that culminate in the transition to the platform phase. The first step involves establishing objectives such as team information, project information, and other relevant details. The second step involves defining the gamification level, which is based on the Capability Maturity Model (CMM) and the Game Maturity Model (Chow and Huang, 2017). This definition of the level helps developers to identify the appropriate portions of the development process for gamification. Subsequently, developers select the methodology that best suits their requirements and evaluate its compatibility with gamification. For instance, the linear nature of the waterfall model in software engineering may not be suitable for implementing feedback strategies in gamification. In the evaluation phase, the compatibility of the selected software engineering methodology with gamification is determined. In cases where the methodology is not suitable, gamification can be skipped, and the original software development process can be maintained.

3.2.1.2 Platform

To analyze the effectiveness of gamification, I have defined two pipelines: the student/developer pipeline and the project pipeline. The first step in the student/developer pipeline involves defining the various roles, such as requirement analyst, function developer, or tester. The second step involves determining the motivations behind gamification based on the role assigned. For example, a requirement analyst might be motivated to increase their engagement in the requirement analysis process. In the third step, a quantitative analysis of the gamification's effectiveness is conducted by defining appropriate metrics of behavior or activities. These metrics are based on the student/developer or project pipeline. Finally, the metrics are assigned to different roles; for example, testers are rewarded with points for completing a test case, while function developers receive points for developing a function (Elgrably and Oliveira, 2018).

3.2.1.3 Gamification Design

In this stage of the methodology, I aim to design a gamified environment for students by utilizing various gamification strategies. The process is divided into four steps. The first step involves the ideation of suitable gamification elements and mechanisms such as points, leaderboards, guides, rewards, etc. In the second step, the researcher integrates these elements and mechanisms in a systematic manner to create a clear and concise guide for the students. The guide is designed to foster professional practical activities within the students. The third step involves the development of a gamification prototype, which is then subjected to a trial phase. If the prototype performs as intended, the process can proceed, otherwise, the researcher will need to revisit the design and make necessary modifications.

3.2.1.4 Development

In this stage, students are engaged in software development within a gamified framework, which aims to enhance their professional practical skills. The first step involves the selection and implementation of appropriate platforms and tools, along with the establishment of corresponding metrics. The gamification elements, such as point systems for specified behaviors and activities, are then integrated into the process. Subsequently, the researcher assesses the effectiveness of the gamification approach through a combination of quantitative analysis and data collected from surveys or interviews, aimed at determining the positive impact of gamification on software engineering practices.

3.2.1.5 Conclusion

The model begins with the identification of a research question and the development of a research plan. The researcher must then decide on the appropriate research design, data collection methods, and data analysis techniques to answer the research question. Once the research plan is in place, the researcher can proceed with the actual implementation of the gamification strategy. This involves the selection of gamification elements, such as points, badges, leaderboards, and missions, that are best suited for the target audience and the research question. The researcher must then design and test the gamification strategy, collecting and analyzing data from the participants. Finally, the researcher must interpret the results of the gamification strategy and draw meaningful conclusions

about the effectiveness of the strategy. This may involve comparing the results to previous literature, and making recommendations for future research.

3.2.2 Gamification Configurable Method

In previous sections, the need for a configurable method to select gamification element sets with relevant benefits was highlighted. Ten widely-utilized gamification elements have been identified, including points, leaderboards, achievements/badges, rewards, levels, story/theme, clear goals, feedback, progress, and challenge (Hamari et al., 2014).

Points and leaderboards are among the most widely used gamification elements, particularly when used in conjunction. Researchers have used these elements to improve user motivation and engagement by using points as a benchmark for performance and utilizing leaderboards to reflect user rankings based on points (Albilali and Qureshi, 2016; Bianchini et al., 2016a, 2016b; Chow and Huang, 2017; Dubois and Tamburrelli, 2013; Francisco-Aparicio et al., 2013; Garcia et al., 2017; Nah et al., 2014; E. D. Santos and Oliveira, 2018; Sisomboon et al., 2019; Siutila, 2018; Unkelos-Shpigel and Hadar, 2015). The leaderboard serves to showcase the relative position of users within a group or team. The effectiveness of this pairing is further supported by the psychological principle of self-determination theory (SDT) (Botte et al., 2020; Shi and Cristea, 2016).

Another effective pairing is rewards and achievements/badges, which researchers have used to encourage user participation (Chow and Huang, 2017; Elgrably and Oliveira, 2018; Xiaozhou Li, 2018; Prause and Jarke, 2015). When users complete tasks, they are awarded badges or other achievement milestones, which serves to keep users focused on the task at hand and increase their engagement.

The use of challenges and feedback has been shown to increase user motivation and engagement (Albilali and Qureshi, 2016; Chow and Huang, 2017; Nah et al., 2013; Prause and Jarke, 2015; Sisomboon et al., 2019; Uskov and Sekar, 2014; Wongso et al., 2014). This is supported by the role-motivation-interaction (RMI) theory (El Shoubashy et al., 2020). Clear goals, another gamification element, have been utilized to enhance user focus (Albilali and Qureshi, 2016; Chow and Huang,

2017; Dubois and Tamburrelli, 2013; Elgrably and Oliveira, 2018; García et al., 2017; Simões et al., 2013; Sisomboon et al., 2019; Wongso et al., 2014).

In order to select appropriate gamification strategies for software engineering practice, three gamification strategy pairs have been summarized. These include points and leaderboards, rewards and achievements/badges, and challenges and feedback. The combination of points and leaderboards has been shown to increase users' motivation by providing a standard for their work and reflecting their relative position within a team or group. The pairing of rewards and achievements/badges is effective in increasing user participation and engagement by simulating focus on tasks. The use of challenges and feedback together helps increase users' motivation and engagement, supported by the role-motivation-interaction theory. These gamification strategy pairs can serve as useful reference for future software engineering practice.

3.3 Conclusion

In this study, I present a comprehensive model for the daily implementation of gamification research. The model incorporates the ability to apply quantitative analysis and provides a systematic approach for researchers to analyze and evaluate the effectiveness of gamification strategies.

In addition to the model for the day-to-day development process, I also present a gamification configurable method to assist researchers in their selection of gamification elements. This method takes into consideration the intended outcome of the gamification strategy, providing a systematic approach to gamification element selection.

In conclusion, the model and the configurable method presented in this study provide a comprehensive and systematic approach to gamification research. These tools will enable researchers to design, implement, and evaluate gamification strategies in a rigorous and meaningful way, thereby advancing the field and contributing to a deeper understanding of the potential of gamification.

Chapter 4 Studies Setting

This chapter focuses on the basic background of observational as well as experimental study, including the backgrounds of the participating students, the observational or experimental duration, the experimental task, the observational or experimental method, and so on. It also describes the ethical preparations made for this experiment. For example, whether the students' understanding of the whole experimental process and the impact of the results before the start of the experiment are described.

4.1 Setting

4.1.1 Observational Study Setting

I conducted an observational study that involved 237 third-year undergraduate students from a European university. The sample included students from different academic years (2018, 2019, and 2020), with an average of approximately 80 students per academic year. The module was delivered face-to-face in 2018 and 2019, and virtually in 2020. To ensure that my results were not influenced by exogenous factors, I constructed a dummy variable to distinguish between the different methods of module delivery, and results showed that this variable did not affect my findings. Second, I expected participants to have similar programming experiences and academic backgrounds since they were enrolled in the same module, Software Engineering[2], within the same major and university, and were taught by the same instructor.

I collect data from public[3] git repositories belonging to undergraduate students enrolled in the Software Engineering module. At the beginning of the module, students were introduced to using git repositories and spent two weeks learning the principles of unit testing, how to apply it with iteration development, and how to use it to drive TDD-style (test-first dynamic development). I demonstrated some examples of TDD-style development processes using Python and Java. Thus, students were equipped with basic knowledge of testing and TDD, and I did not expect them to have any prior experience or expertise with TDD.

All students in my study were assigned the same task, with was released in week 3 and lasted for four weeks. The task involved completing specific functions and writing test cases to test the corresponding production codes. Students were given the flexibility to implement the function in any programming language of their choice, such as Java, Python, C#, Go, Haskell, among others.

[2] Software Engineering is a module of final year bachelor's degree, and provides students with a solid grounding in various aspects of software engineering process related to building large software systems.

[3] Public git repository allows anyone to view, copy, use, and analyze the contents of the repository, so there are no ethic issues associated with the use of the public repository in this study.

Specifically, the task required them to compute the lowest common ancestor (LCA) in a graph structured as a binomial tree and implement the LCA in directed acyclic graphs (DAG) (Bender et al., 2005). The instructor directed the students to accomplish the task on GitHub and encouraged them to leave a digital footprint in the form of commits. However, the instructor did not impose any mandatory requirements concerning the frequency or contents of the commits.

The source codes and commits of the students were manually collected from their git repositories. Python was used to conduct quantitative analyses on the codes, which generated code quality metrics such as Cyclomatic Complexity (CC), Halstead metrics, line of code (LOC), and Maintainability Index (MI). I then construct proxies for TDD behaviors and engagement level in development activity by evaluating their digital footprint through the commits gathered from git repositories. Detailed variable construction is reported in chapter 5.

4.1.2 Experimental Studies Setting

This part describes the setup of two separate experiments. The first experiment aims to validate the effectiveness of gamification in TDD practice in group setting and the second experiment focuses on individual setting.

4.1.2.1 Group Experiment Setting

The objective of this experiment is to examine whether the gamification method helps students to establish and uphold professional software engineering practices in group setting, such as adhering to TDD behaviors and improving engagement levels, which will afterward benefit maintainability. The experiment is conducted in the Software Design and Implementation module of an Irish university; the participants of this study are the students in their final semester of the Bachelor's degree, who possess basic programming skills and are ready to enter the job market. Hence, the results and observations of this experiment can be assumed to be representative of both academic and entry-level engineers of industrial circumstances.

The module's assignment necessitates the teams to produce a complex application for the Android platform. Our trial period is 45 days. Prior to the experiment, we educate the students in the fundamentals of unit testing and its application via an iterative development approach for two weeks. They are also instructed in test-first dynamic development (writing unit test first and then coding) with TDD-style, thereby attaining fundamental knowledge about using GitHub and TDD, albeit not expected to be experts. The students have selected Java as their major programming language for the development of the application.

Participation in the experiment is voluntary, with no grading rewards associated, thus allowing students the autonomy to choose whether or not to participate in the experiment. This is in accordance with the recommendations of Callan et al. (2015) that, when utilizing gamification in educational settings, students should be given the option to volunteer.

The experiment was conducted across two development teams. The treatment group comprised six third-year undergraduate students from the 2020-2021 academic year, and the control group

comprised twenty third-year undergraduate students from the 2021-2022 academic year. Both groups were from the same module, receiving the same content and teaching form (online), just from different academic years. The number of participants in the treatment group was limited, as students had to be entirely voluntary and demonstrate no hesitation in receiving the gamification treatment. To better demonstrate the effectiveness of gamification, a control group was also established, which did not receive any gamification treatment during the experiment period. The details of the treatment are reported in chapter 5, gamification strategy part.

The experiment design follows the Pretest-Posttest Control Group Design. On the first day, students in the treatment group were introduced to gamification strategies and how to 'play'. In order to compare the performance with and without gamification, the experiment was divided into two stages. From day 1 to day 22, the treatment group (O1) did not receive any gamification treatment, whereas from day 23 to day 45, the treatment group (O2) received a gamification treatment every 7 days on days 23, 30, and 37. The gamification treatment consisted of scores, leaderboard ranking, and feedback. This information is communicated via emails. The email content includes their points, their position on the leaderboard, and advice on how to earn more points. An explanation of the point calculation method is also provided at beginning. Figure 4.1 shows an example of notification email and Figure 4.2 shows that the students were emailed individually.

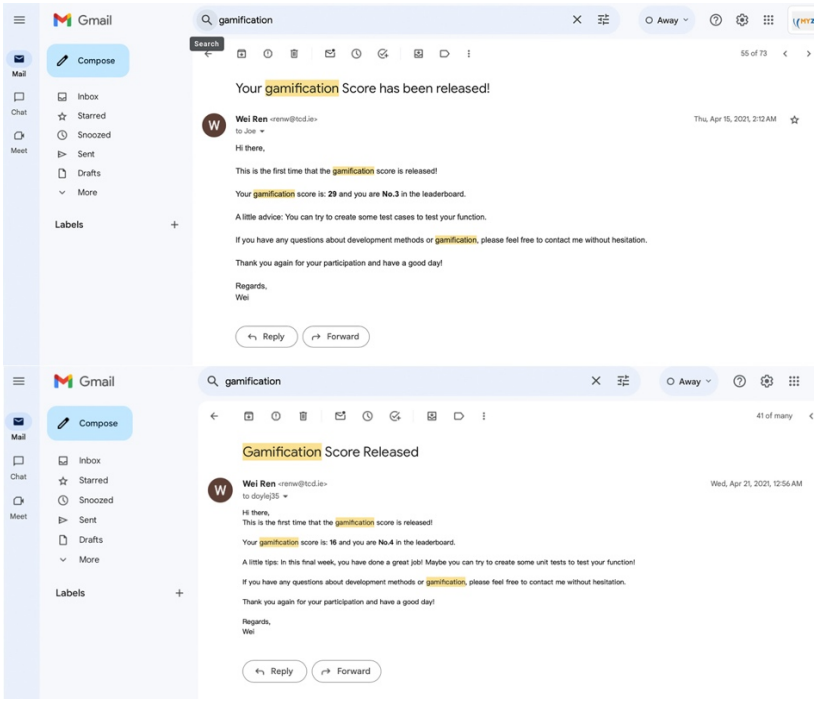


Figure 4.1 Example of Notification Email

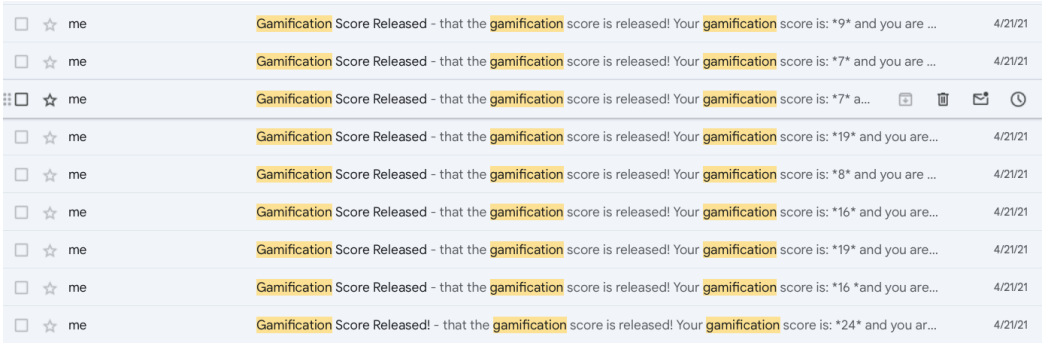


Figure 4.2 Students Emailed Individually

Midway into the experiment, the gamification intervention is introduced with the purpose of monitoring shifts in the performance of the student cohort both before and after the intervention. This methodology seeks a more lucid comprehension of how gamification impacts their performance. Employing the identical group of students throughout the entire process is anticipated to yield more substantial and compelling outcomes.

A more detailed description of the gamification methods can be found in chapter 5. The students in the control group did not receive any information regarding gamification during the duration of the experiment, as represented by O3 and O4 in Table 5.1. Meanwhile, all students in both groups were provided with a clear understanding of the task and its deadline. Data was collected one week prior to the deadline to eliminate any confounding effects (Costello, 1984).

Treatment Group	O1	X (gamification treatment)	O2
Control Group	O3		O4

Table 4.1 Experiment Structure

4.1.2.2 Individual Experiment Setting

Prior studies have been criticized for their inadequate experimental design in gamification research, including a limited number of participants and short experiment duration (Denny et al., 2018; Hamari et al., 2014; Seaborn and Fels, 2015). In order to address these limitations, a controlled experiment was conducted with 162 third-year undergraduate students from a European university, spanning a period of 44 days. The participants were comprised of 73 students from the academic year 2021 and 89 students from the academic year 2022. The study was conducted on an individual basis, and due to the Covid-19 pandemic, the module was delivered online in both academic years. To ensure consistency and control for extraneous variables, the teaching materials, methods of delivering lectures, and tasks were identical in both academic years. Additionally, the participants shared similar programming experiences and academic backgrounds as they were all enrolled in the same module (Software Engineering) at the same university and major, and taught by the same instructor.

The data was collected from the students' git repositories. At the start of the module, the students were familiarized with the usage of git repositories. In preparation for the task, the principles of unit testing, along with its application in iteration development and TDD-style (test-driven development) were imparted to the students. Demonstrative examples of TDD-style development processes were provided using both Python and Java programming languages to ensure that the students possess a basic understanding of testing and TDD, although it was not expected for them to be experts in the field.

The assignment task, which was released in week 7, was administered to all students over a period of approximately six weeks (44 days). To eliminate the potential impact of the task deadline on the experiment results, the experiment ended one week prior to the task deadline (Costello, 1984). The assignment required students to develop specific functions and accompanying test cases for the corresponding production codes. The students were allowed to choose their preferred programming language, including Java, Python, C#, Go, Haskell, and others. They were then tasked with utilizing the GitHub API to retrieve and display data related to the logged-in developer, followed by the construction of a data visualization that sheds light on an aspect of the software engineering process,

such as a social graph of developers and projects, or an illustration of individual or team performance.

The source code and commits of the students were collected from their Git repositories. I conducted a quantitative analysis of the code using Python, and generated code quality metrics such as Cyclomatic Complexity and Maintainability Index. Subsequently, I evaluated the students' digital footprint through the analysis of their commits in the Git repositories, and constructed proxies for their TDD behaviors and level of engagement in development activities. The method for variable construction is detailed in chapter 5.

In accordance with the recommendations by (Callan et al., 2015), the participation of the students in the experiment was completely voluntary and no grades were associated with the experiment. This approach allowed the students to have autonomy over their participation and removed any potential coercion to participate in the experiment.

In this study, gamification strategies, such as rewards, points, leaderboards, and feedback, are employed. To test my hypotheses, the experiment is conducted in three parts. Part 1 consists of two groups: the treatment group, comprising 89 students from academic year 2022, who receive the gamification strategies, and the control group, comprising 73 students from academic year 2021, who do not receive the strategies. On the first day, the treatment group students are introduced to the gamification strategies and instructed on how to ‘play'. They receive 11 gamification interventions at four-day intervals, from day 4 to day 44, communicated via emails. The email content includes their points, their position on the leaderboard, and advice on how to earn more points. The email also includes accumulated gamification score and next time intervention. An explanation of the point calculation method is also provided. Figure 4.3 and 4.4 shows the contents of email and how it was sent individually.

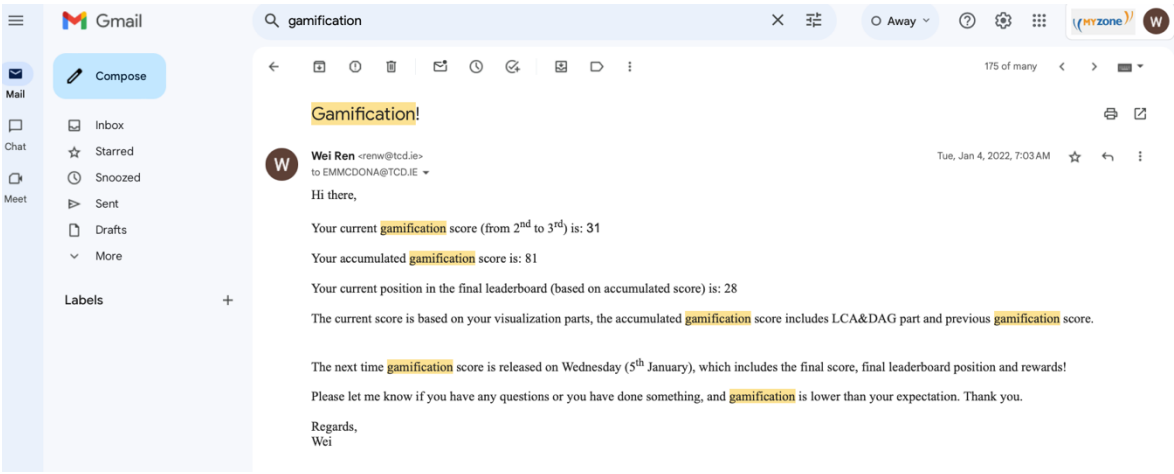


Figure 4.3 An Example of Email Content

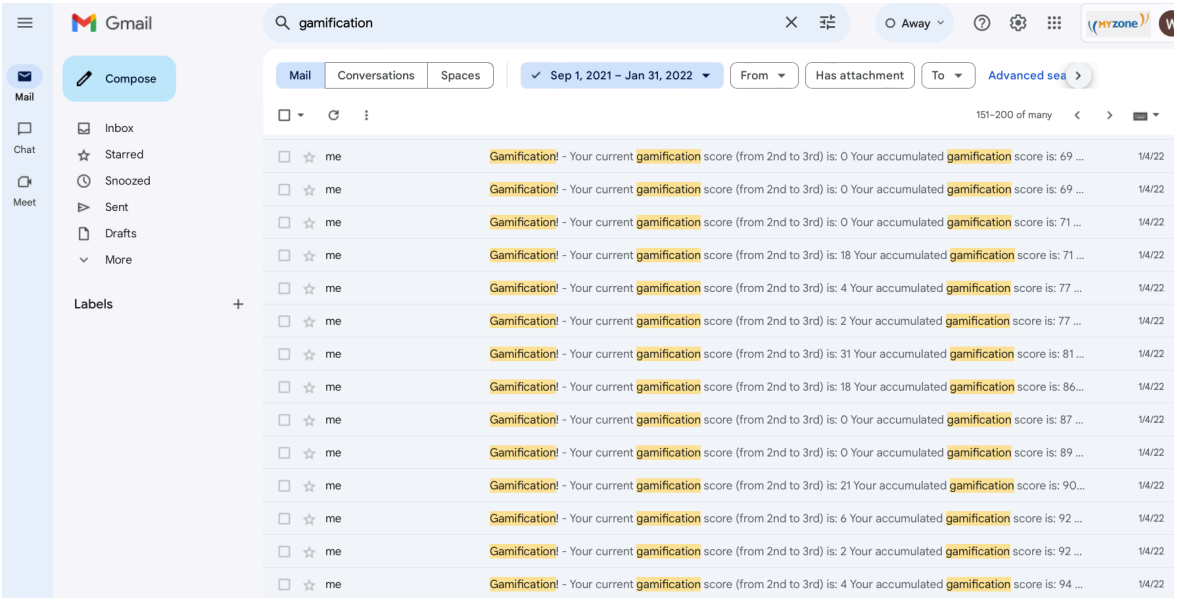


Figure 4.4 Students Emailed Individually

To test the second hypothesis, a second part of the experiment was conducted to distinguish the impacts of different gamification strategies on TDD practice, and compare the impact of a combination of these strategies and their individual effects. As previously stated, gamification strategies included points, leaderboards and feedback. To test Hypothesis 2, a basic strategy 'reward' was implemented across the treatment group, and this group was further divided into three subgroups (1, 2 and 3). Group 1 received only points, group 2 received both points and leaderboards, and group 3 received all gamification strategies, including points, leaderboards and feedback. Students were able to select their preferred subgroup voluntarily, and a number of students (n = 29, 30, 30) were randomly selected to balance the size of each subgroup.

To evaluate the persistence of the gamification effect, I randomly assigned half of the participants in each subgroup (Group 1, 2, and 3) of the second part of the experiment to a control group, ceasing to receive gamification strategies after 8 applications. The remaining participants in each subgroup continued to receive the strategies until the end of the experiment (see Figure 4.1). This allowed us to test the third hypothesis, concerning the endurance of the gamification effect after withdrawal.

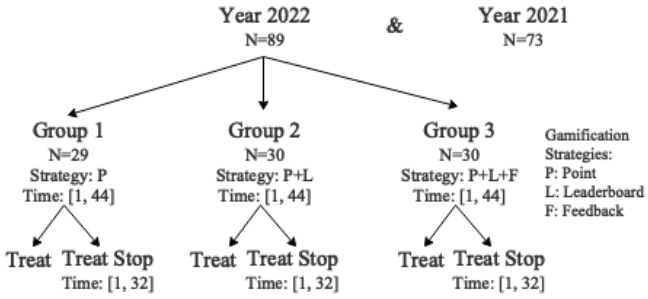


Figure 4.1 Experiment Introduction

4.2 Ethics Preparation

I previously implemented gamification as part of our software engineering teaching strategy, specifically focusing on Test-driven development (TDD) in the modules CSU33012 (Software Engineering) and CSU33D06 (Software Design and Implementation). I conducted an ex post facto study on the performance of students in these modules for potential publication.

4.2.1 Participants Recruiting and Consent

Three consent processes were established for corresponding studies: observational study, group experiment, and individual experiment.

The observational study reviewed the performance of students in the modules CSU33012 from academic year 2018 to 2020, where TDD was an essential part of teaching. Students who had participated in these teaching activities were recruited, and their consent was obtained for inclusion in the study after the module's completion. The data collected was utilized for analysis and potential publication. This study solely sought students' consent for the use of their data.

The group experiment evaluated the performance of students in the CSU33D06 modules in the academic year 2021, while the individual experiment examined the performance of students in the CSU33012 modules from the academic years 2021 to 2022, where both TDD and gamification were employed as teaching mechanisms. Students who had already participated in these teaching activities were recruited, and their consent was obtained for inclusion in the study after the module's conclusion. The data collected was utilized for analysis and potential publication.

These two experiments only requested students' consent for the use of their data. The implementation of teaching activities and gamification strategies preceded this study. Participation in the gamification was optional and extended to all students without any reference to the study. Students were not asked to sign up for the study to avoid any conflict of interest. Separate permission was sought after the activities were completed from students willing to participate in the study regarding the gamification learning experience. No students were disadvantaged based on their

agreement to participate in the study. Also, there was no further action required of them as they had already voluntarily engaged in the teaching activities with appropriate permission.

Only students who signed up for participant consent were engaged in the teaching activity (gamification process). Notification of the teaching activity introduction was sent to all students in the modules. Student participation was entirely voluntary, and they were informed that participation might enhance their engagement in the project, such as through leaderboard rankings reflecting their engagement level.

The data generated during the teaching activity was used to identify measurable differences in performance attributable to the application of gamification strategies.

All participants gave their informed consent to be involved in the study and the detail about consent form can be seen in Appendix 2-a and 2-c.

4.2.2 Potential Influence on Participants

The potential harm lay in the mishandling of individual students' performance data, which could expose them to risks regarding data privacy. However, the potential intangible benefit was that students received feedback on their performance, which could be advantageous for future endeavors such as job interviews where they could demonstrate their experience with TDD and gamification.

Regarding conflicts of interest, this study was designed as an ex post facto study specifically to avoid such conflicts in the teaching of the module. Participation was entirely voluntary and had no impact on the teaching process.

4.2.3 Data Protection

To enhance data protection, a thorough anonymization process was employed. Personal data was completely deleted from files, and placeholders such as '*' was used to represent individuals' personal information, including Github IDs and email addresses. Additionally, a new file without any personal data tags was created, with data obscured through altered values. For example, commit

details were transformed into total numbers of commits. All data was used solely for research purposes, including research papers, and was stored only as long as necessary for the study.

All personal data was fully anonymized and deleted after the study concludes. While it may have still been possible for interested parties to identify the cohort, the study was designed to prevent individual identities from being discerned by interested parties.

The detail of data consent can be seen in Appendix 2-b and 2-d.

Chapter 5 Methodology

In this chapter, I introduced variables definition in observational study and two experimental studies, including variables that measure TDD, variables that measure engagement, and variables that measure maintainability. At the same time, the gamification methods used in two experimental studies and the variables used to measure gamification are also introduced.

In observational study, I examine two relationships: TDD behaviors and engagement level in development activity; engagement and software maintainability. To investigate the first relationship, I treat TDD as independent variables and engagement level as the dependent variable. Similarly, in order to explore the potential positive relationship between engagement level and maintainability, I use engagement response variables as independent variables and maintainability response variables as dependent variables in my regression analysis.

The experiments examine whether gamification motivates students to develop and maintain TDD practice. More precisely, I examine the gamification effectiveness on whether students follow TDD behaviors, increase the engagement level in development activity, and have a positive impact on maintainability, by using statistical and graphic analysis. I adopted an indicator variable to act as a proxy for whether students received gamification treatment, which serves as the independent variable. TDD behaviors, engagement level, and software maintainability were used as the dependent variables. I collect digital footprints (commits) from GitHub repositories and use the data extracted from the codebase to compute maintainability at different points in the repository's timeline.

5.1 Measurement of TDD

Test-driven Development (TDD) is a cyclic development technique that involves two fundamental behaviors: test-first and iteration development. The test-first behavior requires developers to write a test before writing any production code, while iteration development involves making small incremental changes to the codebase until the desired functionality is achieved. An optimal TDD cycle is typically short in duration (Fucci et al., 2016; Jeffries and Melnik, 2007). In this thesis, I construct the response variables for TDD from two aspects: writing unit testing sequence (*SEQ*), and the number of development cycles generated (*Cycle*).

5.1.1 Behavior proxy (1). Unit Testing Sequence

The first response variable TDD behavior, referred to as sequence (*SEQ*), involves creating a failure test case before writing production code and ensuring that the test passes after the production code has been created. To facilitate the analysis of coding activity, I divided it into discrete episodes and assigned *SEQ* to each episode. The delineation of episodes was determined based on the completion of a development cycle. *SEQ* is a binary variable, with a value of 1 indicating that students wrote a test case before writing production codes, and 0 indicating the opposite order. Table 1 provides a detailed description of the test-first procedure. In order to accurately identify the *SEQ* values, I manually analyzed the commits, examining the description of each commit and the corresponding code changes to determine whether the development cycle followed a test-first approach. None of the observed students adopted a mix method, meaning that they either wrote test cases first or wrote them later, but not both, during the completion of the tasks. The *SEQ* equation is shown below:

$$SEQ = \begin{cases} 1 & type(i) = test - first \\ 0 & otherwise \end{cases}$$

Test creation -> Test compilation error -> Function code editing -> Test failure -> Code editing -> Test pass
Test creation -> Test compilation error -> Function code editing -> Test pass
Test creation -> Function code editing -> Test failure -> Code editing -> Test pass
Test creation -> Function code editing -> Test pass

Table 5.1 Test-Driven Development Definition

5.1.2 Behavior proxy (2). Development Cycle

In addition to Sequence, iteration development is another important characteristic of TDD, and it is reflected by the cycles generated (*Cycle*). Hence, I utilize *Cycle* as another TDD response variable. This measure captures the cyclic nature of the development process employed. The use of *Cycle* to represent iteration development is attributable to the constant task and time constraints. A higher *Cycle* value indicates more adherence to the TDD style, which is characterized by incremental code changes.

I defined the development cycle by manually analyzing the commits and code changing, which is the process of creating a new function and passing all existing tests. A test compilation error will render a cycle incomplete, thereby implying that a cycle can only be considered valid when the test compilation is successful. The level of commit granularity generally provides a sufficient basis for observing the correct order. However, in cases where the commit granularity is insufficient, an alternative approach would involve manually reviewing the code to determine the number of development cycles. If the students generate test cases before production code, it is classified as test-first.

I have further classified *Cycle* into two types based on the criterion of whether to add a test case first: TDD_Cycle (*TC*) and General_Cycle (*GC*), which are recognized by GitHub commits. *TC* is determined by *Cycle* following the test first approach, whereas *GC* is determined by test last style. It is worth noting that regardless of whether the iterative speed is slow or fast, both *TC* and *GC* are indicative of TDD characteristics.

I also constructed the number of test cases (*Test*) to represent the essential elements of good TDD behaviors. The *Test* variable is defined as the sum of all processes of creating a new test case and it being passed by the corresponding function over the experiment period. The response variables for TDD are shown in Table 5.2.

Type	Variables	Data Type
Unit Testing Sequence	SEQ	Dummy
Development Cycle	Cycle	Continuous
Development Cycle	TC (Cycle following TDD style)	Continuous
Development Cycle	GC (Cycle following other styles)	Continuous
Number of Test Cases	Test	Continuous

Table 5.2. Test-Driven Development Response Variables

Despite being an essential aspect of TDD, refactoring is not addressed in this study due to the students' limited exposure to small-scale projects, which restricts their opportunities for engaging in refactoring activities. Additionally, the complexity of learning refactoring techniques poses a challenge for students, as it is a skill that is better suited for experienced developers. Consequently, this study does not consider motivating refactoring.

5.2 Measurement of Engagement Level in Development Activities

5.2.1 Background

In this thesis, I focused on the measurement of engagement in software development activities. One method that offers insights into development efforts is through analysis git commits (Chávez et al., 2017; Dyer et al., 2015; Wen et al., 2016). Commits are snapshots of the entire git repository at specific times, providing essential information for understanding software development activity (Chávez et al., 2017; Hal et al., 2019; Kolassa et al., 2013; Wen et al., 2016). Commit numbers have been used to reflect various aspects of software development, such as continuous integration process (Baltes et al., 2018), maintenance activities (Levin and Yehudai, 2017; Mariano et al., 2019), and bugginess (Eyolfson et al., 2011).

Utilizing git for code submission enables the measurement of an individual's engagement in development process. Factors such as frequency, numeric, and quality of commits serve as indicators of engagement level in development activities. For instance, a high volume of submissions may denote higher engagement level, whereas sporadic or minimal commits may imply lower levels of engagement.

Therefore, Git commits provide an objective means to quantify behavioral engagement in software development activities. By measuring the frequency and number of commits, this thesis can gain insight into the level of engagement of individual developers and teams in various software development tasks. The use of Git commits as a measure of engagement is important to gain a better understanding of the impact of engagement on software development.

5.2.2 The Engagement proxy

The engagement response variables is commit-based, including number of commits (*NC*) and commit frequency (*FEQ*). The NC measure represents the total number of commits made by developers during the development process, while the FEQ measure reflects the frequency of commits. A higher number of commits and a higher frequency value suggest more development

activity (Kolassa et al., 2013) , and a higher engagement level. To ensure the validity of my results, I exclude duplicate changes and comments for codes. In this study, I focus on three types of engagement, for example, overall engagement, engagement in coding and engagement in testing, engagement in different types of testing.

5.2.2.1 Proxy (1). Commit Number

To examine whether there are any differences in maintainability between the engagement in coding and testing, this study decomposes the total number of commits (*NUMBER_COMMITS*) into two distinct categories: commits related to production codes (*PROD_COMMITS*) and commits related to test codes (*TEST_COMMITS*).[4] If a commit contains both test and production code, it will be counted in both *TEST_COMMITS* and *PROD_COMMITS*, but only once in *NUMBER_COMMITS*. Furthermore, to explore potential impacts on maintainability based on the different types of testing activities engaged in, TEST_COMMITS is further divided into two types: creating new test cases (*NEW_TEST_COMMITS*) and maintaining existing test cases (*MAINTAIN_TEST_COMMITS*).[5]

Type	Variables	Data Type
Overall	NUMBER_COMMITS	Continuous
Commits Related to Production Code	PROD_COMMITS	Continuous
Commits Related to Test Code	TEST_COMMITS	Continuous
Commits Related to Creating New Test Case	NEW_TEST_COMMITS	Continuous
Commits Related to Maintenance New Test Case	MAIN_TEST_COMMITS	Continuous

Table 5.3 Engagement Response Variables - Commit Number

5.2.2.2 Proxy (2). Commit Frequency

FEQ refers to the frequency of updating repositories in a certain period. The calculation of *FEQ* is shown below, where the denominator is the number of working days (excluding weekends and bank holidays), between the first and last commits. Although all students were assigned the same tasks, their working periods varied widely, ranging from 2 to 21 days. Similar with the number of commits (NUMBER_COMMITS*)*, I also divide *FEQ* into two categories: the frequency of updating

[4] *NTC* refers to the total number of commits for updating test cases, such as creating or modifying test codes. *NPC* is defined as the total number of commits for updating production codes.

[5] If a commit contains the record of both, this commit will be counted to *NNTC* and *NMTC* separately.

production code (*PROD_FEQ*) and the frequency of updating test code (*TEST_FEQ*). And then TEST_FEQ is further divided into two categories: the frequency of updating new test cases (*NEW_TEST_FEQ*) and the frequency of maintaining existing test cases (*MAINTAIN_TEST_FEQ*).

$$FEQ = \frac{\text{NUMBER\_COMMITS}}{Number\ of\ Days}$$

$$PROD\_FEQ = \frac{\text{PROD\_COMMITS}}{Number\ of\ Days} \qquad TEST\_FEQ = \frac{\text{TEST\_COMMITS}}{Number\ of\ Days}$$

$$NEW\_TEST\_FEQ = \frac{NEW\_TEST\_COMMITS}{Number\ of\ Days} \qquad MAINTAIN\_TEST\_FEQ = \frac{MAINTAIN\_TEST\_COMMITS}{Number\ of\ Days}$$

Type	Variables	Data Type
Overall	FEQ	Continuous
Frequency of Updating Production Code	PROD_FEQ	Continuous
Frequency of Updating Test Code	TEST_FEQ	Continuous
Frequency of Creating New Test Code	NEW_TEST_FEQ	Continuous
Frequency of Maintenance Test Code	MAINTAIN_TEST_FEQ	Continuous

Table 5.4 Engagement Response Variables - Frequency

5.2.3 Limitation of Engagement Measurement

However, it's essential to acknowledge the limitations of using git commits as a single measure of engagement in development activities. Firstly, not all development activities can be reflected in commits, especially if developers are working on debugging without committing changes (Just et al., 2016). Second, the quality of commits can fluctuate, as some may consist of minor code refactoring or adjustments, predominantly serving to maintain legacy code rather than constituting significant contributions. (Kim et al., 2020). So, commit-based engagement metrics may offer more reliable insights for initiating new projects rather than maintaining legacy codebases. Furthermore, commit-based metrics of engagement may not be suitable for maintaining or developing highly complex, extensive systems due to the inherent limitations of git commit data in capturing the intricacies of such endeavors (Perez De Rosso and Jackson, 2016). Moreover, factors such as collaboration and problem-solving skills may not be adequately captured through commit analysis alone (Martinez and Monperrus, 2019).

Considering these constraints, one alternative approach is conducting questionnaires to gather developers' subjective opinions, providing valuable insights into their levels of engagement

(Breevaart et al., 2012). Additionally, emerging techniques like the utilization of biological data hold promise in offering a more precise measure of engagement, potentially tapping into physiological indicators such as heart rate variability or electrodermal activity (Diaz and Yudin, 2017). Furthermore, advancements in artificial intelligence present possibilities, with AI agents capable of analyzing various facets of developer behavior and interaction to infer levels of engagement (Han et al., 2023). These alternative methods offer different ways to enhance our understanding and measurement of engagement in development activities.

However, given the focus on measuring engagement in development activities for students working on developing a new project that is neither huge nor very complex, commit-based engagement metrics may be suitable. In this context, where the projects are smaller and less intricate, git commit data could provide valuable insights into student engagement levels. It allows for objective assessment and can help track progress and involvement in project development.

In conclusion, choosing git commits as a metric for gauging engagement in development activities is favored for its objectivity over subjectivity. Despite its limitations, relying on git commits offers a straightforward and impartial approach to assessing involvement in the development process. However, it's essential to acknowledge its constraints and explore ways to enhance its effectiveness in providing insights into team performance and individual contributions in software development projects.

5.3 Maintainability

Previous studies have examined maintainability from various perspectives: including process, architecture, and code level (Dhillon, 2006). To measure maintainability, researchers commonly use metrics such as mean time to repair (MTTR), mean preventive maintenance time (MPMT), mean corrective maintenance time (MCMT), and maximum corrective maintenance time (MaCMT). Apart from those, metrics like Halstead software science, McCabe's Cyclomatic Complexity

(Alsolai and Roper, 2020) , and Maintainability index (Ardito et al., 2020) have been used. For my study, I focus on overall maintainability and analyze codes generated by the latest commit.

Consistent with prior research, I use Cyclomatic Complexity (*CC*) as a measure of maintainability. CC counts the total number of linearly independent paths through a program's source code, with higher CC indicating lower maintainability. Another metric I use is the maintainability index (*MI*), which was introduced by the Software Engineering Institute, Carnegie Mellon University (SEI) (D. Coleman et al., 1994). A higher MI indicates better software maintainability. Recently, Microsoft Visual Studio proposed a new calculation method for *MI* (Reddy and Ojha, 2019) , as shown in equation below:

$$MI = MAX\left(0, \left(171 - 5.2 \times lnV - 0.23 \times G - \frac{16.2 \times lnLOC \times 100}{171}\right)\right)$$

Here, V represents Halsted Volume, G represents McCabe's Cyclomatic Complexity, and LOC is the total count of code lines. I use CC and MI as response variables of maintainability in my study, and focus on the latest commit to analyze the final maintainability of the tasks.

Type	Variables	Data Type
Cyclomatic Complexity	CC	Continuous
Maintainability Index	MI	Continuous

Table 5.5 Maintainability

To calculate MI and CC more directly and quickly, I used the official python library Multimetric[4], which aims to calculate code metrics (e.g. Line of code, Maintainability index, Cyclomatic complexity, Halstead science metrics, etc.) in various languages (e.g. C, C++, C#, Haskell, Go, Java, Python, Ruby, etc.). Python library is a collection of modules that contain functions and classes that can be used by other programs to perform various tasks. The library uses standard methods to calculate the corresponding metrics, and it has passed python's dependency verification. Therefore, we use this Multimetric to obtain maintainability related data.

[4] https://pypi.org/project/multimetric/#description

In order to obtain more suitable maintainability data in different experimental environments, I adopted different calculation methods for maintainability in different experiments. To elucidate the effect of gamification on group performance (Experiment I), I construct multiple maintainability indicators, including CC_mean, CC_o, MI_mean, and MI_o. CC_mean reflects the mean value of CC over the course of the entire experiment, while CC_o reflects the CC value upon completion of the project, thereby providing an overall view of the influence of gamification. Similarly, MI_mean and MI_o are used to evaluate the effects of gamification from a global perspective. I employ the fixed effects model (FEM) to explore the correlation between gamification and software maintainability (Hedges, 1994), taking gamification as independent variables, and CC, CC_mean, CC_o, MI, MI_mean, and MI_o as dependent variables.

In Experiment II (individual setting), I chose the maintainability of the code at the time of the last commit, however there may be some risks associated with using MI and CC alone. For example, although the calculation of *MI* is considered the size of software (LOC), but it is varied due to different programming languages. To better interpret the maintainability, I consider using the number of functions (*Function*) to process maintainability metrics (*MI* and *CC*). *Function* is defined as a block of organized code that is used to perform a single task, such as showing data with a chart bar, account information, etc. Although, all students must fulfil the same fundamental requirements for the work, but some may go above and beyond. As I have the *Function* variable, so I manipulate *MI* and *CC* with the *Function* variable to generate another two variables: *FM* and *FC*. *FM* means at the same number of functions higher *FM* has better maintainability. Similarly, *FC* means at the same level of functions higher *FC* has better maintainability due to lower *CC*. The formula is shown following:

$$FM = MI \times Function$$

$$FC = \frac{Function}{CC}$$

5.4 Control Variables

In order to control for potential alternative explanations, I examine other factors that may influence engagement in software development activities and maintainability. Gender is a factor that has been linked to engagement (Qiu et al., 2019), and thus, a dummy variable *Gender* is constructed, with a value of 1 for male students and 0 for female students. Readability, as represented by Halsted Difficulty (*HD*) and Comment Ratio (*CR*), is another factor that is believed to have an impact on maintainability (Aggarwal et al., 2002; Alawad et al., 2019; R. Coleman and Boldt, 2018). In addition, I construct a dummy variable *Lan_OO*, which takes the value of 1 if students use object-oriented languages such as Java, Python, or C++, and 0 otherwise. Project size (*LOC*) is also taken into consideration. User experience has been identified as a factor related to maintainability (Banker and Datar, 1989), and is thus included as a control variable. As the students in my sample are undergraduates who are unfamiliar with Github, I measure their programming ability through the variable *use_exp*, which is a proxy for how long they have been signed up for Github. In observational study, I control for the difficulty of tasks (*LCA_DAG)*, which takes the value of 1 if students do LCA, and 0 otherwise. By including these variables in my analysis, I aim to ensure that any observed effects on maintainability can be confidently attributed to the variables of interest.

Type	Variables	Data Type
Readability - Halstead Difficulty	HD	Continuous
Readability – Comment Ratio	CR	Continuous
Object-Oriented Language	Lan_OO	Dummy
Project Size	LOC	Continuous
Different Task	LCA_DAG	Dummy

Table 5.6 Control Variables

5.5 Gamification Design

In this section, I introduced gamification design in Experiment I and Experiment II respectively.

5.5.1 Design Gamification in Group Setting

In the field of software engineering, the objective of gamification is not to entertain engineers, but rather as a mechanism to alter their behaviors (Deterding et al., 2011). I aim to examine whether gamification strategies change students' development behaviors to those that can improve software maintainability[5], and improve engagement levels in development activity[6]. The central aspect of gamification design is the selection of appropriate strategies (Prause et al., 2012). Previous literature suggests that incorporating points, leaderboards, and rewards simultaneously can modify user behavior (Manal M. Alhammad and Moreno, 2018). This can be explained by the fact that students are more likely to conform to the behaviors that are encouraged by researchers, in pursuit of higher scores, positions, or financial rewards. Such desired behavior in this paper refers to TDD style. However, it is worth noting that using scores and ranking in conjunction may lead to demotivation among those at the bottom of the ranking (Manal M. Alhammad and Moreno, 2018). Nonetheless, the small sample size of six students in the gamification group eliminates this possibility. Additionally, continuous feedback can encourage users to remain engaged in the project, further enhancing their level of engagement (Ren et al., 2020). Thus, the design of gamification strategies in this study incorporates points, leaderboards, rewards, and feedback as the primary components.

Points and Leaderboard - The gamification rules are crafted to provide users with points, which enable them to advance their position on the leaderboard and, eventually, receive rewards. Points are assigned based on whether students adopt Test-Driven Development (TDD) behaviors and are

[5] TDD behaviors, including fast iterative development and writing unit testing, can improve software maintainability

[6] Higher engagement level in development activity is positively related to software maintainability

more engaged in development activities. These assessments are obtained through manual analysis of the commits. The provisions of the gamification rules are presented as follows:

1. Generating a failing unit test before the function code, the participant will get 1 point when the unit test passes after finishing the function code.

2. Finishing a development cycle (creating new function code and related test cases), the student will get 2 points.

3. One commit of updating production code, the student will get an extra 1 point.

4. One commit of updating testing code, the student will get an extra 1 point.

As students earn these scores, it is important to ensure that the new changes do not affect existing codes. If the new code causes a bug, then the student will not receive points for knowing that the issue is resolved. Additionally, the testing phase constitutes a crucial aspect of the study. While it is acknowledged that individual test cases may not pass, the commendable effort put forth by the students in conducting these tests is still duly recognized and rewarded. This approach is intended to foster a positive reinforcement mechanism, motivating students to actively engage in further test case execution.

However, the students' contribution to project administration is also factored into the evaluation. Effective project management entails comprehensive documentation and administration (Stellman and Greene, 2005). This study prioritizes Test-Driven Development (TDD) behaviors and engagement levels in development activities, thus assigning fewer points for work in documentation and administration compared to code production. The second segment of the gamification rules is outlined below:

5. Update one commit about documentation (e.g., XML file), the student will get 0.5 points.

6. Update one commit about administration (e.g., merge), the student will get 0.5 points.

In order to support students' self-coordination, they are notified of their position on the leaderboard, and they receive additional leaderboard bonus points based on their position. The rule for getting the bonus is as below:

7. The highest ranking will earn 6 points, and the second highest will get 5 points, etc. The ranking will be refreshed every week, and the points will be accumulated.

The details of getting points are presented in table 5.3.

Behavior	Score
Generating a failing unit test before function code	1
Finishing a development cycle (a unit test and function code)	2
One commit related to documentation	0.5
One commit related to administration	0.5
Score by ranking	[1, 6]

Table 5.3 Point rules

Rewards - At the end of the experiment, students get real-world rewards (shopping vouchers) to motivate their engagement in development activities.

Feedback - The feedback is a suggestion about how to get more points based on students' most recent activity, which is distributed together with points and leaderboards.

5.5.2 Design Gamification in Individual Setting

I aim to examine whether gamification strategies can motivate students to do TDD practice by change students' development behaviors[7], and increase engagement levels in development activity[8], to improve software maintainability. The central aspect of gamification design is the selection of appropriate gamification strategies to formulate the rules (Hamari et al., 2014). My gamification design involves the creation of structured rules and the selection of suitable gamification strategies. The rules, integrated into a real-world scenario, facilitate behavior change and heightened

[7] TDD behaviors, including fast iterative development and writing unit testing, can improve software quality

[8] Higher engagement level in development activity is positively related to software quality

engagement by clearly defining the desired behaviors, activities, etc. and guiding participants towards achieving these goals through the implementation of various gamification strategies.

Previous research indicates that points, leaderboards, feedback, and rewards are the most frequently utilized gamification strategies. The simultaneous adoption of points, leaderboards, and rewards can alter an individual's behavior. This can be attributed to the desire to attain a higher score, position, or financial reward, which incentivizes individuals to engage in behaviors that are encouraged by researchers. In this case, the encouraged behavior refers to TDD practices. Furthermore, providing feedback has been shown to sustain engagement levels and enhance participation in the project (Ren et al., 2020). Hence, in constructing the gamification strategies, I have selected points, leaderboards, rewards, and feedback.

Points and Leaderboard: The gamification rules are designed to give users points, which help them to improve their position on the leaderboard and, ultimately, get rewards. In this study, the calculation of points depends on how well students adhere to TDD behaviors[9] and increase their level of engagement in development activities[10]. The detail about the TDD behaviors and engagement is shown in the following. The leaderboard is a way to support self-coordination. Students are notified of their position on the leaderboard, and they receive additional leaderboard bonus points based on their position. Each time the point is released, the leaderboard is reranked according to the most recent point and the final position on the leaderboard is based on accumulated points. Here are the relevant gamification rules:

1. Generating a development cycle (product code and related unit cases), earn 2 points.

2. Creating a new test case, earn 2 points.

[9] The TDD behaviors include writing more test cases and fast iteration, which means generating more development cycles. The development cycle is defined as a process of creating and testing a function, and it is manually recognized by the commits recording in code repositories (GitHub).

[10] Higher engagement in development activities is positively related to code quality.

3. Writing one commit related to the test case, earning 1 point, and writing one commit related to the production case, earning 0.5 points.

4. The highest ranking each time will earn extra 15 points, and the second highest will get extra 14 points, etc., and only the top 15 students can get these extra points.

Feedback: The feedback mechanism in this study offers targeted recommendations to students based on their recent performance and development activities. For example, a feedback message might state, "In order to enhance your gamification score, it is advisable to incorporate unit testing into your functions."

Reward: At the end of the experiments, the top five students in the leaderboard get real-world rewards (gift cards), which are notified at the beginning of the experiments.

5.6 Measurement of Gamification

In this study, the measurement of gamification treatment is a crucial aspect of the research design. I presented the measurements in experiments I and II, respectively.

5.6.1 Measurement in Group Experiment

Assessing the efficacy of gamification is a central component of my research design. I consider gamification intervention as a 'shock' during the experiment. I have set up three effect-windows: 1 day, 3 days, and 5 days post-gamification treatment, and accordingly created three dummy variables: *gamification_1* (1 day), *gamification_2* (3 days), and *gamification_3* (5 days). For instance, if on day 0 students receive gamification treatment, gamification_1 is equal to 1 on day 1, and 0 otherwise; gamification_2 is equal to 1 on days 1 and 3, and 0 otherwise; and gamification_3 is equal to 1 on days 1, 3, and 5, and 0 otherwise. As the treatment interval is every 7 days, so I observe its impact by the fifth day. I start from day 1 rather than day 0 to better examine the changes in maintainability after introducing gamification and allow one day for students to react. Similarly, I construct variables on a one-day basis rather than a daily basis, since the effect on maintainability is not immediate.

The rationale behind establishing three distinct observation periods was to facilitate a more comprehensive examination of the impacts of gamified stimuli on students at various junctures, encompassing both immediate and prolonged effects of gamification.

5.6.2 Measurement in Individual Experiment

In this study, the measurement of gamification treatment is a crucial aspect of the research design. The gamification treatment is treated as a "shock" treatment during the experiment. Six independent variables, in the form of dummy variables, are constructed to reflect the impact of different gamification strategies on various hypotheses.

To test RQ3a in individually setting, an independent variable named "Gamification" is constructed. Gamification is assigned a value of 1 if the group received gamification treatment and the students

were from the academic year 2021-2022. Conversely, a value of 0 is assigned if the students from the academic year 2020-2021 did not receive gamification treatment.

To test RQ3b, four independent variables are constructed, namely "Leaderboard," "All," "Extra," and "Feedback." These variables are used to compare the effect of a combination of gamification strategies with the effect of using them individually. The variable "Leaderboard" is used to determine the effectiveness of the gamification strategy of leaderboard, by comparing the subgroup's performance between receiving only points (Group 1) and receiving both points and leaderboard (Group 2). The value of "Leaderboard" is 0 if students only receive points as a gamification strategy, and 1 if they receive both points and leaderboard. The variable "All" is used to compare the impact of all gamification strategies (Group 3) with that of only receiving points (Group 1). "All" is 0 if students only receive points, and 1 if they receive all gamification strategies. The variable "Extra" further compares the performance of Group 1, which only received points, with Groups 2 and 3, which received multiple gamification strategies. "Extra" is 0 if students only receive points, and 1 if they receive any extra gamification strategies. The variable "Feedback" distinguishes the impact of the gamification strategy of feedback, with a value of 0 if students receive both points and leaderboard (Group 2), and 1 if students receive all gamification strategies (Group 3). Figure 5.1 illustrates the groups being compared, as well as the gamification strategies each group consists of.

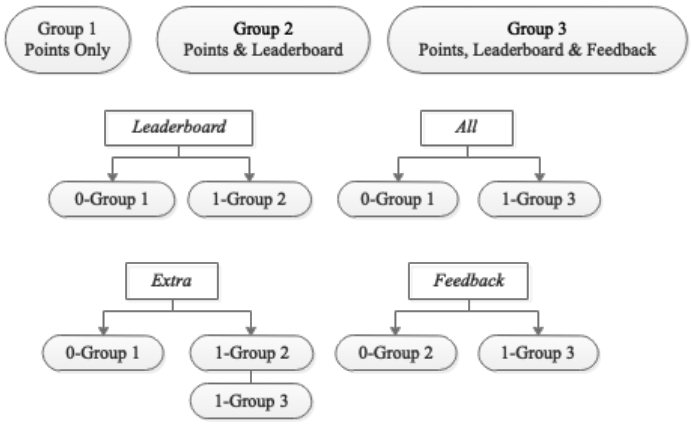


Figure 5.1 Gamification Variables

The students who made up the groups either voluntarily selected them or were chosen at random by the researcher. A dummy variable "Random" was created to test if this confounding factor would affect the results. "Random" is 0 if students voluntarily selected the groups (either Group 1, 2, or 3), and 1 if they were assigned to the groups by the researchers. By comparing the students' performance, it was possible to determine whether the formation strategy of the groups had an impact on the outcomes.

To answer RQ3c, an independent variable named "Continue" is constructed to reflect whether students continuously received the gamification treatment. "Continue" is 0 if the students stopped receiving the gamification treatment and 1 if they continued receiving it.

5.7 Diagnostics

In order to address issues related to skewed distributions and outliers, I employed various data transformations. Specifically, I log-transformed the highly skewed-distributed variables, like *CC* (Landman et al., 2016), and other skewed-distributed variables to normalized variables to ensure analysis is sound. Additionally, I winsorized the continuous variables at the 1% and 99% levels to eliminate the impact of outliers and extreme values (Blaine, 2018). To deal with the heteroscedasticity problem, robust standard errors were included in all models (Krishnamoorthy, 2016).

To address potential problems of multicollinearity, which can affect the accuracy of coefficients estimates and p-values (Farrar and Glauber, 1967), I use post-estimation diagnostics such as the Variance Inflation Factor (VIF). It is reckoned as a multicollinearity issue if the mean VIF is over 4.0 (Jobson, 2012). Results from the untabulated data indicate that all models utilized do not exhibit multicollinearity problems, with mean VIF values ranging from 1.3 to 2.

To address the possibility of endogeneity problems arising from omitted confounding variables that could potentially bias my findings, I conducted an Impact Threshold of a Confounding Variable (ITCV) test, which is an index created by Frank (2000). This test determines the magnitude of the endogeneity error required to invalidate the OLS inference. If the ITCV value is larger than the impacts caused by the endogeneity error, then the OLS inference is less likely to be invalidated. In other words, the larger the ITCV value, the more robust the regression results are to potential endogeneity problems arising from unobserved confounding variables.

The Regression Equation Specification Error Test (RESET) was also applied to detect whether omitted variables are causing model miss specification, as originally proposed by Ramsey (Ramsey, 1969). A RESET p-value below 0.05 implies the existence of an omitted variable bias. The results from the statistical analysis indicated that all RESET values were greater than 0.05, thereby suggesting that the regression models were free from endogeneity problems caused by unobserved variables.

Chapter 6 Result

In this chapter, we present the results of observational study and two experimental studies. In observational study, regression results are mainly used. The results of the two experimental studies are presented in the form of graph and regression results.

6.1 Result of Observational Study

To answer my research questions (RQ2), I employ ordinary least squares (OLS) regressions. OLS regression is a widely adopted statistical method for estimating the coefficients of linear regression equations and exploring the relationships between one or more independent variables and a dependent variable. It is particularly suitable for continuous (or dummy) predictors and continuous response variables, which is the case for my study variables. Therefore, OLS regression is an appropriate analytical method in this study.

In order to investigate whether TDD behaviors affect engagement level in development activities (RQ2a), I employed two response variables: the sequence of writing test case and the number of development cycle. These variables were used to determine whether students were following the TDD method. Additionally, I measured the level of engagement using the number of commits and commits frequency. To answer my second research question (RQ2b), which posits that engagement level is positively related to maintainability, I examined three types of engagement: overall engagement, engagement in coding and engagement in testing. I also examined engagement in different types of testing. The relationship between statistical methods adopted and hypotheses is illustrated in Figure 6.1.

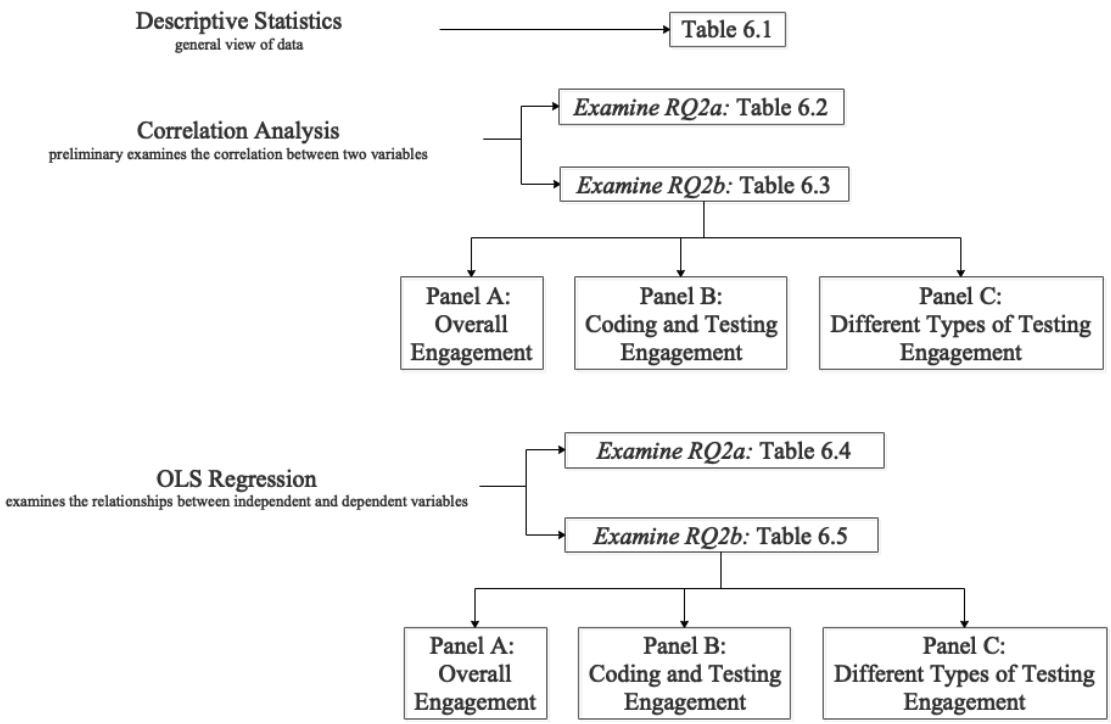


Figure 6.1 Connection Between Statistical Methods and Hypotheses

6.1.1 Descriptive Statistics

The descriptive statistics for the entire sample are shown in Table 6.1. The mean value of *SEQ* (0.321) implies that 76 students in my sample employ TDD behaviors, and the remaining 161 students use non-TDD behaviors. The mean value of the *Cycle* is 2.422 and the standard deviation is nearly 2, which means students perform similarly in terms of the development cycle.

From the perspective of commit frequency, *PROD_FEQ* and *TEST_FEQ* have similar mean values and standard deviations, which means that participants have similar commit frequency on testing codes and production codes. The mean value of *NEW_TEST_COMMITS* is higher than *MAINTAIN_TEST_COMMITS*, suggesting that students have higher commit numbers on maintaining existing testing codes.

The mean value of *CC* is 1.886, indicating the simplicity of the tasks. In terms of control variables, the mean value of *lan_OO* is 0.886, indicating that 210 students selected an object-oriented programming language. However, the minimum to maximum value for *LOC* is hugely volatile, which is consistent with the associated standard deviation (55.592), suggesting that their

performance might be influenced by programming skills. Each student in my study chose to either employ only TDD behaviors or non TDD behaviors throughout the development process of completing the task. As a result, each student had only one CC and cycle value, which were determined by the latest commit.

Variable	Obs	Mean	Std. Dev.	Min	Max	Skewness	Kurtosis
SEQ	237	0.321	0.468	0	1	0	0
Cycle	237	2.422	2.081	0	15	0	0
TC	237	1.181	2.368	0	15	0	0
GC	237	1.241	1.224	0	6	0	0.002
NUMBER_COMMITS	237	2.146	0.664	0	4.111	0.152	0.295
PROD_COMMITS	237	1.483	0.704	0	3.135	0.094	0.598
TEST_COMMITS	235	1.333	0.765	0	3.689	0.882	0.327
NEW_TEST_COMMITS	237	2.422	2.081	0	15	0	0
MAINTAIN_TEST_COMMITS	237	2.591	3.325	-1	30	0	0
N_M	237	0.658	0.475	0	1	0	.
FEQ	237	0.646	0.82	-1.846	2.996	0.402	0.851
PROD_FEQ	237	-0.017	0.845	-2.251	2.442	0.664	0.756
TEST_FEQ	235	-0.177	0.934	-2.944	2.14	0.252	0.571
NEW_TEST_FEQ	237	0.626	0.631	0	4	0	0
MAINTAIN_TEST_FEQ	237	0.621	0.838	-0.125	5.5	0	0
CC	210	1.886	0.778	0	3.807	0	0.67
MI	237	4.011	0.282	3.072	4.69	0.029	0.876
HD	237	61.841	33.81	2	220.143	0	0
lan_OO	237	0.886	0.318	0	1	0	0
LCA_DAG	237	0.392	0.489	0	1	0.006	.
LOC	237	99.684	55.592	5	303	0	0.099

Table 6.1 Descriptive Statistics

6.1.2 Correlation Analysis

In this section, I use the Pearson correlation matrix, a statistical tool that gauges the linear correlation between two variables (Benesty et al., 2009), to examine if the associations and the corresponding directions are consistent with my expectations. Table 6.2 presents a summary of the pairwise correlations for my key variables of interest, TDD behaviors, engagement level in development activity, and maintainability.

Variables	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
SEQ	1							
Cycle	0.426*	1						
NUMBER_COMMITS	0.236*	0.638*	1					
PROD_COMMITS	0.157*	0.559*	0.888*	1				
TEST_COMMITS	0.300*	0.614*	0.877*	0.584*	1			
FEQ	0.294*	0.399*	0.567*	0.478*	0.530*	1		
PROD_FEQ	0.231*	0.352*	0.504*	0.600*	0.327*	0.924*	1	
TEST_FEQ	0.343*	0.412*	0.525*	0.279*	0.679*	0.920*	0.721*	1

*** $p<0.01$, ** $p<0.05$, * $p<0.1$

Table 6.2 Correlation Between TDD and Engagement

Table 6.2 shows a positive association between the writing test sequence (*SEQ*) and various measures of engagement in development activity, such as *NUMBER_COMMITS, PROD_COMMITS,* and *TEST_COMMITS*. Moreover, significant positive relationships are also observed between *SEQ* and *FEQ*, *TEST_FEQ* and *PROD_FEQ* at the 1% significance level. The results suggest that using TDD behaviors is positively related with engagement in development activity, proxied by commit number and commit frequency. These results suggest that using Test-Driven Development (TDD) practices is associated with increased engagement in software development activities.

I noticed that *PROD_COMMITS, TEST_COMMITS*, and *NUMBER_COMMITS* are strongly correlated with each other (0.888) at the 1% significance level, which indicates that they are highly correlated, but this is not an issue since they are not in the same model. I also noticed that *TEST_COMMITS* and *NEW_TEST_COMMITS* or *MAINTAIN_TEST_COMMITS* are strongly related to each, but not surprisingly, as *NEW_TEST_COMMITS* and *MAINTAIN_TEST_COMMITS* are part of *TEST_COMMITS*.

Similarly, it is the same case for *TEST_FEQ, NEW_TEST_FEQ*, and *MAINTAIN_TEST_FEQ*. However, this does not affect my forthcoming analysis, as I separately regress the variables to alleviate any potential multicollinearity issues. Table 6.2 corresponds to the Table 6.4.

Table 6.3 panel A, B and C report the correlations between engagement in development activity and maintainability. I find a positive relationship between *NUMBER_COMMITS, PROD_COMMITS*, and *TEST_COMMITS* with *MI*, and negatively correlated to *CC*. In panel B, I find that *FEQ, PROD_FEQ,* and *TEST_FEQ* are negatively correlated with *CC*, which initially confirms my expectation that the engagement is positively associated with maintainability. In panel C, I find that maintainability is correlated with creating new test cases (*NEW_TEST_COMMITS and NEW_TEST_FEQ*), which means the generation of novel test cases yields a more favorable outcome in terms of code maintainability when compared to the persistent upkeep and modification of extant test code. Overall, the correlation results initially confirm my proposition. The three panels of Table 6.3 correspond to the three panels of Table 6.5.

Variables	(1)	(2)	(3)	(4)
NUMBER_COMMITS	1			
FEQ	0.567*	1		
CC	-0.316*	-0.197*	1	
MI	0.287*	0.081	-0.650*	1

*** $p<0.01$, ** $p<0.05$, * $p<0.1$

Table 6.3 Panel A. Correlation Between Engagement and Maintainability (General)

Variables	(1)	(2)	(3)	(4)	(5)	(6)
PROD_COMMITS	1					
TEST_COMMITS	0.584*	1				
PROD_FEQ	0.600*	0.327*	1			
TEST_FEQ	0.279*	0.679*	0.721*	1		
CC	-0.295*	-0.279*	-0.189*	-0.187*	1	
MI	0.234*	0.283*	0.048	0.107	-0.650*	1

*** $p<0.01$, ** $p<0.05$, * $p<0.1$

Table 6.3 Panel B. Correlation Between Engagement and Maintainability (Production Code vs. Test Code)

Variables	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
PROD_COMMITS	1							
NEW_TEST_COMMITS	0.559*	1						
MAIN_TEST_COMMITS	0.365*	0.294*	1					
PROD_FEQ	0.600*	0.352*	0.208*	1				
NEW_TEST_FEQ	0.271*	0.579*	0.108	0.680*	1			
MAINTAIN_TEST_FEQ	0.259*	0.198*	0.737*	0.444*	0.421*	1		
CC	-0.295*	-0.385*	-0.1	-0.189*	-0.274*	-0.097	1	
MI	0.234*	0.359*	0.108	0.048	0.186*	0.052	-0.650*	1

*** $p<0.01$, ** $p<0.05$, * $p<0.1$

Table 6.3 Panel C. Correlation Between Engagement and Maintainability (New Test vs. Maintain Test)

I also report the correlation between response variables in this study because I am also intrigued by the potential correlation between the dependent variables. However, it is crucial to note that the presence or absence of such correlation will not affect the regression analysis, given that each dependent variable is constructed from distinct regressions.

6.1.3 Regression Models

Our main results are presented in Table 6.4 and Table 6.5. The p-values, enclosed in square brackets underneath the coefficients, and the effect in these tables are beta values. I adopt the ordinary least square (OLS) model to examine the relationships between the TDD behaviors, engagement level in development activities and maintainability respectively. I show the standard error in brace and p-

value in brackets, which is applied Bonferroni correction of False Discovery Rate to avoid type-I errors.

6.1.3.1 Association between TDD behaviors and engagement level in development activities

Table 6.4 shows the OLS regression results that whether TDD behaviors affect engagement level. As shown in Model 1, *Cycle* has a significantly positive correlation with *NUMBER_COMMITS*, which means that students following the iterative development style generate more commits, which indicates are more likely to get involved. As shown in Models 2 and 3, *Cycle* has a strongly significant positive correlation with *PROD_COMMITS* and *TEST_COMMITS*. *SEQ* is negatively related to *PROD_COMMITS*, but only significant at the 10% level and the coefficient is -0.15 which is not economically significant.

Model 4 shows that *SEQ* and *Cycle* are all positively and statistically significant with *FEQ* and *TEST_FEQ* at the 1% level, which indicates that students following the test first order and the iterative development style have higher update commit frequency and commit frequency related to testing, which means that they are more involved. Prior theoretical work proposes that TDD behaviors promote developers to focus on the unit testing activities, and my empirical result provides some evidence. Unexpectedly, I do not find a significant relationship between *SEQ* and *PROD_FEQ*. But *PROD_FEQ* is positively significant at the 1% level with *Cycle*, implying that iterative development style is positively related to commit frequency.

Table 6.4 indicates a negative correlation between *SEQ* and *PROD_COMMITS*, albeit the correlation is economically negligible. Conversely, cycle exhibits a positive correlation with all response variables, while *SEQ* displays a positive correlation with *FEQ* and *TEST_FEQ*. Thus, table 6.4 supports my hypothesis (*H1*) that following TDD behaviors is positively related to engagement level in development activities.

$$Engagement = \alpha + \beta_1 * SEQ + \beta_2 * Cycle + \beta_3 * Control\ Variables + \varepsilon$$

	(1)	(2)	(3)	(4)	(5)	(6)
	NUMBER_COM	PROD_COM	TEST_COM	FEQ	PROD_FEQ	TEST_FEQ

Cycle	0.210***	0.203***	0.219***	0.132***	0.126***	0.146***
	{0.026}	{0.026}	{0.028}	{0.028}	{0.028}	{0.032}
	[0.000]	[0.000]	[0.000]	[0.000]	[0.000]	[0.000]
SEQ	-0.063	-0.150*	0.079	0.264**	0.178	0.411***
	{0.077}	{0.081}	{0.091}	{0.129}	{0.130}	{0.140}
	[0.416]	[0.067]	[0.383]	[0.042]	[0.175]	[0.004]
Constant	1.659***	1.039***	0.773***	0.241***	-0.379***	-0.667***
	{0.066}	{0.071}	{0.074}	{0.074}	{0.079}	{0.086}
	[0.000]	[0.000]	[0.000]	[0.001]	[0.000]	[0.000]
VIF	1.22	1.22	1.22	1.22	1.22	1.22
Observations	237	237	235	237	237	235
R-squared	0.409	0.320	0.379	0.178	0.132	0.204

Robust standard errors in braces
Robust p-val in brackets
*** p<0.01, ** p<0.05, * p<0.1

Table 6.4 Association Between TDD and Engagement

The reason that I separately regress the *TC* and *GC*, as well as *Cycle*, is due to the severe multicollinearity issue for these variables that are observed from the correlation matrix (VIF value is 5.99). However, this result is reasonable since *Cycle* is the sum of *GC* and *TC*. My result shows *TC, GC, and SEQ* are all positively related to engagement level in development activities. These results also jointly suggest that using test-first approach is positively related to engagement level. The detail can be seen below.

	1	2	3	4	5	6
Variables	NC	NPC	NTC	FEQ	FP	FT
SEQ	0.382***	0.227	0.613***	0.603***	0.448**	0.885***
	[0.005]	[0.111]	[0.000]	[0.003]	[0.030]	[0.000]
GC	0.360***	0.331***	0.400***	0.247***	0.218***	0.308***
	[0.000]	[0.000]	[0.000]	[0.000]	[0.001]	[0.000]
TC	0.162***	0.163***	0.163***	0.096***	0.097***	0.097***
	[0.000]	[0.000]	[0.000]	[0.000]	[0.000]	[0.001]
Constant	1.386***	0.807***	0.439***	0.033	-0.546***	-0.964***
	[0.000]	[0.000]	[0.000]	[0.786]	[0.000]	[0.000]
Observations	237	237	235	237	237	235
R-squared	0.462	0.354	0.435	0.198	0.144	0.234

Robust pval in brackets
*** p<0.01, ** p<0.05, * p<0.1

Appendix Table 6.4-A Association Between Development Type and Engagement

6.1.3.2 Association between the engagement level in development activities and maintainability

In Model 1 of Table 6.5 Panel A, I first examine the correlation between overall engagement in development and maintainability. In this Panel, *NC* is negatively significant at the 1% level with *CC* (with a p-value of 0.000), and *NC* is positively significant with *MI* (with a p-value of 0.000) in Model 2, indicating a positive relationship with maintainability. However, I do not find statistical significance between *FEQ,* and *CC* or *MI*. This might be caused by different development style; some students prefer update in a longer period, and some prefer to finish the task in a short period. This can give us some sense that the students who left more commits and engaging in a long-term stability development style has positive relationship with software maintainability. Table 6.5 Panel A answers my research question (*RQ2b*) that engagement in development activity is positively related to maintainability.

$$Maintainability = \alpha + \beta_1 * NUNMBER\ COMMITS + \beta_2 * FEQ + \beta_3 * Control\ Variables + \varepsilon$$

VARIABLES	(1) CC	(2) MI
NUMBER_COMMITS	-0.388***	0.172***
	{0.084}	{0.034}
	[0.000]	[0.000]
FEQ	-0.057	-0.034
	{0.072}	{0.022}
	[0.428]	[0.129]
HD	0.001	-0.001
	{0.002}	{0.001}
	[0.623]	[0.279]
lan_OO	0.005	-0.013
	{0.153}	{0.036}
	[0.972]	[0.708]
DAG	-0.063	0.060**
	{0.104}	{0.027}
	[0.544]	[0.030]
loc	0.006***	-0.003***
	{0.001}	{0.000}
	[0.000]	[0.000]
Constant	2.121***	4.014***
	{0.000}	{0.000}
	[0.000]	[0.000]
VIF	1.68	1.72
Observations	210	237
R-squared	0.274	0.538

Robust standard errors in braces
Robust p-val in brackets
*** p<0.01, ** p<0.05, * p<0.1

Table 6.5 Panel A. Engagement of Development Process and Maintainability

In Table 6.5 Panel B, I further examine the correlation between different phases of engagement (coding and testing) and maintainability. Models 3 to 6 examine whether engagement level in either coding phase (*PROD_COMMITS*, *PROD_FEQ*) or testing phase (*TEST_COMMITS*, *TEST_FEQ*) can affect maintainability. I find a strongly negative significance between testing codes (*TEST_COMMITS*, *TEST_FEQ*) and *CC*, and a strongly positive relationship with *MI*. These results indicate that engaging in testing phase (generating more commits related to testing and higher updating commits frequency) is positively related to maintainability. Furthermore, *PROD_COMMITS* is significantly positive with *MI* at the 5% level, and with *CC* at the 1% level in Models 3 and 4. Interestingly, unlike the positive association of the effort put in *PROD_COMMITS*, I do not find the statistical significance between *PROD_FEQ* and maintainability, which suggests that the effort put in the commit frequency of the coding phase does not influence the maintainability. Table 7 Panel B verifies my hypothesis (*H2*) that the engagement in coding and testing activity is positively related to maintainability but engaging in testing activities is more worth trying.

$$Maintainability = \alpha + \beta_1 * PROD\ COMMITS + \beta_2 * TEST\ COMMITS + \beta_3 * PROD\ FEQ + \beta_4 * TEST\ FEQ + \beta_5 * Control\ Variables + \varepsilon$$

	(3)	(4)	(5)	(6)
VARIABLES	CC	MI	CC	MI
PROD_COMMITS	-0.215***	0.040**		
	{0.073}	{0.019}		
	[0.003]	[0.032]		
TEST_COMMITS	-0.246***	0.114***		
	{0.067}	{0.022}		
	[0.000]	[0.000]		
PROD_FEQ			-0.104	-0.022
			{0.075}	{0.021}
			[0.170]	[0.292]
TEST_FEQ			-0.141**	0.067***
			{0.069}	{0.020}
			[0.043]	[0.001]
HD	0.002	-0.001*	0.003	-0.001**
	{0.002}	{0.000}	{0.002}	{0.000}
	[0.253]	[0.061]	[0.191]	[0.035]
lan_OO	-0.041	-0.014	0.051	-0.034
	{0.152}	{0.034}	{0.158}	{0.034}
	[0.789]	[0.676]	[0.747]	[0.314]
LCA_DAG	-0.062	0.049*	0.008	0.023
	{0.100}	{0.025}	{0.103}	{0.027}
	[0.538]	[0.056]	[0.939]	[0.396]
LOC	0.005***	-0.003***	0.004***	-0.003***
	{0.001}	{0.000}	{0.001}	{0.000}

	[0.000]	[0.000]	[0.002]	[0.000]
Constant	1.900***	4.164***	1.184***	4.396***
	{0.182}	{0.056}	{0.158}	{0.035}
VIF	1.72	1.75	1.91	1.94
Observations	208	235	208	235
R-squared	0.285	0.550	0.208	0.445

Robust standard errors in braces
Robust p-val in brackets
*** p<0.01, ** p<0.05, * p<0.1

Table 6.5 Panel B. Engagement in Coding and Testing Phases and Maintainability

$$Maintainability = \alpha + \beta_1 * NTC + \beta_2 * NPC + \beta_3 * Feq + \varepsilon$$

In Table 6.5 Panel C, I examine the correlation between engagement in different types of testing and maintainability. Models 7 to 10 show the relationship with maintainability between engaging in creating new test cases and maintaining existing test case. Panel C indicates that creating a new test case (*NEW_TEST_COMMITS, NEW_TEST_FEQ*) is negatively significant with *CC*, and is positively related to *MI*. Only *MAINTAIN_TEST_FEQ* (one proxy for maintaining an existing test case) is negatively related to *CC*. Interestingly, I find that engaging in creating new test cases and maintaining existing test case activities contributes unequally to maintainability. I suggest that engaging in creating new test cases can bring greater benefits on maintainability.

$$Maintainability = \alpha + \beta_1 * PROD\ COMMITS + \beta_2 * NEW\ TEST\ COMMITS + \beta_3 * MAINTAIN\ TEST\ COMMITS + \beta_4 * PROD\ FEQ + \beta_5 * NEW\ TEST\ FEQ + \beta_6 * MAINTAIN\ TEST\ FEQ + \beta_7 * Control\ Variables + \varepsilon$$

	(7)	(8)	(9)	(10)
VARIABLES	CC	MI	CC	MI
PROD_COMMITS	-0.175**	0.044*		
	{0.076}	{0.022}		
	[0.022]	[0.051]		
NEW_TEST_COMMITS	-0.093**	0.033***		
	{0.044}	{0.010}		
	[0.034]	[0.001]		
MAINTAIN_TEST_COMMITS	-0.047*	0.006		
	{0.026}	{0.004}		
	[0.073]	[0.103]		
PROD_FEQ			-0.064	-0.025
			{0.078}	{0.018}
			[0.408]	[0.182]
NEW_TEST_FEQ			-0.180	0.060**
			{0.139}	{0.029}

			[0.198]	[0.040]
MAINTAIN_TEST_FEQ			-0.150	0.053*
			{0.112}	{0.029}
			[0.183]	[0.069]
N_M	-0.247	0.012	-0.204	0.037
	{0.151}	{0.031}	{0.145}	{0.032}
	[0.103]	[0.709]	[0.163]	[0.257]
HD	0.001	-0.001*	0.001	-0.001*
	{0.002}	{0.000}	{0.002}	{0.000}
	[0.487]	[0.097]	[0.457]	[0.090]
lan_OO	0.055	-0.034	0.067	-0.033
	{0.134}	{0.033}	{0.139}	{0.031}
	[0.682]	[0.307]	[0.632]	[0.288]
LCA_DAG	-0.033	0.037	-0.012	0.032
	{0.094}	{0.025}	{0.101}	{0.028}
	[0.726]	[0.139]	[0.906]	[0.247]
LOC	0.005***	-0.003***	0.005***	-0.003***
	{0.001}	{0.000}	{0.001}	{0.000}
	[0.000]	[0.000]	[0.000]	[0.000]
Constant	2.034***	4.199***	1.596***	4.274***
	{0.203}	{0.057}	{0.221}	{0.055}
	[0.000]	[0.000]	[0.000]	[0.000]
VIF	1.89	1.85	1.99	2.01
Observations	210	237	210	237
R-squared	0.311	0.538	0.227	0.452

Robust standard errors in braces
Robust p-val in brackets
*** p<0.01, ** p<0.05, * p<0.1

Table 6.5 Panel C. Engagement in New or Maintaining Test and Maintainability

I observed that the ITCV value exceeded that of all other independent variables in each model, indicating that the regression models do not suffer endogeneity problems due to unobserved variables. I also test RESET p-values for all models, which are all smaller than 0.005 implying that the result are not influenced by omitted variables. Although there are more variables are applied in the regression in Panel B and C, but the R2 does not change too much, all around 0.2-0.5.

6.1.4 Additional Analysis

This part is used to further support the result of section 4.5.3. I conduct a bivariate analysis to compare the means of key variables of interest between two groups (Babbie, 2007). If the p-value is significant at the threshold (0.01, 0.05, and 0.1) are often used as the threshold), then there is evidence to reject the null hypothesis that there are equal means in two different groups, in other words, the mean difference among two groups is statistically significant. If the p-value associated with the t-test is greater than the threshold, then do not reject the null hypothesis and indicate no significant difference between the two groups (Boneau, 1960).

I examine the relationship between the number of the development cycle using TDD and without TDD. Table 4.8 reports bivariate results that the students who follow test-first dynamic behavior (*SEQ*) generate more development cycles, higher by 205%. My result provides some initial evidence that students are more likely to improve the iterative speed by following test-first dynamic.

Furthermore, the p-value for the mean difference test for groups of whether following test-first dynamic behavior or not is strongly significant, suggesting that the group following test-first dynamic has significantly higher commit number (*PROD_COMMITS*, *TEST_COMMITS,* and *NUMBER_COMMITS*), and commit frequency (*PROD_FEQ*, *TEST_FEQ,* and *FEQ*). Collectively, the bivariate result confirms the proposition that following test-first dynamic improves engagement in development activity.

	obs1	obs2	Mean1	Mean2	dif	St Err	t value	p value
cycle by SEQ: 0 1	161	76	1.814	3.711	-1.897	0.263	-7.25	0
NC by SEQ: 0 1	161	76	2.039	2.374	-0.335	0.09	-3.7	0
NPC by SEQ: 0 1	161	76	1.407	1.644	-0.237	0.097	-2.45	0.015
NTC by SEQ: 0 1	159	76	1.175	1.664	-0.489	0.102	-4.8	0
FEQ by SEQ: 0 1	161	76	0.48	0.996	-0.515	0.11	-4.7	0
FP by SEQ: 0 1	161	76	-0.15	0.266	-0.416	0.115	-3.65	0.001
FT by SEQ: 0 1	159	76	-0.399	0.286	-0.684	0.122	-5.6	0

Table 6.6 Bivariate Analysis of Sequence

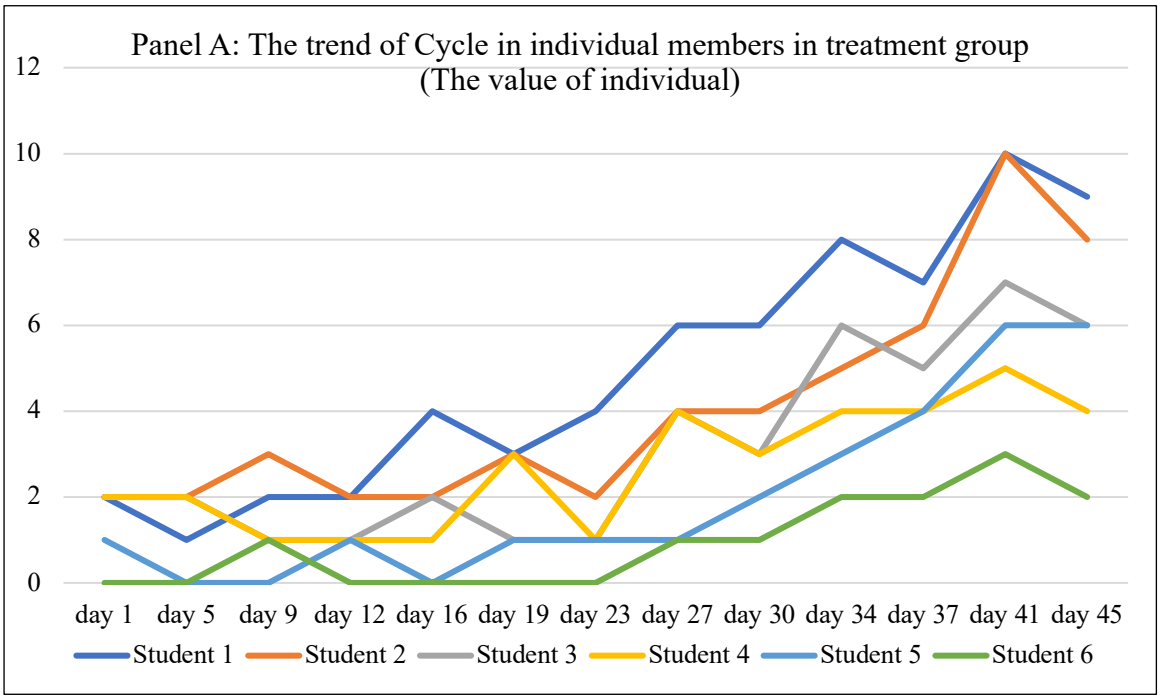
6.2 Result of Group Experiment

In order to assess the influence of gamification on the Test-Driven Development (TDD) behavior and engagement levels in development activity, I apply graphical representations to depict the trend. Subsequently, I examine the association between gamification strategies and maintainability using a regression analysis. For this purpose, three independent variables (gamification_1, gamification_2, and gamification_3) are used to denote gamification intervention, while six dependent variables (CC, CC_mean, CC_o, MI, MI_mean, and MI_o) are employed to measure maintainability. To avoid alternative explanations, two control variables (LOC and CR) that might affect maintainability are also incorporated.

6.2.1 Gamification and Behaviors

Figure 6.2 demonstrates that TDD behaviors are altered by gamification intervention. In Panel A, the higher the behavior score, the more development cycles generated by an individual (students 1, 2, 3 ... 6) in accordance with TDD. The average value for the period is represented by the value of the day. Panel A highlights that gamification intervention increases the behavior score of the individuals, as well as the effectiveness of the intervention is sustained even after the intervention is withdrawn. Although the score of Day 41 drops slightly, this is due to the fact that the number of development cycles has already reached a high level and consequently, it is difficult to maintain improvement. Since the number of participants in the treatment and control groups differ, the number of development cycles is averaged in the analysis.

Panel B in Figure 6.2 assesses the average of the groups. Without gamification treatment, both groups show similar TDD behaviors. However, post-gamification intervention (Day 23), the treatment group outperforms the control group. This comparison provides evidence that gamification strategies are effective. Thus, the results confirm Hypothesis 1 that gamification changes development behaviors to follow TDD.



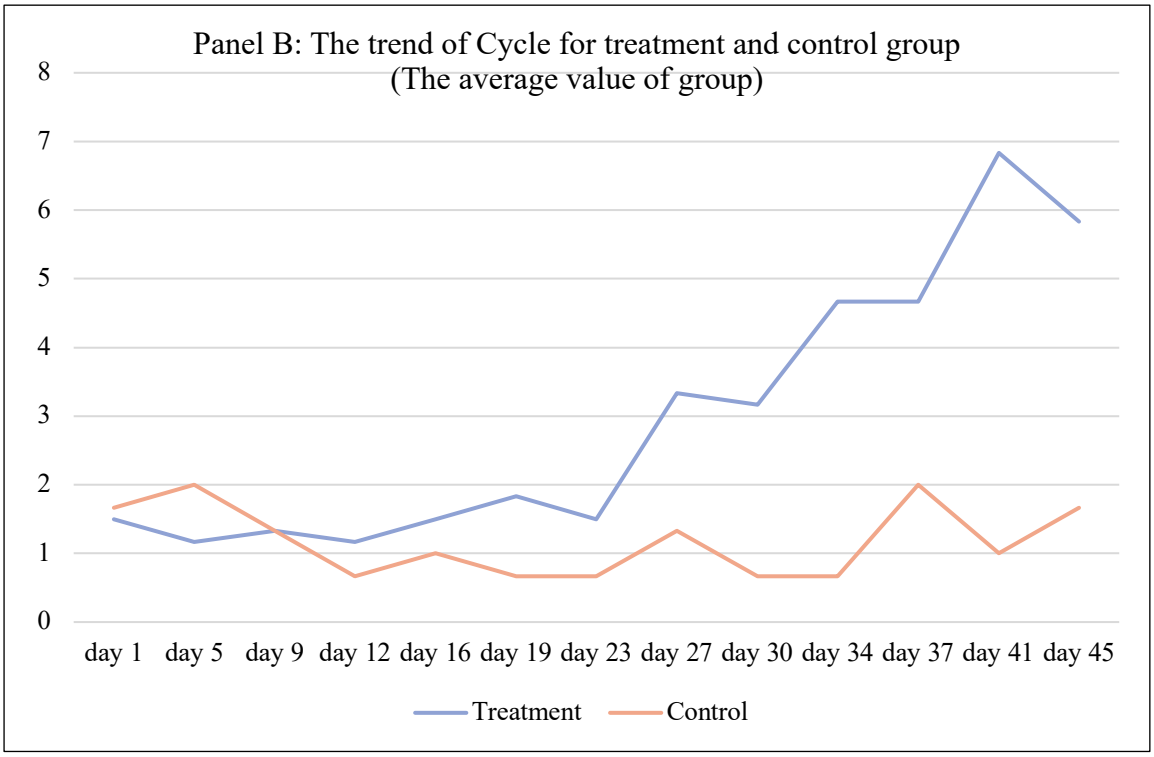
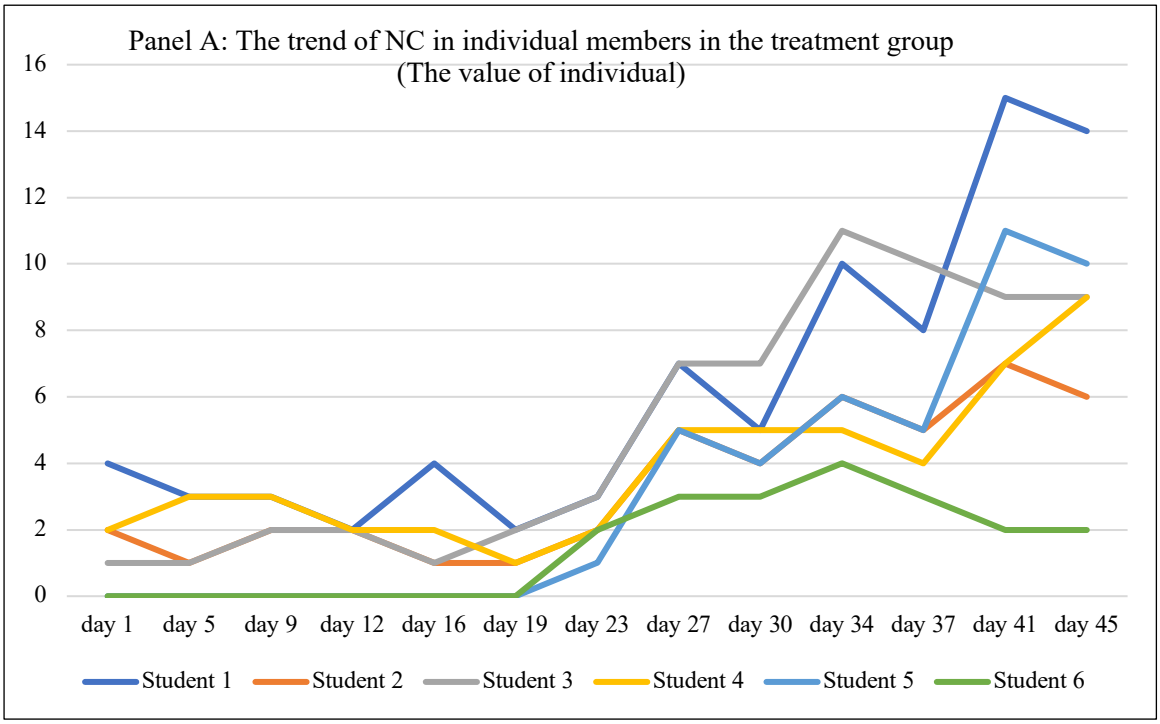


Figure 6.2 Gamification and TDD Behaviors

6.2.2 Gamification and Engagement

Figure 6.3 illustrates the engagement level in development activity from the number of commits (NC) perspective. The average value of NC for each day is displayed in Panel A. Following the

implementation of the gamification intervention on day 23, there is a notable increase in value for all students on day 30. Student 6 is the exception, as he had already completed his tasks prior to day 37. Panel B similarly displays a comparable trend between the treatment and control groups before day 23, however, the treatment group experiences a greater improvement in NC after the gamification intervention. Following the conclusion of the gamification intervention on day 41, NC numbers remain unchanged.



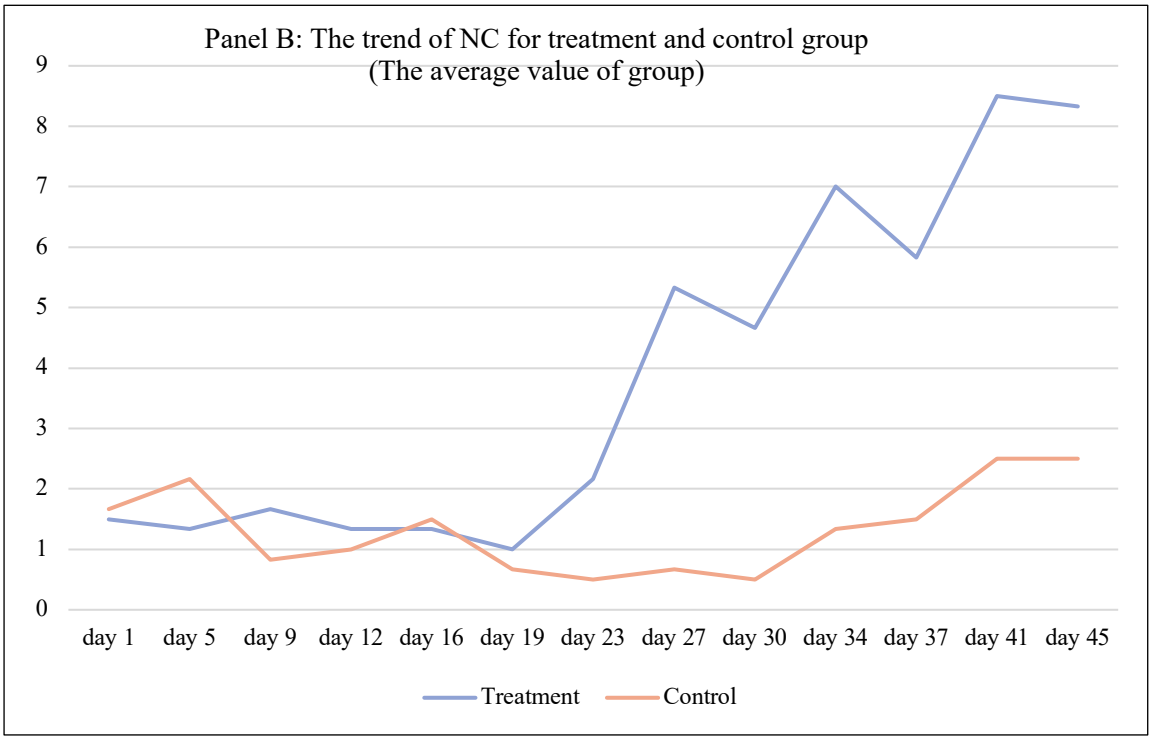
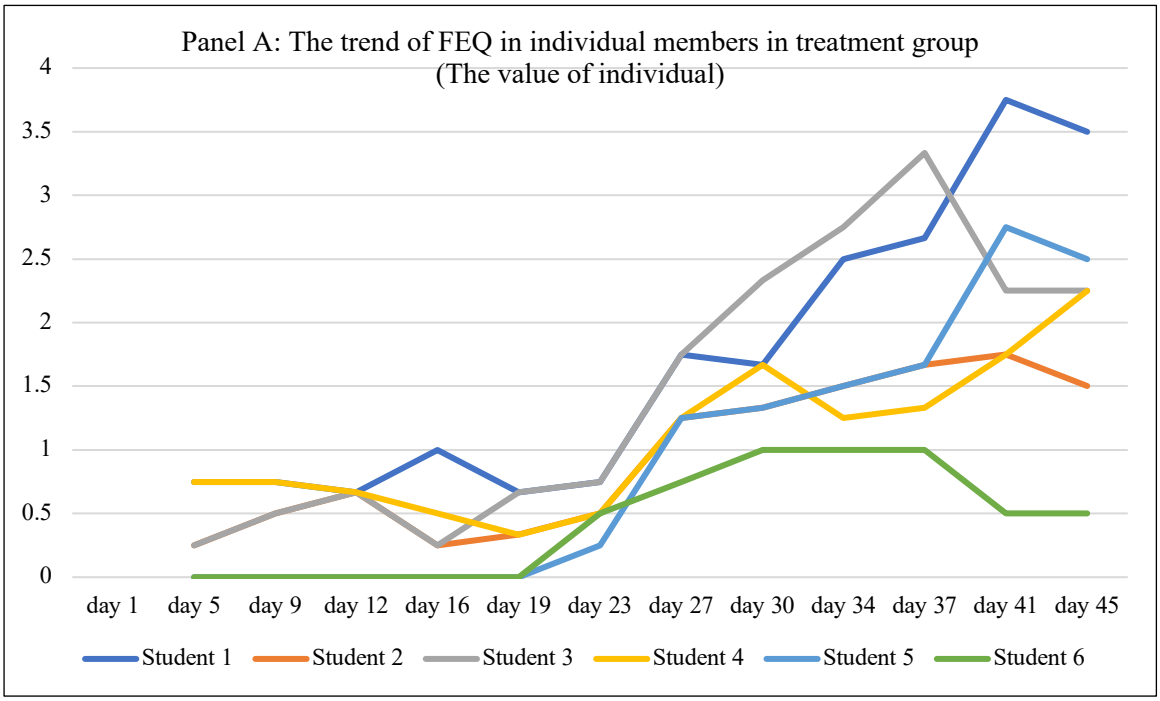


Figure 6.3 Gamification and Engagement (NC)

Figure 6.4 illustrates the engagement level in development activity from a frequency perspective. The point of the day is the average frequency of engagement (FEQ) during the period. For instance, the value of day 5 is the average of FEQ between day 1 to day 5. Panel A of Figure 6.4 indicates

that an individual's FEQ value increased after the gamification intervention. To further evaluate the frequency changes after the implementation of the gamification intervention, I compare the average of the team's frequency between the treatment and the control group, as demonstrated in Panel B of Figure 6.4. From day 1 to day 23, the treatment group and control group display a similar trend, while the treatment group displays superior performance after the gamification strategy was introduced (after day 23), suggesting that the gamification strategy has a positive impact on improving engagement level. Additionally, the FEQ in the treatment group remained at a similar level even after the termination of the gamification intervention (day 41), indicating that the effectiveness of gamification intervention can be sustained for an extended period. This further substantiates Hypothesis 1, that gamification is beneficial in altering behavior towards Test-Driven Development (TDD) and increasing engagement levels.



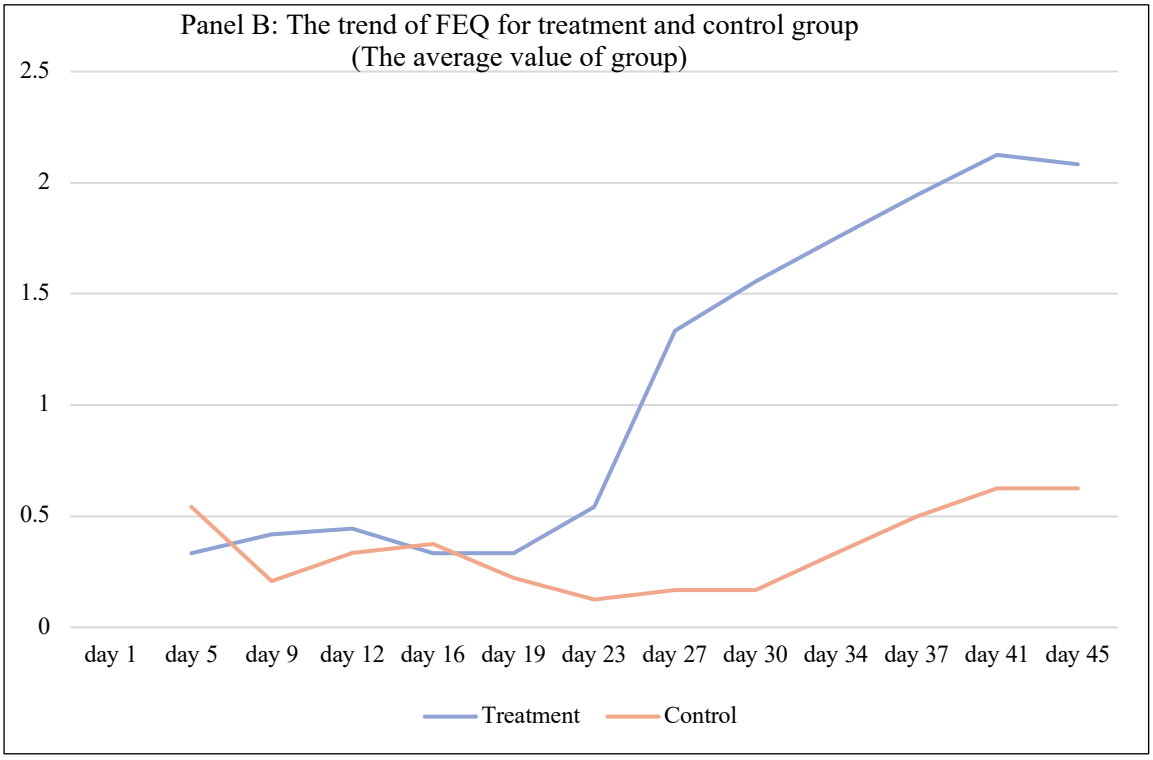


Figure 6.4 Gamification and Engagement (FEQ)

6.2.3 Gamification and Maintainability

The observations tested spanned the entire experimental period of the treatment group, approximately 45 days. Descriptive statistics for the entire sample are presented in Table 6.7. The

mean values of gamification_1 (0.288), gamification_2 (0.456), and gamification_3 (0.628) indicate that 72, 114, and 157 files in my sample employed gamification intervention, with the remaining files not receiving gamification. For the control variable, the range of values for LOC was wide, in accordance with the associated standard deviation (83.994), indicating the files were varied.

Variable	Obs	Mean	Std. Dev.	Min	Max
CC	184	1.623	1.086	0	3.367
CC mean	184	1.623	1.086	0	3.367
CC o	250	4.704	7.52	0	26
MI	250	89.475	16.254	54.563	134.967
MI mean	250	89.475	16.254	54.563	134.967
MI o	250	84.475	16.697	54.563	134.967
gami1	250	0.288	0.454	0	1
gami2	250	0.456	0.499	0	1
gami3	250	0.628	0.484	0	1
loc	250	104.388	83.994	5	525
comment ratio	250	9.551	9.421	0	47.04

Table 6.7 Descriptive Statistics

Table 6.8 presents the results of my analysis on the correlation between gamification and maintainability. In Panel A, Models 1-6 examine the correlation between gamification and Cyclomatic Complexity (CC) in the short term. Contrary to expectation, a significant positive correlation was found between gami_1, gami_3, and CC in Models 1 and 3, and a significant positive correlation between gami_1, gami_3, and CC_mean in Models 4 and 6. This suggests that gamification leads to higher CC in the short term and worse maintainability. However, a strongly significant negative correlation between gamification_1, gamification_2, and CC_o in Models 7 and 8 was found, indicating that gamification leads to lower CC in the long term and better maintainability. These results support my hypothesis (H2) that gamification is positively related to software quality in the long term. Furthermore, they indicate that gamification is negatively related to maintainability in the short term.

$$Maintainability = \alpha + \beta_1 * gami\ 1 + \beta_2 * Control\ Variables + \varepsilon$$

$$Maintainability = \alpha + \beta_1 * gami\ 2 + \beta_2 * Control\ Variables + \varepsilon$$

$$Maintainability = \alpha + \beta_1 * gami\ 3 + \beta_2 * Control\ Variables + \varepsilon$$

	1	2	3	4	5	6	7	8	9
VARIABLES	CC_ln	CC_ln	CC_ln	CC_mean	CC_mean	CC_mean	CC_o	CC_o	CC_o
gami1	0.116*			0.116*			-0.000**		

	[0.055]			[0.055]			[0.022]		
gami2		0.091			0.091			-0.000**	
		[0.101]			[0.101]			[0.049]	
gami3			0.100*			0.100*			0
			[0.082]			[0.082]			[0.966]
loc	0.006***	0.007***	0.006***	0.006***	0.007***	0.006***	0.000***	0.000***	0.000***
	[0.000]	[0.000]	[0.000]	[0.000]	[0.000]	[0.000]	[0.000]	[0.000]	[0.000]
comment_ratio	0.041***	0.040***	0.039***	0.041***	0.040***	0.039***	0.000***	0.000***	0.000***
	[0.000]	[0.000]	[0.000]	[0.000]	[0.000]	[0.000]	[0.002]	[0.004]	[0.001]
Constant	0.416***	0.407***	0.400***	0.416***	0.407***	0.400***	2.000***	2.000***	2.000***
	[0.003]	[0.005]	[0.006]	[0.003]	[0.005]	[0.006]	[0.000]	[0.000]	[0.000]
Observations	184	184	184	184	184	184	250	250	250
R-squared							1	1	1
Number of group	30	30	30	30	30	30			

*** p<0.01, ** p<0.05, * p<0.1

Table 6.8 Panel A Gamification and Maintainability in Short Term

In Panel B, Models 1-6 examine the correlation between gamification and Maintainability Index (MI) in the long term. Contrary to expectation, a strongly significant negative correlation was found between gami_1, gami_3, and MI in Model 1, and gami_3 and MI_mean in Model 3. This suggests that gamification leads to lower MI in the short term and worse maintainability. However, a strong correlation in the long term was found, which showed a positive association between gami_1 and MI_o that was significant at the 5% level and a positive association between gami_1 and MI_o that was significant at the 10% level. Additionally, a positive association between gami_1, gami_3, and MI_d was found, which suggests that gamification can accelerate the improvement of MI. In conclusion, Table 6.8 verifies my hypothesis (H2) that gamification is positively related to maintainability in the long term and accelerates the improvement of maintainability.

	1	2	3	4	5	6	7	8	9
VARIABLES	MI	MI	MI	MI_mean	MI_mean	MI_mean	MI_o	MI_o	MI_o
gami1	-0.766			-0.766			1.825**		
	[0.162]			[0.162]			[0.015]		
gami2		-0.760			-0.760			0.899	
		[0.129]			[0.129]			[0.267]	
gami3			-1.119**			-1.119**			1.792*
			[0.029]			[0.029]			[0.062]
loc	-0.122***	-0.123***	-0.122***	-0.122***	-0.123***	-0.122***	-0.137***	-0.136***	-0.137***
	[0.000]	[0.000]	[0.000]	[0.000]	[0.000]	[0.000]	[0.000]	[0.000]	[0.000]
CR	-0.494***	-0.486***	-0.485***	-0.494***	-0.486***	-0.485***	-0.483***	-0.486***	-0.482***
	[0.000]	[0.000]	[0.000]	[0.000]	[0.000]	[0.000]	[0.001]	[0.001]	[0.001]
Constant	108.349***	108.503***	108.740***	108.349***	108.503***	108.740***	102.815***	102.910***	102.207***
	[0.000]	[0.000]	[0.000]	[0.000]	[0.000]	[0.000]	[0.000]	[0.000]	[0.000]
Observations	250	250	250	250	250	250	250	250	250
R-squared							0.590	0.588	0.590
Number of group	40	40	40	40	40	40			

*** p<0.01, ** p<0.05, * p<0.1

Table 6.8 Panel B Gamification and Maintainability in Long Term

The remaining results of my analysis show that all models employed do not have multicollinearity problems, with the mean VIF at 1.3 to 2. The ITCV value is greater than all other remaining independent variables in each model, and all RESET values are greater than 0.05. Thus, my results are robust and do not suffer endogeneity problems.

6.3 Result of Individual Experiment

In order to investigate the impact of gamification on encouraging students to adopt and sustain Test-Driven Development (TDD) practices for the improvement of software maintainability, I utilize several proxies to measure relevant variables. To evaluate students' adherence to TDD, I use the number of development cycles and the number of test cases. The level of engagement is gauged by the number of commits. To quantify software maintainability, I employ cyclomatic complexity and maintainability index as proxies.

In addition, I aim to examine the hypothesis that the combined effects of utilizing various gamification strategies may differ from the effects of using them individually. Thus, I differentiate the effects of three gamification strategies, namely points, leaderboard, and feedback. Finally, I assess the third hypothesis that the impact of gamification endures even after its withdrawal, by examining a single variable, *Sustain*.

6.3.1 Gamification and TDD Practice (RQ3)

Figure 6.5 and Figure 6.6 demonstrate the impact of gamification on students' adherence to Test-Driven Development (TDD) practices. Figure 6.5 presents the changes in the number of development cycles observed in the treatment group. The value on each axis of the graph represents the total number of development cycles throughout the gamification treatment period. For instance, when the x-axis value is equal to 1, the y-axis value is 34, indicating the number of development cycles generated between the initiation of gamification (Day 1) and the first release of gamification treatment (Day 4). Similarly, Figure 6.6 depicts the variations in the number of test cases in the treatment group. The graphs reveal a consistent increase in both the number of development cycles and test cases in the treatment group.

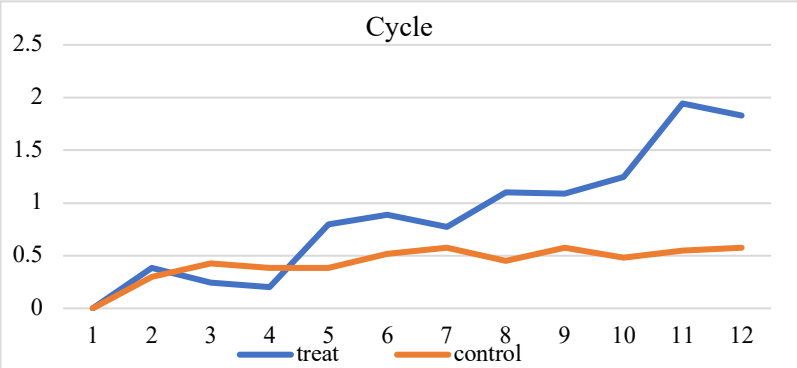


Figure 6.5 Gamification Impact on Number of Development Cycle

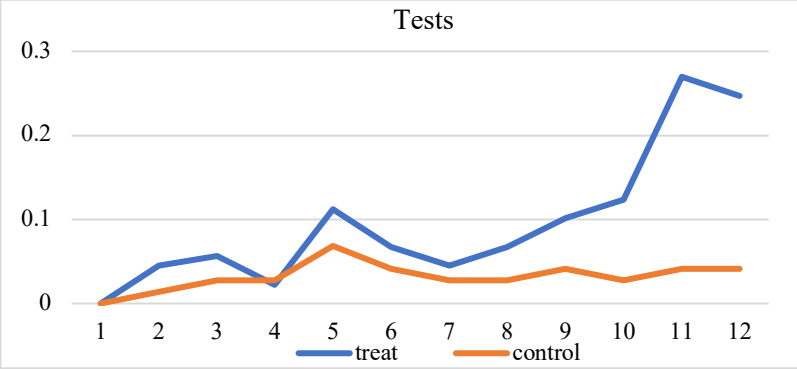


Figure 6.6 Gamification Impact on Number of Test Case

Figure 6.7 presents the impact of gamification on TDD practices. It depicts the transformation of students' development behavior towards adherence to TDD, the enhancement of engagement levels in development activities, and the improvement of software maintainability. To better illustrate the influence of gamification on the dependent variables, the treatment and control groups are analyzed in two steps. First, the ratio between the treatment and control groups is calculated and then the log value of the ratio is taken. A value higher than 0 indicates that the treatment group has a higher value for the corresponding variable. For instance, if the value of "Cycle" is greater than 0, it suggests that the treatment group generates more development cycles than the control group.

The results depicted in Figure 6.7 support the first hypothesis (H1), which states that gamification can alter students' development behavior towards TDD. This is evidenced by the increase in the number of development cycles and test cases in the treatment group. Additionally, gamification treatment improves the engagement level in development activities as indicated by higher values of

NTC, NPC, and *NC*. Finally, gamification also enhances software maintainability by increasing *MI, FM,* and *FC* and lowering *CC*.

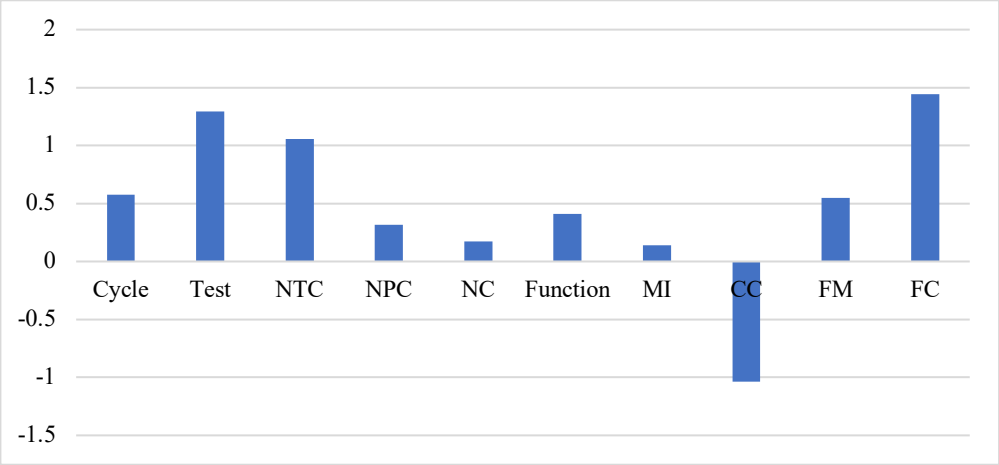


Figure 6.7 Gamification Impact on TDD Practice

The objective of the present study is to assess the effect of gamification on the students' adherence to Test-Driven Development (TDD) practice. In order to do so, bivariate analysis and Ordinary Least Squares (OLS) regression were utilized. The bivariate analysis was employed to compare the means of relevant variables between two groups (Babbie, 2007). When the p-value is significant at the commonly used threshold levels (0.01, 0.05, and 0.1), it suggests that there is a statistically significant difference in the means between two groups and rejection of the null hypothesis of equal means in two groups. However, if the p-value is greater than the threshold, the null hypothesis of equal means is not rejected and no significant difference is indicated (Boneau, 1960).

I adopt the OLS model to examine the relationships between gamification and TDD practice, and the results also answer RQ3. P-values are in square brackets below the coefficients. Model 1-2 of Table 6.9 Panel A examines the correlation between *gamification* and TDD behaviors. Model 3-5 of Panel A shows that gamification had a positive correlation with engagement level in development activities. I find a significant positive correlation between *gamification* and *Function, MI, and FM* in Models 6, 7, and 9, and a significant negative correlation between *gamification* and *CC* in Model 8, which means that gamification leads to better software maintainability. Panel A answer my RQ3 that gamification can motivate students to develop and maintain TDD practice, including changing students' behaviors to

follow TDD, increasing engagement in development activities, and ultimately improving software maintainability.

$$TDD\ Behaviors = \alpha + \beta_1 * Gamification + \beta_2 * Control\ Variables + \varepsilon$$

$$Engagement = \alpha + \beta_1 * Gamification + \beta_2 * Control\ Variables + \varepsilon$$

$$Maintainability = \alpha + \beta_1 * Gamification + \beta_2 * Control\ Variables + \varepsilon$$

Variables	1 Cycle	2 Test	3 NTC	4 NPC	5 NC	6 MI	7 CC	8 FM	9 FC
Gamification	3.571***	0.733*	1.133**	4.827*	2.623	12.399***	-0.340*	92.015***	0.102
	[0.003]	[0.066]	[0.025]	[0.073]	[0.393]	[0.000]	[0.062]	[0.000]	[0.902]
gender	-0.301	0.392	1.014*	-3.376	-3.179	0.454	-0.144	-11.112	-1.098
	[0.902]	[0.353]	[0.075]	[0.628]	[0.645]	[0.886]	[0.535]	[0.746]	[0.389]
CR	0.021	-0.005	0.004	0.123	0.081	-0.149	-0.009*	-0.231	-0.017
	[0.695]	[0.602]	[0.798]	[0.393]	[0.615]	[0.367]	[0.081]	[0.843]	[0.655]
lan_OO	1.347	-0.323	-0.2	4.586	4.681	-6.383**	-0.006	37.833	1.655
	[0.352]	[0.404]	[0.695]	[0.179]	[0.232]	[0.021]	[0.974]	[0.176]	[0.104]
HD	0.071***	0.007***	0.007	0.128***	0.147***	-0.112***	0	0.570**	0.021*
	[0.000]	[0.002]	[0.123]	[0.000]	[0.000]	[0.010]	[0.849]	[0.029]	[0.068]
loc	-0.001**	0	0	-0.001**	-0.001**	-0.003***	0	-0.015***	0
	[0.023]	[0.397]	[0.787]	[0.047]	[0.043]	[0.000]	[0.135]	[0.001]	[0.448]
use_exp	0.499	0.032	-0.02	1.366	1.43	-1.440*	0.032	5.517	0.393
	[0.218]	[0.762]	[0.876]	[0.179]	[0.177]	[0.087]	[0.613]	[0.352]	[0.201]
Constant	0.534	-0.287	-0.829	6.857	11.097	76.484***	2.383***	79.918*	3.913**
	[0.845]	[0.401]	[0.169]	[0.355]	[0.158]	[0.000]	[0.000]	[0.087]	[0.028]
Observations	162	162	162	162	162	162	162	162	162
R-squared	0.216	0.058	0.062	0.162	0.147	0.391	0.097	0.195	0.098

Robust pval in brackets
*** p<0.01, ** p<0.05, * p<0.1

Table 6.9 Panel A

The relationship between TDD practice with and without gamification treatment was analyzed. The results of the bivariate analysis, as presented in Table 6.10, indicate that students who received gamification treatment showed 162% and 336% increase in development cycles and test cases, respectively. The engagement level in development activity (NTC and NPC) and software maintainability (MI, CC, and FM) in the treatment group was found to be superior to that in the control group. The p-value of the mean difference test between the two groups (with and without gamification treatment) was found to be highly significant, suggesting that the group with gamification treatment showed significantly improved TDD behaviors (more Cycle and Test),

higher engagement level (NTC and NPC), and improved maintainability (MI, CC, and FM). These findings collectively support the hypothesis that gamification plays a positive role in improving TDD practice, including TDD behaviors, engagement level in development activities, and software maintainability.

Gamification	Control	Treatment	Mean1	Mean2	dif	St Err	t value	p value
Cycle	73	89	5.657	9.191	-3.534	1.33	-2.65	0.009
Test	73	89	0.301	1.011	-0.71	0.364	-1.95	0.053
NTC	73	89	0.424	1.472	-1.048	0.458	-2.3	0.024
NPC	73	89	16.698	21.91	-5.212	2.943	-1.75	0.079
NC	73	89	21.767	24.528	-2.761	3.293	-0.85	0.403
Function	73	89	2.082	3.045	-0.963	0.315	-3.05	0.003
MI	73	89	60.148	73.541	-13.393	3.392	-3.95	0
CC	73	89	2.245	1.876	0.369	0.172	2.15	0.033
FM	73	89	117.434	207.554	-90.12	20.039	-4.5	0
FC	73	89	5.357	5.361	-0.003	0.805	0	0.997

Table 6.10 Bivariate Analysis of Gamification Impact on TDD practice

6.3.2 Comparing the Effect of Different Gamification Strategies (RQ4)

Figure 6.8 represents the impact of the use of different gamification strategies on TDD practices. The first pillar represents Group 1, while the second and third pillars depict the ratio of Group 2 and Group 3 to Group 1, respectively. For instance, if the value of the second pillar is greater than 1, this indicates that Group 2 has a higher value for the dependent variable in comparison to Group 1. The figure indicates that the magnitude of the impact on TDD practice increases with the utilization of more gamification strategies.

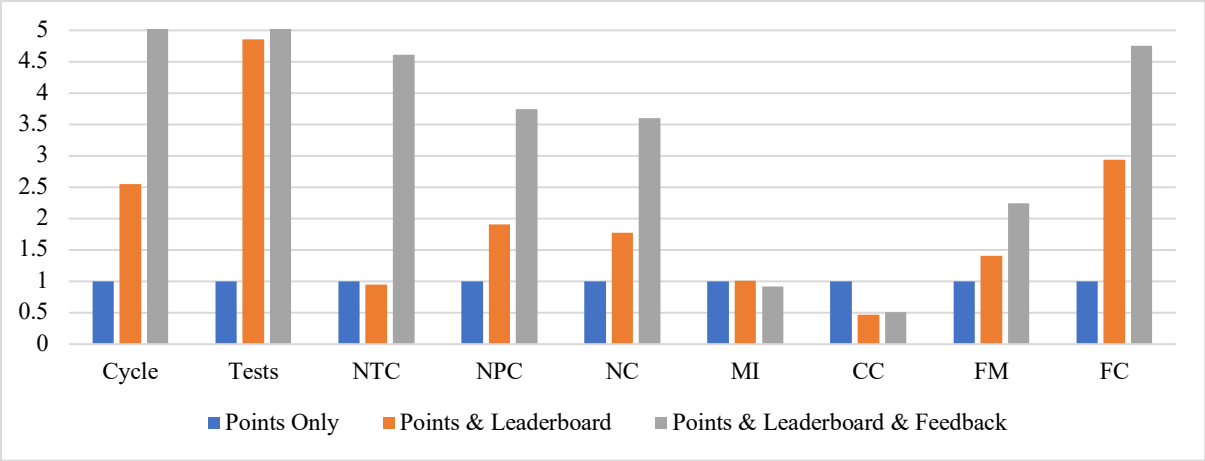


Figure 6.8 Compare Gamification Strategies' Impact on TDD Practice

To further analyze the relationship between different gamification strategies and TDD practice, I present Figure 6.9. To provide a comprehensive view of the results, I process the data in two steps. Firstly, I calculate the ratio between two groups for each variable. Then, I take the logarithm of the ratio. For instance, consider the independent variable Leaderboard and the dependent variable *Cycle*. When *Leaderboard* equals 1, the value of *Cycle* is 6.32 and when Leaderboard equals 0, the value of *Cycle* is 2.48. Thus, the ratio between the two groups is 2.54 (calculated as 6.32 divided by 2.48). The final step is to take the logarithm of the ratio, yielding a value of 0.93 (calculated as the natural logarithm of 2.54). The other values for each gamification strategy are calculated in the same manner.

The results of Figure 6.9 indicate that the combination of leaderboard and points has a more substantial effect on TDD practice compared to using points alone. The *Leaderboard* has a significant impact on maintainability, especially with regards to the number of functions. The results of the independent variable *All* suggest that implementing all gamification strategies, as opposed to using points solely, has a more considerable impact on TDD practice. This conclusion is further strengthened by the results of the independent variable *Extra*, although the outcome is similar. This may suggest that the impact of feedback is limited, and its effects require further examination, as indicated by the results of the independent variable *Feedback*. The results suggest that utilizing feedback as a gamification technique has a more significant impact on TDD practice compared to not using it, albeit to a limited extent. The results of Figure 6.9 provide support for the second hypothesis (H2) that utilizing gamification strategies in combination is more effective than using them individually. The results indicate that the effectiveness of gamification increases with the integration of its strategies. The results of the fifth group indicate that randomly or voluntarily selected gamification elements do not seem to have a significant impact on the effectiveness of gamification. These conclusions are further supported by the results of the bivariate test that is discussed in the following section.

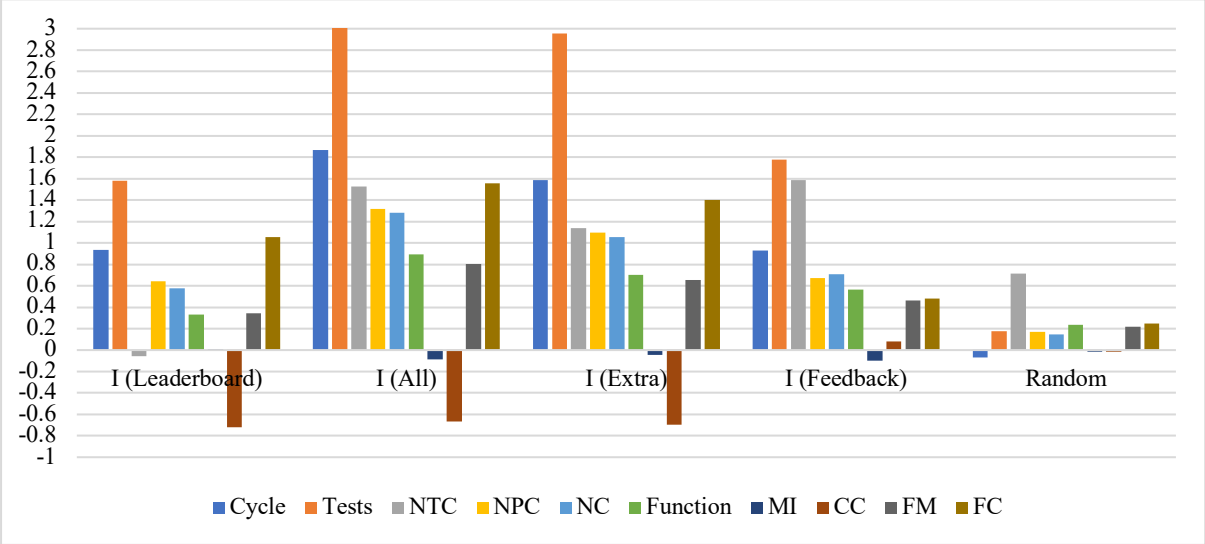


Figure 6.9 Distinguish the Impact of Different Gamification Strategies on TDD Practice

To examine the impact of the utilization of a leaderboard on the TDD practice, a bivariate analysis was performed. The independent variable of *Feedback* was analyzed to determine the differences between utilizing or not utilizing a leaderboard. The results of the bivariate analysis, as presented in Table 6.11, indicated that all dependent variables with the exception of *NTC, function,* and *MI* had better mean values when a leaderboard was combined with points, compared to the use of points alone. For instance, students who received both points and a leaderboard demonstrated 255% and 486% higher development cycles and test cases, respectively, than those who received points only. The results of Tables 6.3 to 6.6, which correspond to the independent variables of *All, Extra, Feedback* and *Random*, respectively, were consistent with the findings of Figure 6. In conclusion, the bivariate results support the hypothesis that utilizing gamification strategies together is more effective than utilizing them individually, and the effectiveness of gamification increases with the application of each successive strategy. The results also suggest that the utilization of randomly or voluntarily selected gamification elements does not seem to impact the gamification's effectiveness.

	Point Only	Leaderboard	Mean1	Mean2	dif	St Err	t value	p value
Cycle by Lead: 0 1	27	25	2.482	6.32	-3.839	1.607	-2.4	0.021
Test by Lead: 0 1	27	25	0.074	0.36	-0.286	0.165	-1.75	0.089
NTC by Lead: 0 1	27	25	0.593	0.56	0.033	0.435	0.05	0.941
NPC by Lead: 0 1	27	25	9.148	17.44	-8.292	3.566	-2.35	0.024
NC by Lead: 0 1	27	25	10.667	18.96	-8.293	3.756	-2.2	0.032
function by Lead: 0 1	27	25	1.778	2.48	-0.702	0.469	-1.5	0.141

MI by Lead: 0 1	27	25	75.9	76.826	-0.927	5.686	-0.15	0.871
CC by Lead: 0 1	27	25	2.322	1.636	0.686	0.205	3.35	0.002
FM by Lead: 0 1	27	25	120.964	179.231	-58.267	30.857	-1.9	0.065
FC by Lead: 0 1	27	25	4.253	4.119	0.134	1.083	0.1	0.902

Table 6.11 Bivariate Analysis on Leaderboard

	Point Only	All	Mean1	Mean2	dif	St Err	t value	p value
Cycle by All: 0 1	27	37	2.482	16.027	-13.546	2.345	-5.8	0
Test by All: 0 1	27	37	0.074	2.135	-2.061	0.86	-2.4	0.019
NTC by All: 0 1	27	37	0.593	2.73	-2.137	1.075	-2	0.051
NPC by All: 0 1	27	37	9.148	34.243	-25.095	5.144	-4.9	0
NC by All: 0 1	27	37	10.667	38.406	-27.739	5.546	-5	0
function by All: 0 1	27	37	1.778	4.351	-2.574	0.582	-4.45	0
MI by All: 0 1	27	37	75.9	69.602	6.298	4.878	1.3	0.202
CC by All: 0 1	27	37	2.322	1.712	0.61	0.186	3.3	0.002
FM by All: 0 1	27	37	120.964	289.879	-168.915	37.531	-4.5	0
FC by All: 0 1	27	37	4.253	7.008	-2.755	1.297	-2.1	0.037

Table 6.12 Bivariate Analysis on All

	Point Only	Extra	Mean1	Mean2	dif	St Err	t value	p value
Cycle by Extra: 0 1	27	62	2.482	12.113	-9.632	2.24	-4.3	0
Test by Extra: 0 1	27	62	0.074	1.419	-1.345	0.69	-1.95	0.054
NTC by Extra: 0 1	27	62	0.593	1.855	-1.262	0.867	-1.45	0.148
NPC by Extra: 0 1	27	62	9.148	27.468	-18.32	4.669	-3.9	0
NC by Extra: 0 1	27	62	10.667	30.564	-19.898	5.021	-3.95	0
function by Extra: 0 1	27	62	1.778	3.597	-1.819	0.522	-3.5	0.001
MI by Extra: 0 1	27	62	75.9	72.515	3.385	4.155	0.8	0.417
CC by Extra: 0 1	27	62	2.322	1.681	0.64	0.173	3.7	0.001
FM by Extra: 0 1	27	62	120.964	245.263	-124.299	33.276	-3.75	0.001
FC by Extra: 0 1	27	62	4.253	5.843	-1.59	1.117	-1.4	0.158

Table 6.13 Bivariate Analysis on Extra

	No Feedback	Feedback	Mean1	Mean2	dif	St Err	t value	p value
Cycle by Feedback: 0 1	25	37	6.32	16.027	-9.707	2.716	-3.55	0.001
Test by Feedback: 0 1	25	37	0.36	2.135	-1.775	0.903	-1.95	0.054
NTC by Feedback: 0 1	25	37	0.56	2.73	-2.17	1.096	-2	0.052
NPC by Feedback: 0 1	25	37	17.44	34.243	-16.803	5.697	-2.95	0.005
NC by Feedback: 0 1	25	37	18.96	38.406	-19.445	6.006	-3.25	0.002
function by Feedback:0 1	25	37	2.48	4.351	-1.871	0.595	-3.15	0.003
MI by Feedback: 0 1	25	37	76.826	69.602	7.225	3.498	2.05	0.043
CC by Feedback: 0 1	25	37	1.636	1.712	-0.076	0.201	-0.4	0.707
FM by Feedback: 0 1	25	37	179.231	289.879	-110.648	37.777	-2.95	0.005

FC by Feedback: 0 1	25	37	4.119	7.008	-2.889	1.27	-2.25	0.026

Table 6.14 Bivariate Analysis on Feedback

	volunteer	random	Mean1	Mean2	dif	St Err	t value	p value
Cycle by Random: 0 1	28	34	12.572	11.736	0.836	2.947	0.3	0.777
Test by Random: 0 1	28	34	1.286	1.53	-0.243	0.918	-0.25	0.791
NTC by Random: 0 1	28	34	1.179	2.412	-1.233	1.103	-1.1	0.269
NPC by Random: 0 1	28	34	24.893	29.588	-4.696	5.979	-0.8	0.435
NC by Random: 0 1	28	34	28.107	32.588	-4.481	6.39	-0.7	0.486
function by Random: 0 1	28	34	3.143	3.97	-0.828	0.624	-1.35	0.19
MI by Random: 0 1	28	34	73.07	72.058	1.012	3.567	0.3	0.777
CC by Random: 0 1	28	34	1.729	1.643	0.087	0.198	0.45	0.663
FM by Random: 0 1	28	34	219.087	266.819	-47.733	39.331	-1.2	0.23
FC by Random: 0 1	28	34	4.78	6.718	-1.938	1.281	-1.5	0.136

Table 6.15 Bivariate Analysis on Random

In order to obtain statistical support for Figure 6.15, I also use the OLS model to evaluate the relationship between gamification strategies and TDD practice, and the results are shown below. P-values are in square brackets below the coefficients. Table A to D examines the correlation between *Leaderboard, All, Extra, Feedback* and TDD practice.

$$TDD\ Practice = \alpha + \beta_1 * Leaderboard + \beta_2 * Control\ Variables + \varepsilon$$

$$TDD\ Practice = \alpha + \beta_1 * All + \beta_2 * Control\ Variables + \varepsilon$$

$$TDD\ Practice = \alpha + \beta_1 * Extra + \beta_2 * Control\ Variables + \varepsilon$$

$$TDD\ Practice = \alpha + \beta_1 * Feedback + \beta_2 * Control\ Variables + \varepsilon$$

In summary, the figures results, bivariate analysis, and OLS regression answer my research question (RQ4): using gamification strategies together better than using them in individually.

VARIABLES	Cycle	Test	NTC	NPC	NC	MI	CC	FM	FC
					Table A				
Leaderboard	3.456**	0.299	-0.235	7.997**	7.988**	-0.603	-0.757***	49.679	-0.464
	[0.028]	[0.152]	[0.615]	[0.037]	[0.043]	[0.868]	[0.001]	[0.104]	[0.620]
gender	0.564	-0.208	0.134	-1.072	-4.104	7.909**	-0.206	-51.096	-2.125
	[0.632]	[0.483]	[0.712]	[0.820]	[0.422]	[0.034]	[0.326]	[0.271]	[0.150]
CR	-0.031	0.001	-0.003	0.040	0.000	-0.108	-0.007	-1.066	-0.024
	[0.397]	[0.876]	[0.786]	[0.759]	[0.998]	[0.346]	[0.339]	[0.334]	[0.460]

lan_OO	-2.093	-0.054	-0.205	-4.561	-4.254	-7.640*	-0.116	-20.175	-0.132
	[0.141]	[0.723]	[0.489]	[0.178]	[0.223]	[0.087]	[0.653]	[0.528]	[0.900]
HD	0.066	0.003	0.033	0.212*	0.248*	-0.476***	0.008	1.464*	0.079***
	[0.258]	[0.510]	[0.159]	[0.052]	[0.053]	[0.000]	[0.176]	[0.097]	[0.002]
loc	-0.001	-0.000	-0.001	-0.002	-0.003	0.003*	-0.000**	-0.030**	-0.001***
	[0.365]	[0.466]	[0.157]	[0.192]	[0.133]	[0.095]	[0.035]	[0.027]	[0.000]
use_exp	0.810*	0.024	0.033	1.454	1.440	-2.495***	-0.039	6.452	0.151
	[0.092]	[0.515]	[0.704]	[0.360]	[0.365]	[0.002]	[0.668]	[0.302]	[0.547]
Constant	-0.707	0.091	-0.464	0.237	3.731	98.647***	2.544***	132.553**	3.752**
	[0.797]	[0.686]	[0.453]	[0.966]	[0.548]	[0.000]	[0.000]	[0.041]	[0.045]
Obs	52	52	52	52	52	52	52	52	52
R-squared	0.287	0.091	0.273	0.337	0.354	0.676	0.304	0.244	0.296

Table B

All	11.144***	1.906**	1.901*	21.803***	23.951***	-5.336	-0.570***	155.098***	2.498*
	[0.000]	[0.030]	[0.089]	[0.000]	[0.000]	[0.207]	[0.004]	[0.000]	[0.059]
gender	-0.031	0.673	2.056***	1.040	0.454	3.273	-0.050	-33.916	-1.064
	[0.992]	[0.273]	[0.009]	[0.908]	[0.961]	[0.391]	[0.829]	[0.545]	[0.576]
CR	0.100	-0.028	-0.006	0.309	0.210	0.058	-0.007	-0.017	-0.005
	[0.200]	[0.198]	[0.870]	[0.166]	[0.449]	[0.614]	[0.284]	[0.994]	[0.942]
lan_OO	1.427	-0.987	-0.502	5.327	3.414	-11.066***	-0.160	45.564	1.742
	[0.582]	[0.313]	[0.689]	[0.389]	[0.635]	[0.007]	[0.512]	[0.437]	[0.373]
HD	0.073***	0.004	0.001	0.103**	0.121**	-0.095**	-0.002	0.414	0.007
	[0.000]	[0.436]	[0.933]	[0.031]	[0.050]	[0.042]	[0.204]	[0.416]	[0.586]
loc	-0.001***	0.000	0.000	-0.001	-0.001	-0.002***	-0.000	-0.016*	-0.000*
	[0.006]	[0.844]	[0.644]	[0.382]	[0.316]	[0.004]	[0.296]	[0.072]	[0.086]
use_exp	-0.049	0.036	-0.204	-0.344	-0.001	-0.584	-0.075	3.683	-0.323
	[0.940]	[0.864]	[0.356]	[0.779]	[0.999]	[0.640]	[0.265]	[0.749]	[0.387]
Constant	-2.003	0.158	-0.299	-2.196	0.950	84.386***	2.809***	117.100	5.256*
	[0.588]	[0.857]	[0.827]	[0.819]	[0.933]	[0.000]	[0.000]	[0.194]	[0.064]
Obs	64	64	64	64	64	64	64	64	64
R-squared	0.490	0.125	0.101	0.386	0.376	0.506	0.235	0.292	0.110

Table C

Extra	7.835***	1.174**	1.004	16.316***	17.440***	-3.583	-0.646***	110.509***	1.260
	[0.000]	[0.019]	[0.160]	[0.000]	[0.000]	[0.356]	[0.000]	[0.000]	[0.235]
gender	0.188	0.420	1.517**	-0.985	-1.549	4.828*	-0.128	-33.430	-1.155
	[0.944]	[0.413]	[0.020]	[0.892]	[0.836]	[0.097]	[0.484]	[0.459]	[0.445]
CR	0.072	-0.014	0.000	0.280	0.216	0.069	-0.008	-0.018	-0.007
	[0.359]	[0.330]	[0.994]	[0.170]	[0.362]	[0.424]	[0.147]	[0.992]	[0.901]
lan_OO	-0.305	-0.665	-0.544	1.043	-0.384	-6.991**	-0.246	29.524	0.968
	[0.891]	[0.353]	[0.557]	[0.830]	[0.944]	[0.028]	[0.210]	[0.490]	[0.500]

HD	0.092***	0.007	0.006	0.138**	0.158**	-0.123**	-0.001	0.651	0.016
	[0.000]	[0.104]	[0.485]	[0.011]	[0.019]	[0.034]	[0.497]	[0.222]	[0.269]
loc	-0.001**	-0.000	0.000	-0.001	-0.001	-0.002**	-0.000*	-0.017**	-0.000*
	[0.013]	[0.886]	[0.870]	[0.314]	[0.258]	[0.028]	[0.052]	[0.049]	[0.050]
use_exp	0.352	0.009	-0.137	0.706	0.925	-1.200	-0.085	6.256	-0.204
	[0.506]	[0.956]	[0.474]	[0.553]	[0.440]	[0.195]	[0.172]	[0.454]	[0.450]
Constant	-2.718	-0.049	-0.318	-2.270	0.248	83.988***	2.927***	108.047	5.059**
	[0.411]	[0.937]	[0.756]	[0.788]	[0.979]	[0.000]	[0.000]	[0.144]	[0.031]
Obs	89	89	89	89	89	89	89	89	89
R-squared	0.374	0.075	0.059	0.298	0.296	0.467	0.227	0.196	0.066

Table D

Feedback	6.611***	1.472**	1.783**	11.489**	13.977***	-1.759	0.186	99.726***	2.740**
	[0.004]	[0.044]	[0.021]	[0.014]	[0.004]	[0.434]	[0.324]	[0.006]	[0.028]
gender	-0.527	0.551	1.901*	-5.382	-2.855	4.641**	-0.005	-20.536	-0.044
	[0.899]	[0.484]	[0.059]	[0.633]	[0.803]	[0.038]	[0.982]	[0.702]	[0.981]
CR	0.074	-0.021	0.009	0.386	0.340	0.086	-0.007	0.455	0.011
	[0.463]	[0.340]	[0.809]	[0.112]	[0.221]	[0.166]	[0.267]	[0.832]	[0.862]
lan_OO	-1.759	-1.230	-1.298	-0.008	-3.040	-3.125	-0.387*	43.412	1.048
	[0.537]	[0.300]	[0.351]	[0.999]	[0.619]	[0.187]	[0.074]	[0.365]	[0.498]
HD	0.048	-0.009	-0.023	0.035	0.040	0.020	-0.000	0.182	0.003
	[0.139]	[0.619]	[0.263]	[0.498]	[0.562]	[0.548]	[0.814]	[0.731]	[0.824]
loc	0.007	0.003	0.005	0.016	0.017	-0.030***	-0.000	0.020	-0.000
	[0.320]	[0.408]	[0.236]	[0.114]	[0.182]	[0.000]	[0.321]	[0.800]	[0.917]
use_exp	0.496	-0.002	-0.216	1.284	1.512	-0.053	-0.148**	8.920	-0.460
	[0.509]	[0.995]	[0.399]	[0.476]	[0.406]	[0.917]	[0.022]	[0.486]	[0.147]
Constant	1.398	0.160	-0.748	7.990	8.044	79.637***	2.355***	142.253*	4.606*
	[0.749]	[0.851]	[0.522]	[0.486]	[0.506]	[0.000]	[0.000]	[0.079]	[0.054]
Obs	62	62	62	62	62	62	62	62	62
R-squared	0.392	0.164	0.218	0.306	0.314	0.689	0.213	0.160	0.113

6.3.3 Examine Sustainability of Effect of Gamification

In this section, I examine the third hypothesis (H3), which posits that the effect of gamification is sustainable over time, using a combination of a figure and a table. The data is processed and presented in Figure 6.10, utilizing the method previously demonstrated. The variables in Figure 6.10 are all close to 1, suggesting that there is no significant difference between the groups that continued to receive gamification treatment and those that discontinued it. To further verify this conclusion, I conduct a bivariate analysis in Table 6.16, the results of which are statistically insignificant, with each p-value exceeding 0.1. This leads us to conclude that the effect of gamification is indeed sustainable over time.

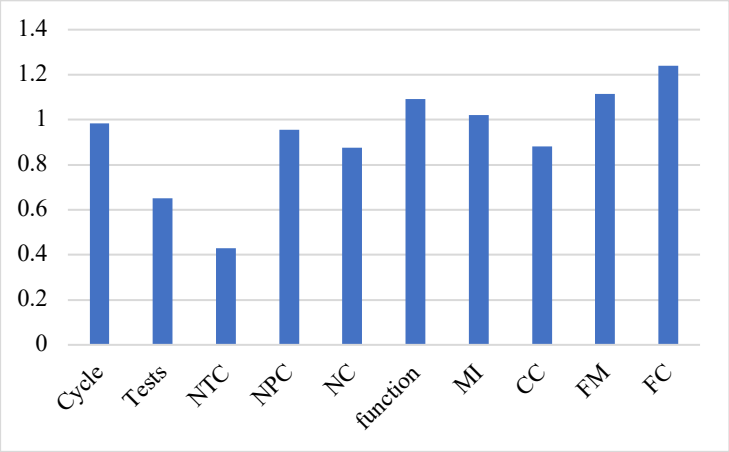


Figure 6.10 Sustainability of Gamification

	stop	continue	Mean1	Mean2	dif	St Err	t value	p value
Cycle by Continue: 0 1	44	45	8.204	10.155	-1.951	2.258	-0.85	0.39
Test by Continue: 0 1	44	45	1.046	0.978	0.068	0.648	0.1	0.917
NTC by Continue: 0 1	44	45	1.387	1.556	-0.169	0.806	-0.2	0.834
NPC by Continue: 0 1	44	45	22.137	21.689	0.448	4.657	0.1	0.923
NC by Continue: 0 1	44	45	25.182	23.889	1.293	5.014	0.25	0.797
function by Continue: 0 1	44	45	2.978	3.111	-0.134	0.512	-0.25	0.794
MI by Continue: 0 1	44	45	77.916	69.264	8.652	3.722	2.3	0.022
CC by Continue: 0 1	44	45	1.839	1.913	-0.074	0.171	-0.45	0.664
FM by Continue: 0 1	44	45	209.97	205.192	4.778	32.955	0.15	0.885
FC by Continue: 0 1	44	45	5.274	5.446	-0.172	1.038	-0.15	0.869

Table 6.16 Bivariate Analysis of Sustainability of Gamification

Chapter 7 Discussion

In this section, we begin with an overall discussion of the validity of the study. Subsequently We discuss each in the order in which the research was conducted, including the emergence of the research questions, key findings, and implications and significance for reality.

7.1 Threat to Validity

In this section, I discuss the threats to the validity of my study, which is classified by (Wohlin et al., 2012):

-Internal validity threats are the influences of independent variables on causality beyond the researcher's knowledge.

-External validity threats limit the results from an experimental study conducted in a different context to other practices, such as industry.

-Construct validity refers to whether the experimental results could be generalized to other theories and concepts behind this experiment.

-Conclusion validity threats are concerned with issues that affect the ability to draw the correct conclusion about relations between the treatment and the outcome of an experiment.

7.1.1 Internal Validity

In this study, various factors that could threaten the internal validity were considered and efforts were made to mitigate their impact. The potential threats to internal validity include selection, history, maturation, testing, instrumentation, and statistical regression.

With regards to selection, the reason that all my sample consists of the third-year level participants is that they are in their last year of bachelor's degree and are ready for to job market. Also, the sample was constructed to minimize the selection effect by selecting participants from the same college, grade, course, time limits, and task. This ensured that the participants have similar skills and task difficulty levels. However, I acknowledge that this study might have potential threats like adopting different development tools. Although this requirement is not applied to the project, the impact might be small since the task assigned is simple.

In experimental studies, it is important to acknowledge the potential influence of self-selection bias. The participants who volunteered to be part of the gamification intervention may possess

characteristics or motivations that differ from those who did not participate. This could introduce a potential bias in the results, as the participants who self-selected may already be more motivated or interested in the topic, potentially inflating the observed effectiveness of the gamification intervention. Consequently, the findings may not be fully generalizable to all development teams or software development contexts. But the results provide insights into the effects of gamification within the specific sample studied.

In terms of history, in observational study, while some cohorts attended lectures in person and others attended online, I conducted a bivariate analysis to assess whether the mode of instruction had an impact on my results. My findings indicate that the mode of instruction did not significantly affect my results. In experimental studies, all students attended lectures online. Therefore, I can conclude that there is no history threat associated with the mode of instruction in this study.

Regarding maturation validity, in observational study, the project only lasts for four weeks, during which students are more likely to keep the same level of skills, thus the threat of maturation validity can be reduced. In experimental studies, the project only lasted for 45 days, which is not a significant time period for skill level changes. So, the threat of maturation validity could be reduced.

To minimize the testing threat, the project is released as the first task at the beginning of each semester, and this module is the first course that introduces TDD. However, I cannot guarantee students do not have experience with TDD before participating in this project.

The instrumentation threat could be eliminated since all data is collected from the same source GitHub, and the students had similar abilities to use GitHub. To minimize the statistical regression threat, I do several diagnostic, for example, extreme values were excluded in the regression test.

7.1.2 External Validity

The external validity limits the generalizability of my results (Ciolkowski et al., 2004; Hannay and Jørgensen, 2008) to industry.

Interaction of selection and treatment: the participants in the study do not represent the entire population of software developers, especially senior engineers. However, the size and the complexity of the task do not require a high level of industrial experience, thus I believe the students could be appropriate participants as entry-level practitioners. Also, my sample could be initially representative of both students from an academic perspective and entry-level engineers from an industry perspective on an Irish base. To control other factors that might impact my dependent variable (maintainability), and since most are unobservable, such as personal skills and experience, therefore I construct the sample based on certain criteria to try to minimize the selection effect. Students that participated in this project are from the same college, grade, and course, and are assigned the same tasks and time limits, so the participants have similar knowledge and task difficulty level.

Interaction of setting and treatment: As the size and the complexity of this task is limited, it might not be representative of real-world tasks, such as complex software or advanced system development.

Interaction of history and treatment: To minimize the impact of external threats, such as a pandemic and the occurrence of environmental disasters, I collect all data online and test the impact of covid on the module.

7.1.3 Construct Validity

Inadequate preoperational explication of constructs: The TDD behaviors and maintainability are based on transparent, previous definitions that were argued and tested. However, I acknowledge the challenge of measuring engagement level in development activity, which is hard to quantify. As a result, to be as accurate as possible, I understand the engagement level from the behavioral engagement view and adopt two instrumental variables from different perspectives including commit number and commit frequency. In the future, it could be defined more accurately through interviews with students and developers.

Evaluation apprehension: This study is not exposed to this threat because the students are not evaluated based on the results they obtained in the experiment.

Experimenters' expectancies: The participants will not consciously bias the results based on their expectations of the results, as this is an ex post facto study.

7.1.4 Conclusion Validity

Low statistical power: The limited statistical power of the test resulting from the small sample size may make it more difficult to draw accurate inferences from the outcomes of my investigation. However, my sample size is 237 and it can eliminate this threat (The power analysis results show that a sample size of 156 is sufficient).

Fishing and the error rate: fishing has been mitigated because I analysis data with program and same standard to limit the fostering for specific outcomes. As for error rate, I chose three adequate significances level (0.1, 0.05, and 0.01) while testing null hypotheses.

Reliability of measures: The majority of the measures were automatized, and can be repeated with the same outcome. When human judgement was necessary, I adopted a uniform standard.

Random heterogeneity of participants: Theoretically, due to the similar backgrounds of the subjects, i.e., students from the same university, same module, same grade, and had similar experience, heterogeneity should be reduced. Empirically, I adopt Robust Standard Error to solve this problem.

7.2 Discussion of Studies

7.2.1 Observational Study

Drawing inspiration from recent research suggesting that the benefits of TDD might not be solely attributed to its test-first characteristics, but rather, to the augmented focuses on coding and testing (Fucci et al., 2016; Fucci et al., 2015). This leads to my question that does following TDD behaviors have positive impacts on engagement level in development activities (RQ2a). My results support this conjecture, whereby a stronger impact is observed for engagement during the testing phase relative to the coding phase. I then elaborate empirically on the consequences of engagement levels on software maintainability (RQ2b), and find that engagement level has significant and positive performance consequences for developer's code quality. My results remain robust when controlling for various factors such as readability, different tasks, size, and programming language. Of particular interest, the positive impact on maintainability is especially pronounced for engagement in the testing phase versus the coding phase. Thus, the enhancement of engagement levels can have substantial implications not only for software quality but also for developer performance.

Our findings underscore the importance of engagement levels in explaining the performance consequences of coding quality. It integrates ideas from software psychology and propose a novel metric for quantifying engagement levels during the software development process. Unlike prior work that measures engagement using broad indicators, such as duration of time spent, which might not be entirely applicable to the domain of software engineering, my study constructs the engagement's response variables on a commit-based approach instead. This approach can be easily implementable and generalizable to all GitHub users, enabling the monitoring of performance across both long-run or short-run horizon.

Our study has several practical implications. Beyond the conventional notion that focusing on programming skills leads to a better maintainability, I provide new insights into the significance of engagement levels during software development activities as a determinant of coding quality. Furthermore, my study provides some implications to software engineering education, advocating for the integration of TDD practices in the curricula. Additionally, by combining my data with

longitudinal measures of evolving software codebase quality, I thereby can explain and link professional software engineering practical activity to outcomes in terms of individual software quality.

Relatedly, my findings here are applicable to student and novice developers but cannot necessarily be generalizable to senior engineers. Therefore, a direct extension of my work would be to examine my proposed engagement metric in a broader context, i.e., experienced engineers and sophisticated projects. Further, future studies could also explore the potential role of engagement in productivity such as LOC per hour. Additionally, apart from the examination of TDD method on individual projects, it would be of special interest to explore whether behavior-driven development (BDD) impacts engagement levels at the team level, where BDD is a team methodology.

7.2.2 Group Experiment

This section provides a summary of the research design employed in the group setting experiment, including the research questions, experimental design, and the resulting findings.

The research question (RQ3) addressed in this study are centered on the impact of gamification on students' TDD practices and the improvement of software maintainability. This is motivated by prior literature that suggests gamification can enhance user performance and promote personal focus.

To address these research questions, a 45-day experimental study was conducted, and data was collected from the GitHub repository. The effectiveness of gamification on TDD behaviors and the engagement level in development activity was analyzed using graphical expressions. The relationship between gamification and software maintainability was evaluated through statistical analysis (OLS).

The findings of this study support the hypothesis that gamification can effectively improve students' TDD practices and software maintainability. Gamification was found to positively influence TDD behaviors and the engagement level in development activities, as well as positively relate to software maintainability. These results align with the notion that gamification can enhance software engineering practice performance.

The study under consideration is the first of its kind to explore the potential of gamification in enhancing complex software development activities. This is a significant contribution in itself. Moreover, the findings of this study have important implications for educators, software developers, and project managers who are seeking to improve the efficiency of software development. By incorporating gamification into their curriculum, educators can enhance the TDD practices of their students. Similarly, software developers can use gamification strategies to motivate team members and improve project outcomes, while project managers can increase the engagement levels of their team members and improve overall project quality through gamification. The study also proposes gamification as a promising and cost-effective alternative to traditional methods such as introducing new methodologies or enhancing expertise. This finding is particularly important for organizations seeking efficient ways to improve their software development processes.

7.2.3 Individual Experiment

In this section, I summarize the significance of the experiment in individual setting, outline recommendations for educator, and discuss the potential implications of my findings for future research in the field of gamification. My study focuses on the application of gamification strategies on the practice of Test-Driven Development (TDD) in individual setting (RQ3), and examines the most commonly used gamification strategies such as rewards, points, leaderboards, and feedback in the software engineering domain (RQ4), and gamification sustainability (RQ5) (Çeker and Özdaml, 2017; Koivisto and Hamari, 2019; Ren et al., 2020).

TDD is a critical software development practice aimed at improving software maintainability (Beck, 2003; C. Chen et al., 2017; Mäkinen and Münch, 2014; Tosun et al., 2017), but it can be challenging for students and novice developers to adopt and maintain (Hammond and Umphress, 2012). My study is motivated by previous research that suggests that gamification can improve users' performance in completing a task (Pedreira et al., 2015). My study aims to investigate whether gamification strategies can motivate students to adopt and maintain TDD practice and improve software maintainability.

Our results show that gamification strategies can positively impact students' TDD practice and software maintainability. In particular, I observed that gamification can change students' behaviors to follow TDD, and increase their engagement in development activities. Furthermore, I found that the combination of gamification strategies has a more positive impact than using individual strategies. My study also highlights the sustainability of the impact of gamification, even after the strategies have been withdrawn.

The practical implications of my study are significant. This contributes to the current literature by providing a clearer understanding of the potential impact of gamification on complex development activities. In addition, this study can help educators and junior software developers to design more effective gamification interventions that focus on specific areas of TDD practices. This experiment distinguishes the impact of different gamification strategies on TDD practice by constructing various gamification strategy combinations. This can help inform the design and implementation of gamification interventions in software development contexts. This contribution is significant as it suggests that gamification can have long-lasting effects on TDD practice, leading to sustainable improvements in software development efficiency. These findings have important implications for universities and the entre-level industry since they suggest that gamification can positively impact TDD practices and can be easily replicated and generalized to broader contexts.

However, it should be noted that my findings apply only to students and novice developers and may not necessarily generalize to senior engineers or complex projects. Therefore, future studies should explore the potential role of gamification in software management practices, such as team communication, and extend the scope of my work to broader contexts, including experienced engineers and sophisticated projects.

7.2.4 Engagement

In the context of software development activities, "engagement" refers to the level of involvement and focus by developers towards their tasks and projects. The studies measure engagement primarily through observable behaviors during software development activities. For instance, in the observational study, engagement is inferred from developers' activities, with a focus on coding and

testing phases. The activities, such as writing tests, code, and frequent testing, is considered indicative of engagement. Similarly, in the group and individual experiments, engagement is assessed based on students' development activities.

While these studies provide valuable insights into engagement levels within software development contexts, it's essential to acknowledge the limitations in measuring engagement through observable behaviors alone. Engagement is a multifaceted construct that encompasses not only outward actions but also internal states such as motivation, interest, and cognitive involvement, which may not always be directly observable or accurately captured through development behaviors.

Moreover, the claims made regarding the impact of engagement on software quality should be interpreted with caution. While the studies suggest a positive association between engagement and software maintainability, it's important to recognize that correlation does not imply causation. Other variables and factors may influence these outcomes, and the observed relationships may be influenced by confounding variables or alternative explanations.

In summary, while the studies provide valuable insights into engagement levels and their impact on software maintainability, it's important to recognize the complexity of the concept of engagement and the limitations of measuring it through development behaviors alone. Further research exploring alternative measures and deeper understandings of engagement in software development activities can contribute to a more comprehensive understanding of its role and implications.

7.2.5 Limits of Empirical Study

Statistical analysis plays an important role in empirical research, providing quantitative insights into the relationships between variables, the significance of observed effects, and the generalizability of findings. Through statistical techniques such as regression analysis and correlation analysis, I can uncover associations within the data, which can inform the impact of factors such as TDD behaviors, engagement levels, and gamification techniques on software quality. While statistical analysis is a powerful tool for quantifying relationships and making inferences, it's important to acknowledge its limitations in capturing the complexity of human behavior.

An important limitation is that complex human behavior may not be fully captured by statistical analysis alone. Human behavior is multifaceted, influenced by many factors including individual differences, social context, and cultural background. While statistical methods can quantify relationships between variables, they may not adequately capture the underlying mechanisms that drive human behavior. For example, while statistical analysis may reveal a significant correlation between engagement levels and coding quality, it may not provide insights into the specific motivational factors, or cognitive processes that underlie this relationship.

While quantitative data can provide valuable insights, it can be complemented with qualitative approaches, such as interviews or surveys, to gain a deeper understanding. Future research could benefit from quantitative analysis with qualitative approaches to gain a deeper understanding of the underlying factors. Additionally, exploring broader contexts and extending the study to include more diverse populations, such as experienced engineers and sophisticated projects, could further enhance the generalizability and practical relevance of the findings.

Moreover, statistical analysis is inherently limited by the quality and scope of the data collected. Empirical studies often rely on data that are subject to various sources of bias, measurement error, and confounding variables, which can impact the validity and reliability of statistical findings. Therefore, the factors such as effect size and practical relevance are also considered to provide a more nuanced interpretation of the results.

Overall, while the current study makes significant contributions, there are opportunities for future research to build upon these findings and deepen our understanding of the relationship between engagement and software development outcomes.

Chapter 8 Conclusion and Future Work

This thesis expends explanation and examines the application of gamification strategies in software engineering practice in education setting. Chapter 3 presenting a framework and configurable method for selecting appropriate gamification elements. Chapter 4 demonstrates studies setting, and Chapter 5 shows the methodology, which includes variables definition and gamification design. Chapter 6 shows the results of the thesis, and Chapter 7 discuss the validity. Finally, Overall, this research highlights the potential of gamification as a practical tool to enhance the efficiency of software engineering practice.

8.1 Contribution

This thesis presents a systematic approach to applying gamification strategies in software engineering practice, offering a theory-based framework for designing gamification interventions. The thesis highlights the positive impact of gamification on Test Driven Development (TDD) practices in both individual and group settings.

The main contributions to the field of software engineering and gamification include: first of all, it provides a novel, theory-based framework to designing gamification for software engineering, offering researchers a systematic approach to apply gamification strategies on software engineering practice.

Second, this thesis provide a novel analysis of the relationship between TDD, engagement level in development activity, as well as maintainability. It is a novel study to construct a set of commit-based proxies to measure engagement in development activities, especially for software engineering area. This thesis also provide empirical evidence beyond the conventional notion that focusing on programming skills leads to a better maintainability, underscore the importance of engagement levels in development activities.

Third, this thesis did comprehensive analysis of various gamification strategies, provide empirical evidence of gamification impact on software engineering practice, and provide insights for educators, junior developers, and managers seeking to improve software engineering practice efficiency.

For educators, this work facilitates effective teaching of software engineering concepts and skills.

For junior developers, this study could encourage self-engagement and improved performance in their work.

For managers, this work might provide a way to optimize team performance and efficiency.

For society, the potential benefit of this thesis is to enable a more cost-effective development process.

It underscore the role of gamification as a promising and cost-effective alternative to traditional methods.

8.2 Limitation

The current thesis focuses on the application of gamification in educational settings, and the impact of gamification on software engineering practices. While the results suggest that gamification can be effective in encouraging students to adopt and maintain best practices in software engineering, the current study is limited to education setting, and the effects of gamification may vary across different contexts and settings.

Replication of the evaluations: The chosen sample for the evaluations comprised students from Trinity College Dublin. However, in order to assess the effectiveness of gamification on software engineering practice, further evaluations ought to be conducted using samples from other universities, such as undergraduate students from China and the United States of America. These replications should be performed using a larger sample size, in order to acquire more accurate results.

In order to increase the validity of the results, it is important to consider the participants' different cultural backgrounds. This will allow for a more comprehensive assessment of the effectiveness of gamification as a teaching tool. Furthermore, it is worth noting that different nationalities may have different attitudes towards gaming and software engineering. For example, the American culture may be more accepting of gaming than the Chinese culture. It is therefore important to consider the attitudes of each culture when conducting the studies.

Extend to broader contexts: The effectiveness of gamification in educational software engineering has been established, yet it is crucial to further examine its implementation in a variety of other contexts. For instance, the response to gamification strategies among senior developers might vary greatly; they may reject the idea of participating in a "game". Furthermore, the complexity of tasks posed in industrial environments may also impact the efficacy of gamification, as it may be difficult to identify which behaviors have a positive influence on project outcomes. Consequently, these considerations should not be overlooked and provide ample opportunity for further research.

External validity: Some of the studies were conducted in controlled environments, which may not reflect the complexity and variability of real-world software development. Future studies should consider conducting experiments in more realistic settings to improve external validity.

Long-term sustainability: Although some studies suggest that the effects of gamification interventions can be long-lasting, it is unclear whether the effects would persist over longer periods of time or in different contexts. Further research is needed to investigate the long-term sustainability of gamification interventions.

8.3 Implications for future research

Due to the limits of this doctoral thesis, I leave some unaddressed questions that might be of interest for future research. One important direction is to replicate the evaluations using different samples from various universities and countries. While the current study was limited to students from Trinity College Dublin, future studies should aim to expand the scope to include undergraduates from China, the United States, and other countries. In doing so, I can assess whether the effectiveness of gamification as a teaching tool varies across different cultural contexts and nationalities.

Another area for future research is to extend the study to other contexts beyond educational settings. Specifically, it may be useful to explore how gamification strategies can be implemented in industrial environments with senior developers. Additionally, I could investigate how gamification can be used to promote specific behaviors that contribute to project outcomes. This could help address the concern that the complexity of tasks in industrial settings may make it difficult to identify which behaviors have a positive influence on project outcomes.

Another important consideration for future research is the external validity of studies. While my studies have been conducted in controlled environments, it is unclear whether the results would hold up in more realistic settings. Future studies should aim to address this issue by conducting experiments in more realistic settings to improve external validity.

Finally, it is worth exploring the long-term sustainability of gamification interventions. While my studies suggest that the effects of gamification can be long-lasting, it is unclear whether these effects would persist over longer periods of time or in different contexts. Additional research is needed to investigate the long-term sustainability of gamification interventions.

One potential limitation of the current thesis is that all of the studies were conducted by me, which could introduce potential bias in data collection and analysis. Future studies should consider using multiple researchers to minimize the potential for bias.

References

Aggarwal, K. K., Singh, Y., & Chhabra, J. K. (2002). *An integrated measure of software maintainability*.

Akpolat, B. S., & Slany, W. (2014). *Enhancing software engineering student team engagement in a high-intensity extreme programming course using gamification.* Paper presented at the 2014 IEEE 27th conference on software engineering education and training (CSEE&T).

Al-Yafi, K., & El-Masri, M. (2016). Gamification of e-government services: A discussion of potential trans formation.

Alawad, D., Panta, M., Zibran, M., & Islam, M. R. (2019). An empirical study of the relationships between code readability and s oftware complexity. *arXiv preprint arXiv:1909.01760*.

Albilali, A. A., & Qureshi, R. J. (2016). Proposal to Teach Software Development Using Gaming Technique. *International Journal of Modern Education & Computer Science, 8*(8).

Alhammad, M. M., & Moreno, A. M. (2018). Gamification in software engineering education: A systematic mapping. *Journal of Systems and Software, 141*, 131-150.

Alhammad, M. M., & Moreno, A. M. (2020). Challenges of gamification in software process improvement. *Journal of Software: Evolution and Process, 32*(6), e2231.

Allamanis, M., Barr, E. T., Bird, C., & Sutton, C. (2014). *Learning natural coding conventions.* Paper presented at the Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering.

Alsolai, H., & Roper, M. (2020). A systematic literature review of machine learning techniques for soft ware maintainability prediction. *Information and Software Technology, 119*, 106214.

Alsunki, A. A. M., Ali, M. A., Jaharadak, A. A., & Tahir, N. M. (2020). *Framework of software developers engagement antecedents and productivity-A review.* Paper presented at the 2020 16th IEEE International Colloquium on Signal Processing & Its Applications (CSPA).

Anderson, D. J. (2003). *Agile management for software engineering: Applying the theory of cons traints for business results*: Prentice Hall Professional.

Ardito, L., Coppola, R., Barbato, L., & Verga, D. (2020). A Tool-Based Perspective on Software Code Maintainability Metrics: A S ystematic Literature Review. *Scientific Programming, 2020*.

Ašeriškis, D., & Damaševičius, R. (2014). *Gamification of a project management system*.

Avritzer, A., Paulish, D., Cai, Y., & Sethi, K. (2010). Coordination implications of software architecture in a global softwar e development project. *Journal of Systems and Software, 83*(10), 1881-1895.

Babbie, E. (2007). The practice of social research. 11th. *Belmont, CA: Thomson Wadsworth, 24*(511), 66.

Bacharach, S. B., & Aiken, M. (1977). Communication in administrative bureaucracies. *Academy of Management Journal, 20*(3), 365-377.

Bakker, A. B., & Demerouti, E. (2008). Towards a model of work engagement. *Career development international*.

Baltes, S., Knack, J., Anastasiou, D., Tymann, R., & Diehl, S. (2018). *(No) influence of continuous integration on the commit activity in Git Hub projects*.

Banker, R. D., & Datar, S. M. (1989). SOFrWARE COMPLEXITY AND MAINTAINABILITY.

Barata, G., Gama, S., Jorge, J. A. P., & Gonçalves, D. J. V. (2014). *Relating gaming habits with student performance in a gamified learning experience*.

Barreto, C. F., & França, C. (2021). *Gamification in Software Engineering: A literature Review*.

Beck, K. (2000). *Extreme programming explained: embrace change*: addison-wesley professional.

Beck, K. (2003). *Test-driven development: by example*: Addison-Wesley Professional.

Bender, M. A., Farach-Colton, M., Pemmasani, G., Skiena, S., & Sumazin, P. (2005). Lowest common ancestors in trees and directed acyclic graphs. *Journal of Algorithms, 57*(2), 75-94.

Benesty, J., Chen, J., Huang, Y., & Cohen, I. (2009). Pearson correlation coefficient. In *Noise reduction in speech processing* (pp. 1-4): Springer.

Bertolino, A. (2007). *Software testing research: Achievements, challenges, dreams*.

Bhat, T., & Nagappan, N. (2006). *Evaluating the efficacy of test-driven development: industrial case studies.* Paper presented at the Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering.

Bianchini, D., Fogli, D., & Ragazzi, D. (2016a). *Promoting citizen participation through gamification*.

Bianchini, D., Fogli, D., & Ragazzi, D. (2016b). *TAB sharing: A gamified tool for e-participation*.

Bíró, G. I. (2014). Didactics 2.0: A pedagogical analysis of gamification theory from a co mparative perspective with a special view to the components of learnin g. *Procedia-Social and Behavioral Sciences, 141*, 148-151.

Bissi, W., Neto, A. G. S. S., & Emer, M. C. F. P. (2016). The effects of test driven development on internal quality, external q uality and productivity: A systematic review. *Information and Software Technology, 74*, 45-54.

Bjarnason, E., & Sharp, H. (2017). The role of distances in requirements communication: a case study. *Requirements Engineering, 22*(1), 1-26.

Bjarnason, E., Sharp, H., & Regnell, B. (2019). Improving requirements-test alignment by prescribing practices that mi tigate communication gaps. *Empirical Software Engineering, 24*(4), 2364-2409.

Bjarnason, E., Smolander, K., Engström, E., & Runeson, P. (2016). A theory of distances in software engineering. *Information and Software Technology, 70*, 204-219.

Bjarnason, E., Wnuk, K., & Regnell, B. (2011). *Requirements are slipping through the gaps—A case study on causes & ef fects of communication gaps in large-scale software development*.

Bjørnson, F. O., Wijnmaalen, J., Stettina, C. J., & Dingsøyr, T. (2018). *Inter-team coordination in large-scale agile development: A case study of three enabling mechanisms*.

Blaine, B. E. (2018). Winsorizing. *The SAGE Encyclopedia of Educational Research, Measurement, and Evalua tion*, 1817.

Blanchard, B. S., Verma, D. C., & Peterson, E. L. (1995). *Maintainability: a key to effective serviceability and maintenance management* (Vol. 13): John Wiley & Sons.

Boehm, B., & Turner, R. (2003). People factors in software management: lessons from comparing agile an d plan-driven methods. *Crosstalk-The Journal of Defense Software Engineering,(Dec 2003)*.

Boehm, B. W. (1976). Software engineering. *IEEE Trans. Computers, 25*(12), 1226-1241.

Boehm, B. W. (1984). Software engineering economics. *IEEE Transactions on Software Engineering*(1), 4-21.

Boehm, B. W., Brown, J. R., & Lipow, M. (1976). *Quantitative evaluation of software quality*.

Boehm, B. W., & Ross, R. (1989). Theory-W software project management principles and examples. *IEEE Transactions on Software Engineering, 15*(7), 902-916.

Boneau, C. A. (1960). The effects of violations of assumptions underlying the t test. *Psychological bulletin, 57*(1), 49.

Borjesson, A., & Mathiassen, L. (2004). Successful process implementation. *IEEE software, 21*(4), 36-44.

Borle, N. C., Feghhi, M., Stroulia, E., Greiner, R., & Hindle, A. (2018). Analyzing the effects of test driven development in GitHub. *Empirical Software Engineering, 23*(4), 1931-1958.

Botella, P., Burgués, X., Carvallo, J., Franch, X., Grau, G., Marco, J., & Quer, C. (2004). *ISO/IEC 9126 in practice: what do we need to know.* Paper presented at the Software Measurement European Forum.

Botte, B., Bakkes, S., & Veltkamp, R. (2020). *Motivation in gamification: constructing a correlation between gamification achievements and self-determination theory.* Paper presented at the Games and Learning Alliance: 9th International Conference, GALA 2020, Laval, France, December 9–10, 2020, Proceedings 9.

Breevaart, K., Bakker, A. B., Demerouti, E., & Hetland, J. (2012). The measurement of state work engagement. *European Journal of Psychological Assessment*.

Briciu, C.-V., & Filip, I. (2018). Applying gamification for mindset changing in automotive software proj ect management. *Procedia-Social and Behavioral Sciences, 238*, 267-276.

Buchan, J., Li, L., & MacDonell, S. G. (2011). *Causal factors, benefits and challenges of test-driven development: Pr actitioner perceptions*.

Buffardi, K., & Edwards, S. H. (2014). *A formative study of influences on student testing behaviors*.

Busenbark, J. R., Yoon, H., Gamache, D. L., & Withers, M. C. (2021). Omitted Variable Bias: Examining Management Research With the Impact T hreshold of a Confounding Variable (ITCV). *Journal of Management*, 01492063211006458.

Calder, B. J., Phillips, L. W., & Tybout, A. M. (1982). The concept of external validity. *Journal of consumer research, 9*(3), 240-244.

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

Cataldo, M., & Herbsleb, J. D. (2012). Coordination breakdowns and their impact on development productivity a nd software failures. *IEEE Transactions on Software Engineering, 39*(3), 343-360.

Çeker, E., & Özdaml, F. (2017). What" Gamification" Is and What It's Not. *European Journal of Contemporary Education, 6*(2), 221-228.

Chávez, A., Ferreira, I., Fernandes, E., Cedrim, D., & Garcia, A. (2017). *How does refactoring affect internal quality attributes? A multi-proje ct study*.

Chen, C., Alfayez, R., Srisopha, K., Boehm, B., & Shi, L. (2017). *Why is it important to measure maintainability and what are the best w ays to do it?*

Chen, J.-C., & Huang, S.-J. (2009). An empirical analysis of the impact of software development problem factors on software maintainability. *Journal of Systems and Software, 82*(6), 981-992.

Cheong, C., Filippou, J., & Cheong, F. (2014). Towards the gamification of learning: Investigating student perception s of game elements. *Journal of Information Systems Education, 25*(3), 233.

Choma, J., Guerra, E. M., & Silva, T. S. d. (2018). *Developers' initial perceptions on TDD practice: A thematic analysis w ith distinct domains and languages*.

Chou, Y.-k. (2019). *Actionable gamification: Beyond points, badges, and leaderboards*: Packt Publishing Ltd.

Chow, I., & Huang, L. (2017). *A software gamification model for cross-cultural software development teams*.

Ciani, L., Guidi, G., Patrizi, G., & Venzi, M. (2018). *System maintainability improvement using allocation procedures*.

Ciolkowski, M., Muthig, D., & Rech, J. (2004). *Using academic courses for empirical validation of software developmen t processes*.

Cohn, M., & Ford, D. (2003). Introducing an agile process to an organization [software development]. *Computer, 36*(6), 74-78.

Coleman, D., Ash, D., Lowther, B., & Oman, P. (1994). Using metrics to evaluate software system maintainability. *Computer, 27*(8), 44-49.

Coleman, R., & Boldt, B. (2018). Aesthetics Versus Readability of Source Code. *International Journal of Advanced Computer Science and Applications, 9*(9), 12-18.

Connolly, T. M., Stansfield, M., & Hainey, T. (2007). An application of games-based learning within software engineering. *British Journal of Educational Technology, 38*(3), 416-428.

Coram, M., & Bohner, S. (2005). *The impact of agile methods on software project management*.

Cordero, R., & Farris, G. F. (1992). Administrative activity and the managerial development of technical pr ofessionals. *IEEE Transactions on Engineering Management, 39*(3), 270-276.

Cosentino, V., Izquierdo, J. L. C., & Cabot, J. (2017). A systematic mapping study of software development with GitHub. *IEEE Access, 5*, 7173-7192.

Costello, S. H. (1984). Software engineering under deadline pressure. *ACM SIGSOFT Software Engineering Notes, 9*(5), 15-19.

Counsell, S., Liu, X., Eldh, S., Tonelli, R., Marchesi, M., Concas, G., & Murgia, A. (2015). *Re-visiting the'Maintainability Index'Metric from an Object-Oriented P erspective*.

Crispin, L. (2006). Driving software quality: How test-driven development impacts software quality. *IEEE software, 23*(6), 70-71.

Crowston, K., & Howison, J. (2005). The social structure of free and open source software development. *First Monday*.

Curcio, K., Navarro, T., Malucelli, A., & Reinehr, S. (2018). Requirements engineering: A systematic mapping study in agile software development. *Journal of Systems and Software, 139*, 32-50.

Cursino, R., Ferreira, D., Lencastre, M., Fagundes, R., & Pimentel, J. (2018). *Gamification in requirements engineering: a systematic review*.

Curtis, B., Krasner, H., & Iscoe, N. (1988). A field study of the software design process for large systems. *Communications of the ACM, 31*(11), 1268-1287.

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

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

Damian, D. (2001). *An empirical study of requirements engineering in distributed software projects: is distance negotiation more effective?*

Damian, D., Helms, R., Kwan, I., Marczak, S., & Koelewijn, B. (2013). *The role of domain knowledge and cross-functional communication in soc io-technical coordination*.

de Almeida Souza, M. R., Constantino, K. F., Veado, L. F., & Figueiredo, E. M. L. (2017). *Gamification in software engineering education: An empirical study.* Paper presented at the 2017 IEEE 30th Conference on Software Engineering Education and Training (CSEE&T).

Defranco, J. F., & Laplante, P. A. (2017). Review and analysis of software development team communication researc h. *IEEE Transactions on Professional Communication, 60*(2), 165-182.

Denny, P. (2013). *The effect of virtual achievements on student engagement*.

Denny, P., McDonald, F., Empson, R., Kelly, P., & Petersen, A. (2018). *Empirical support for a causal relationship between gamification and l earning outcomes*.

Deterding, S., Sicart, M., Nacke, L., O'Hara, K., & Dixon, D. (2011). Gamification. using game-design elements in non-gaming contexts. In *CHI'11 extended abstracts on human factors in computing systems* (pp. 2425-2428).

Dhillon, B. S. (2006). *Maintainability, maintenance, and reliability for engineers*: CRC press.

Diaz, D. B., & Yudin, A. K. (2017). The versatility of boron in biological target engagement. *Nature Chemistry*, *9*(8), 731-742.

Dicheva, D., Dichev, C., Agre, G., & Angelova, G. (2015). Gamification in education: A systematic mapping study. *Journal of educational technology & society, 18*(3), 75-88.

Dingsøyr, T., Moe, N. B., & Seim, E. A. (2018). Coordinating knowledge work in multiteam programs: findings from a lar ge-scale agile development program. *Project Management Journal, 49*(6), 64-77.

Dogša, T., & Batič, D. (2011). The effectiveness of test-driven development: an industrial case study. *Software Quality Journal, 19*(4), 643-661.

Dorling, A., & McCaffery, F. (2012). *The gamification of SPICE*.

Duarte, D., Farinha, C., da Silva, M. M., & da Silva, A. R. (2012). *Collaborative requirements elicitation with visualization techniques.* Paper presented at the 2012 IEEE 21st International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises.

Dubois, D. J., & Tamburrelli, G. (2013). *Understanding gamification mechanisms for software development*.

Dutta, S., & Van Wassenhove, L. N. (1997). An empirical study of adoption levels of software management practices within European firms.

Dyer, R., Nguyen, H. A., Rajan, H., & Nguyen, T. N. (2015). Boa: Ultra-large-scale software repository and source-code mining. *ACM Transactions on Software Engineering and Methodology (TOSEM), 25*(1), 1-34.

Ehrlich, K., & Cataldo, M. (2012). *All-for-one and one-for-all? A multi-level analysis of communication p atterns and individual performance in geographically distributed softw are development*.

El Shoubashy, H., ElKader, H., & Khalifa, N. (2020). What is gamification? A literature review of previous studies on gamification. *Australian Journal of Basic and Applied Sciences, 14*(8), 29-51.

Elgrably, I. S., & Oliveira, S. R. B. (2018). *Gamification and Evaluation of the Use the Agile Tests in Software Qua lity Subjects: The Application of Case Studies*.

Erdogmus, H., Morisio, M., & Torchiano, M. (2005). On the effectiveness of the test-first approach to programming. *IEEE Transactions on Software Engineering, 31*(3), 226-237.

Espinosa, J. A., & Carmel, E. (2003). The impact of time separation on coordination in global software teams : a conceptual foundation. *Software Process: Improvement and Practice, 8*(4), 249-266.

Estdale, J., & Georgiadou, E. (2018). *Applying the ISO/IEC 25010 quality models to software product*.

Eyolfson, J., Tan, L., & Lam, P. (2011). *Do time of day and developer experience affect commit bugginess?*

Farrar, D. E., & Glauber, R. R. (1967). Multicollinearity in regression analysis: the problem revisited. *The Review of Economic and Statistics*, 92-107.

Flemming, W. R. (1978). *Requirements communication*.

Fulcini, T., Coppola, R., Ardito, L., & Torchiano, M. (2023). A Review on Tools, Mechanics, Benefits, and Challenges of Gamified Software Testing. ACM Computing Surveys.

Ford, G. (1994). The progress of undergraduate software engineering education. *ACM SIGCSE Bulletin, 26*(4), 51-55.

Francisco-Aparicio, A., Gutiérrez-Vela, F. L., Isla-Montes, J. L., & Sanchez, J. L. G. (2013). Gamification: analysis and application. In *New trends in interaction, virtual reality and modeling* (pp. 113-126): Springer.

Frank, K. A. (2000). Impact of a confounding variable on a regression coefficient. *Sociological Methods & Research, 29*(2), 147-194.

Fraser, G. (2017). *Gamification of software testing*.

Fredricks, J. A., Blumenfeld, P. C., & Paris, A. H. (2004). School engagement: Potential of the concept, state of the evidence. *Review of educational research, 74*(1), 59-109.

Fucci, D., Erdogmus, H., Turhan, B., Oivo, M., & Juristo, N. (2016). A dissection of the test-driven development process: Does it really ma tter to test-first or to test-last? *IEEE Transactions on Software Engineering, 43*(7), 597-614.

Fucci, D., & Turhan, B. (2014). On the role of tests in test-driven development: a differentiated and partial replication. *Empirical Software Engineering, 19*(2), 277-302.

Fucci, D., Turhan, B., & Oivo, M. (2015). *On the effects of programming and testing skills on external quality a nd productivity in a test-driven development context*.

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

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

Garousi, V., & Mäntylä, M. V. (2016). A systematic literature review of literature reviews in software testi ng. *Information and Software Technology, 80*, 195-216.

Garousi, V., Rainer, A., Lauv\aas Jr, P., & Arcuri, A. (2020). Software-testing education: A systematic literature mapping. *Journal of Systems and Software, 165*, 110570.

Ghobadi, S. (2015). What drives knowledge sharing in software development teams: A literat ure review and classification framework. *Information & Management, 52*(1), 82-97.

Gladstein, D. L. (1984). Groups in context: A model of task group effectiveness. *Administrative science quarterly*, 499-517.

Goles, T., & Chin, W. W. (2005). Information systems outsourcing relationship factors: detailed concept ualization and initial evidence. *ACM SIGMIS Database: the DATABASE for Advances in Information Systems, 36*(4), 47-67.

Hajjdiab, H., & Taleb, A. S. (2011). Adopting agile software development: issues and challenges. *International Journal of Managing Value and Supply Chains (IJMVSC), 2*(3), 1-10.

Hakulinen, L., Auvinen, T., & Korhonen, A. (2013). *Empirical study on the effect of achievement badges in TRAKLA2 online learning environment*.

Hal, S. R. P., Post, M., & Wendel, K. (2019). Generating Commit Messages from Git Diffs. *arXiv preprint arXiv:1911.11690*.

Hallifax, S., Serna, A., Marty, J.-C., & Lavoué, E. (2019). *Adaptive gamification in education: A literature review of current tre nds and developments*.

Hamari, J. (2013). Transforming homo economicus into homo ludens: A field experiment on g amification in a utilitarian peer-to-peer trading service. *Electronic commerce research and applications, 12*(4), 236-245.

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

Hamari, J., Koivisto, J., & Sarsa, H. (2014). *Does gamification work?–a literature review of empirical studies on ga mification*.

Hamari, J., Shernoff, D. J., Rowe, E., Coller, B., Asbell-Clarke, J., & Edwards, T. (2016). Challenging games help students learn: An empirical study on engagemen t, flow and immersion in game-based learning. *Computers in Human Behavior, 54*, 170-179.

Hammond, S., & Umphress, D. (2012). *Test driven development: the state of the practice*.

Han, E., Yin, D., & Zhang, H. (2023). Bots with feelings: Should AI agents express positive emotion in customer service?. *Information Systems Research*, *34*(3), 1296-1311.

Hannay, J., & Jørgensen, M. (2008). The role of deliberate artificial design elements in software engineer ing experiments. *IEEE Transactions on Software Engineering, 34*(2), 242-259.

Harwood, T., & Garry, T. (2015). An investigation into gamification as a customer engagement experience environment. *Journal of Services Marketing*.

Hassan, L. (2017). Governments should play games: Towards a framework for the gamificatio n of civic engagement platforms. *Simulation & Gaming, 48*(2), 249-267.

Hassan, L., & Hamari, J. (2020). Gameful civic engagement: A review of the literature on gamification of e-participation. *Government Information Quarterly, 37*(3), 101461.

He, C., Liu, H., He, L., Lu, T., & Li, B. (2022). More collaboration, less seriousness: Investigating new strategies for promoting youth engagement in government-generated videos during the COVID-19 pandemic in China. *Computers in Human Behavior, 126*, 107019.

Hedges, L. V. (1994). Fixed effects models. *The handbook of research synthesis, 285*, 299.

Henttonen, K., & Blomqvist, K. (2005). Managing distance in a global virtual team: the evolution of trust thr ough technology-mediated relational communication. *Strategic Change, 14*(2), 107-119.

Herbsleb, J. D., & Grinter, R. E. (1999). Architectures, coordination, and distance: Conway's law and beyond. *IEEE software, 16*(5), 63-70.

Herbsleb, J. D., & Mockus, A. (2003). An empirical study of speed and communication in globally distributed software development. *IEEE Transactions on Software Engineering, 29*(6), 481-494.

Herbsleb, J. D., & Moitra, D. (2001). Global software development. *IEEE software, 18*(2), 16-20.

Herranz, E., Guzmán, J. G., de Amescua-Seco, A., & Larrucea, X. (2019). Gamification for software process improvement: a practical approach. *IET Software, 13*(2), 112-121.

Hertel, G., Niedner, S., & Herrmann, S. (2003). Motivation of software developers in Open Source projects: an Internet -based survey of contributors to the Linux kernel. *Research policy, 32*(7), 1159-1177.

Hoegl, M., & Gemuenden, H. G. (2001). Teamwork quality and the success of innovative projects: A theoretical concept and empirical evidence. *Organization science, 12*(4), 435-449.

Hsieh, H. C. L., & Yang, H.-H. (2020). Incorporating gamification into website design to facilitate effective communication. *TheoreTical issues in ergonomics science, 21*(1), 89-111.

Humphrey, W. S. (1989). *Managing the software process*: Addison-Wesley Longman Publishing Co., Inc.

Hunicke, R., LeBlanc, M., & Zubek, R. (2004). *MDA: A formal approach to game design and game research*.

Huotari, K., & Hamari, J. (2017). A definition for gamification: anchoring gamification in the service m arketing literature. *Electronic Markets, 27*(1), 21-31.

Ilhan, A. E., Sener, B., & Hacihabiboglu, H. (2022). Improving Sleep-Wake Behaviors Using Mobile App Gamification. *Entertainment Computing, 40*, 100454.

Ivanova, G., Kozov, V., & Zlatarov, P. (2019). *Gamification in software engineering education*.

Janzen, D., & Saiedian, H. (2008). Does test-driven development really improve software design quality? *IEEE software, 25*(2), 77-84.

Jazayeri, M. (2004). *The education of a software engineer*.

Jeffries, R., & Melnik, G. (2007). Guest Editors' Introduction: TDD–The Art of Fearless Programming. *IEEE software, 24*(3), 24-30.

Jobson, J. D. (2012). *Applied multivariate data analysis: regression and experimental design*: Springer Science & Business Media.

Johnson, L., Becker, S. A., Estrada, V., & Freeman, A. (2014). *NMC horizon report: 2014 K*: The New Media Consortium.

Just, S., Herzig, K., Czerwonka, J., & Murphy, B. (2016, October). Switching to Git: the good, the bad, and the ugly. In *2016 IEEE 27th International Symposium on Software Reliability Engineering (ISSRE)* (pp. 400-411). IEEE.

Kalogiannakis, M., Papadakis, S., & Zourmpakis, A.-I. (2021). Gamification in science education. A systematic review of the literatu re. *Education Sciences, 11*(1), 22.

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

Karlsson, L., Dahlstedt, A. G., Regnell, B., Dag, J. N., & Persson, A. (2007). Requirements engineering challenges in market-driven software developm ent–An interview study with practitioners. *Information and Software technology, 49*(6), 588-604.

Karlström, D., & Runeson, P. (2006). Integrating agile software development into stage-gate managed product development. *Empirical Software Engineering, 11*(2), 203-225.

Kasurinen, J., & Knutas, A. (2018). Publication trends in gamification: A systematic mapping study. *Computer Science Review, 27*, 33-44.

Katz, R. (1982). The effects of group longevity on project communication and performanc e. *Administrative science quarterly*, 81-104.

Keuler, T., Knodel, J., Naab, M., & Rost, D. (2012). *Architecture Engagement Purposes: Towards a Framework for Planning" Just Enough"-Architecting in Software Engineering.* Paper presented at the 2012 Joint Working IEEE/IFIP Conference on Software Architecture and European Conference on Software Architecture.

Khanam, Z., & Ahsan, M. N. (2017). Evaluating the effectiveness of test driven development: advantages an d pitfalls. *International Journal of Applied Engineering Research, 12*(18), 7705-7716.

Kiani, Z. U. R., Smite, D., & Riaz, A. (2013). *Measuring awareness in cross-team collaborations–distance matters*.

Kim, Y., Kim, J., Jeon, H., Kim, Y. H., Song, H., Kim, B., & Seo, J. (2020). Githru: visual analytics for understanding software development history through git metadata analysis. *IEEE Transactions on Visualization and Computer Graphics*, *27*(2), 656-666.

Klock, A. C. T., Gasparini, I., Pimenta, M. S., & Hamari, J. (2020). Tailored gamification: A review of literature. *International Journal of Human-Computer Studies, 144*, 102495.

Koivisto, J., & Hamari, J. (2019). The rise of motivational information systems: A review of gamification research. *International Journal of Information Management, 45*, 191-210.

Kolassa, C., Riehle, D., & Salim, M. A. (2013). *The empirical commit frequency distribution of open source projects*.

Korkala, M., & Abrahamsson, P. (2007). *Communication in distributed agile development: A case study*.

Kraut, R. E., & Streeter, L. A. (1995). Coordination in software development. *Communications of the ACM, 38*(3), 69-82.

Krishnamoorthy, K. (2016). *Handbook of statistical distributions with applications*: CRC Press.

Landers, R. N. (2014). Developing a theory of gamified learning: Linking serious games and ga mification of learning. *Simulation & gaming, 45*(6), 752-768.

Landers, R. N., & Callan, R. C. (2011). Casual social games as serious games: The psychology of gamification i n undergraduate education and employee training. In *Serious games and edutainment applications* (pp. 399-423): Springer.

Landman, D., Serebrenik, A., Bouwers, E., & Vinju, J. J. (2016). Empirical analysis of the relationship between CC and SLOC in a large corpus of Java methods and C functions. *Journal of Software: Evolution and Process, 28*(7), 589-618.

Lane, E. S., & Harris, S. E. (2015). A new tool for measuring student behavioral engagement in large univer sity classes. *Journal of College Science Teaching, 44*(6), 83-91.

Latulipe, C., Long, N. B., & Seminario, C. E. (2015). *Structuring flipped classes with lightweight teams and gamification*.

Law, F. L., Kasirun, Z. M., & Gan, C. K. (2011). *Gamification towards sustainable mobile application*.

Lee, T., Lee, J. B., & In, H. P. (2013). A study of different coding styles affecting code readability. *International Journal of Software Engineering and Its Applications, 7*(5), 413-422.

Legaki, N.-Z., Xi, N., Hamari, J., Karpouzis, K., & Assimakopoulos, V. (2020). The effect of challenge-based gamification on learning: An experiment in the context of statistics education. *International journal of human-computer studies, 144*, 102496.

Lenberg, P., Feldt, R., & Wallgren, L. G. (2015). *Human factors related challenges in software engineering–an industrial perspective*.

Levin, S., & Yehudai, A. (2017). *Boosting automatic commit classification into maintenance activities b y utilizing source code changes*.

Li, X. (2018). *A method to support gamification design practice with motivation analy sis and goal modeling*.

Li, X., & Chu, S. K. W. (2021). Exploring the effects of gamification pedagogy on children's reading: A mixed-method study on academic performance, reading-related mentalit y and behaviors, and sustainability. *British Journal of Educational Technology, 52*(1), 160-178.

Liu, M., Calvo, R. A., Pardo, A., & Martin, A. (2014). Measuring and visualizing students' behavioral engagement in writing a ctivities. *IEEE Transactions on learning technologies, 8*(2), 215-224.

Lounis, S., Kotsopoulos, D., Bardaki, C., Papaioannou, T. G., & Pramatari, K. (2017). *Waste no more: Gamification for energy efficient behaviour at the work place*.

Lycett, M., Macredie, R. D., Patel, C., & Paul, R. J. (2003). Migrating agile methods to standardized development practice. *Computer, 36*(6), 79-85.

M. Pereira, I., JP Amorim, V., A. Cota, M., & C. Gonçalves, G. (2017). *Gamification use in agile project management: an experience report.* Paper presented at the Agile Methods: 7th Brazilian Workshop, WBMA 2016, Curitiba, Brazil, November 7-9, 2016, Revised Selected Papers 7.

Machuca-Villegas, L., & Gasca-Hurtado, G. P. (2018). *Gamification for improving software project management processes: a sy stematic literature review*.

Madeyski, L., & Biela, W. (2007). *Capable Leader and Skilled and Motivated Team Practices to Introduce e Xtreme Programming*.

Majuri, J., Koivisto, J., & Hamari, J. (2018). *Gamification of education and learning: A review of empirical literatu re*.

Mäkinen, S., & Münch, J. (2014). *Effects of test-driven development: A comparative analysis of empirica l studies*.

Malhotra, R., & Chug, A. (2016). Software maintainability: Systematic literature review and current tre nds. *International Journal of Software Engineering and Knowledge Engineerin g, 26*(8), 1221-1253.

Mallardo, T., Calefato, F., Lanubile, F., & Damian, D. (2007). *The effects of communication mode on distributed requirements negotiat ions*.

Malone, T. W., & Crowston, K. (1994). The interdisciplinary study of coordination. *ACM Computing Surveys (CSUR), 26*(1), 87-119.

Mamone, S. (1994). The IEEE standard for software maintenance. *ACM SIGSOFT Software Engineering Notes, 19*(1), 75-76.

Manohar, P. A., Acharya, S., Wu, P., Hansen, M., Ansari, A., & Schilling, W. (2015). Case Studies for Enhancing Student Engagement and Active Learning in S oftware V&V Education. *Journal of Education and Learning, 4*(4), 39-52.

Martinez, M., & Monperrus, M. (2019, May). Coming: A tool for mining change pattern instances from git commits. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)* (pp. 79-82). IEEE.

Mariano, R. V. R., Santos, G. E., Almeida, M. V., & Brandão, W. C. (2019). *Feature changes in source code for commit classification into maintena nce activities*.

Markos, S., & Sridevi, M. S. (2010). Employee engagement: The key to improving performance. *International journal of business and management, 5*(12), 89.

Matsubara, P. G. F., & Da Silva, C. L. C. (2017). *Game elements in a software engineering study group: a case study.* Paper presented at the 2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering Education and Training Track (ICSE-SEET).

Matturro, G., Raschetti, F., & Fontán, C. (2019). A Systematic Mapping Study on Soft Skills in Software Engineering. *J. Univers. Comput. Sci., 25*(1), 16-41.

Mazarakis, A. (2015). Using gamification for technology enhanced learning: The case of feedb ack mechanisms. *Bull. IEEE Tech. Comm. Learn. Technol, 17*(4), 6-9.

Mills, J. E., Treagust, D. F., & others. (2003). Engineering education—Is problem-based or project-based learning the a nswer. *Australasian journal of engineering education, 3*(2), 2-16.

Mishra, A., Ercil Cagiltay, N., & Kilic, O. (2007). Software engineering education: some important dimensions. *European Journal of Engineering Education, 32*(3), 349-361.

Misra, S., Kumar, V., Kumar, U., Fantazy, K., & Akhter, M. (2012). Agile software development practices: evolution, principles, and criticisms. *International Journal of Quality & Reliability Management*.

Mittal, H., & Bhatia, P. (2009). Software maintainability assessment based on fuzzy logic technique. *ACM SIGSOFT Software Engineering Notes, 34*(3), 1-5.

Moldon, L., Strohmaier, M., & Wachs, J. (2021). *How gamification affects software developers: Cautionary evidence from a natural experiment on github.* Paper presented at the 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE).

Monteiro, R. H. B., Almeida Souza, M. c. R., Oliveira, S. R. B., Santos Portela, C., & Cristo Lobato, C. E. (2021). *The diversity of gamification evaluation in the software engineering e ducation and industry: Trends, comparisons and gaps*.

Monterrat, B., Lavoué, É., & George, S. (2014). *Toward an adaptive gamification system for learning environments*.

Morschheuser, B., Hassan, L., Werder, K., & Hamari, J. (2018). How to design gamification? A method for engineering gamified software. *Information and Software Technology, 95*, 219-237.

Mugridge, R. (2003). *Challenges in teaching test driven development*.

Muller, M. M., & Tichy, W. F. (2001). *Case study: extreme programming in a university environment*.

Munir, H., Wnuk, K., Petersen, K., & Moayyed, M. (2014). *An experimental evaluation of test driven development vs. test-last de velopment with industry professionals*.

Muntean, C. I. (2011). *Raising engagement in e-learning through gamification*.

Muszyńska, K. (2020). Gamification of communication and documentation processes in project t eams. *Procedia Computer Science, 176*, 3645-3653.

Myers, G. J., Sandler, C., & Badgett, T. (2011). *The art of software testing*: John Wiley & Sons.

Nah, F. F.-H., Telaprolu, V. R., Rallapalli, S., & Venkata, P. R. (2013). *Gamification of education using computer games*.

Nah, F. F.-H., Zeng, Q., Telaprolu, V. R., Ayyappa, A. P., & Eschenbrenner, B. (2014). *Gamification of education: a review of literature*.

Namara, M., Sloan, H., & Knijnenburg, B. P. (2022). The Effectiveness of Adaptation Methods in Improving User Engagement a nd Privacy Protection on Social Network Sites. *Proceedings on Privacy Enhancing Technologies, 2022*(1), 629-648.

Nanthaamornphong, A., & Carver, J. C. (2017). Test-Driven Development in scientific software: a survey. *Software Quality Journal, 25*(2), 343-372.

Nguyen, V., Deeds-Rubin, S., Tan, T., & Boehm, B. (2007). *A SLOC counting standard.* Paper presented at the Cocomo ii forum.

Nundlall, C., & Nagowah, S. D. (2021). Task allocation and coordination in distributed agile software develop ment: a systematic review. *International Journal of Information Technology, 13*(1), 321-330.

Oman, P., & Hagemeister, J. (1992). *Metrics for assessing a software system's maintainability*.

Orji, F. A., Vassileva, J., & Greer, J. (2021). Evaluating a Persuasive Intervention for Engagement in a Large Univers ity Class. *International Journal of Artificial Intelligence in Education, 31*(4), 700-725.

Ostberg, J.-P., & Wagner, S. (2014). *On automatically collectable metrics for software maintainability eval uation*.

Ouhbi, S., & Pombo, N. (2020). *Software engineering education: Challenges and perspectives*.

Paasivaara, M., & Lassenius, C. (2003). Collaboration practices in global inter-organizational software develo pment projects. *Software Process: Improvement and Practice, 8*(4), 183-199.

Paasivaara, M., & Lassenius, C. (2010). Using Scrum practices in GSD projects. In *Agility across time and space* (pp. 259-278): Springer.

Pakarinen, A., Parisod, H., Linden, I., Aromaa, M. E., Smed, J., Leppänen, V., & Salanterä, S. (2017). *Usability of a gamified application to promote family wellbeing in chi ld health clinics*.

Pancur, M., Ciglaric, M., Trampus, M., & Vidmar, T. (2003). *Towards empirical evaluation of test-driven development in a universit y environment*.

Papis, B. K., Grochowski, K., Subzda, K., & Sijko, K. (2020). Experimental evaluation of test-driven development with interns workin g on a real industrial project. *IEEE Transactions on Software Engineering*.

Parnas, D. L., & Clements, P. C. (1986). A rational design process: How and why to fake it. *IEEE Transactions on Software Engineering*(2), 251-257.

Paula Porto, D., Ferrari, F. C., & Fabbri, S. C. P. F. (2019). *Improving project manager decision with gamification: An experience re port*.

Paula Porto, D., Jesus, G. M., Ferrari, F. C., & Fabbri, S. C. P. F. (2021). Initiatives and challenges of using gamification in software engineeri ng: A Systematic Mapping. *Journal of Systems and Software, 173*, 110870.

Pedreira, O., García, F., Brisaboa, N., & Piattini, M. (2015). Gamification in software engineering–A systematic mapping. *Information and software technology, 57*, 157-168.

Pedreira, O., García, F., Piattini, M., Cortiñas, A., & Cerdeira-Pena, A. (2020). An architecture for software engineering gamification. *Tsinghua Science and Technology, 25*(6), 776-797.

Perera, P., Silva, R., & Perera, I. (2017). *Improve software quality through practicing DevOps*.

Persson, N., & Isberg, A. T. (2019). How reliable is Test-Driven Development.

Perez De Rosso, S., & Jackson, D. (2013, October). What's wrong with git? A conceptual design analysis. In *Proceedings of the 2013 ACM international symposium on New ideas, new paradigms, and reflections on programming & software* (pp. 37-52).

Pesare, E., Roselli, T., Corriero, N., & Rossano, V. (2016). Game-based learning and gamification to promote engagement and motivat ion in medical learning contexts. *Smart Learning Environments, 3*(1), 1-21.

Pikkarainen, M., Haikara, J., Salo, O., Abrahamsson, P., & Still, J. (2008). The impact of agile practices on communication in software development. *Empirical Software Engineering, 13*(3), 303-337.

Pinto, M. B., & Pinto, J. K. (1990). Project team communication and cross-functional cooperation in new pro gram development. *Journal of Product Innovation Management: an international publication of the product development & management association, 7*(3), 200-212.

Piorkowski, D., Park, S., Wang, A. Y., Wang, D., Muller, M., & Portnoy, F. (2021). How ai developers overcome communication challenges in a multidiscipli nary team: A case study. *Proceedings of the ACM on Human-Computer Interaction, 5*(CSCW1), 1-25.

Prause, C. R., & Jarke, M. (2015). *Gamification for enforcing coding conventions*.

Prause, C. R., Nonnen, J., & Vinkovits, M. (2012). *A Field Experiment on Gamification of Code Quality in Agile Developmen t*.

Qahri-Saremi, H., & Turel, O. (2016). School engagement, information technology use, and educational develop ment: An empirical investigation of adolescents. *Computers & Education, 102*, 65-78.

Qiu, H. S., Nolte, A., Brown, A., Serebrenik, A., & Vasilescu, B. (2019). *Going farther together: The impact of social capital on sustained part icipation in open source*.

Rafique, Y., & Mišić, V. B. (2012). The effects of test-driven development on external quality and productivity: A meta-analysis. *IEEE Transactions on Software Engineering, 39*(6), 835-856.

Rahy, S., & Bass, J. (2018). *Information flows at inter-team boundaries in agile information system s development*.

Ramsey, J. B. (1969). Tests for specification errors in classical linear least-squares regre ssion analysis. *Journal of the Royal Statistical Society: Series B (Methodological), 31*(2), 350-371.

Rauf, I., Lopez, T., Sharp, H., Petre, M., Tun, T., Levine, M., . . . Nuseibeh, B. (2022). *Influences of developers' perspectives on their engagement with security in code.* Paper presented at the

Proceedings of the 15th International Conference on Cooperative and Human Aspects of Software Engineering.

Razina, E., & Janzen, D. S. (2007). *Effects of dependency injection on maintainability*.

Reddy, B. R., & Ojha, A. (2019). Performance of Maintainability Index prediction models: a feature sele ction based study. *Evolving Systems, 10*(2), 179-204.

Ren, W., Barrett, S., & Das, S. (2020). *Toward gamification to software engineering and contribution of softwa re engineer*.

Riemer, V., & Schrader, C. (2016). Impacts of behavioral engagement and self-monitoring on the developmen t of mental models through serious games: Inferences from in-game meas ures. *Computers in Human Behavior, 64*, 264-273.

Rocha, F. G., Souza, L. S., Silva, T. S., & Rodríguez, G. (2021). *Enhancing the Student Learning Experience by Adopting TDD and BDD in C ourse Projects*.

Rodrigues, P., Souza, M., & Figueiredo, E. (2018). *Games and gamification in software engineering education: A survey wit h educators*.

Roman, A., & Mnich, M. (2021). Test-driven development with mutation testing–an experimental study. *Software Quality Journal, 29*(1), 1-38.

Rosen, C., Grawi, B., & Shihab, E. (2015). *Commit guru: analytics and risk prediction of software commits*.

Runeson, P. (2006). A survey of unit testing practices. *IEEE software, 23*(4), 22-29.

Ruvimova, A., Lill, A., Gugler, J., Howe, L., Huang, E., Murphy, G., & Fritz, T. (2022). *An exploratory study of productivity perceptions in software teams.* Paper presented at the Proceedings of the 44th International Conference on Software Engineering.

Saleem, A. N., Noori, N. M., & Ozdamli, F. (2021). Gamification applications in E-learning: a literature review. *Technology, Knowledge and Learning*, 1-21.

Sanchez, D. R., Langer, M., & Kaur, R. (2020). Gamification in the classroom: Examining the impact of gamified quizzes on student learning. *Computers & Education, 144*, 103666.

Santos, A., Vegas, S., Dieste, O., Uyaguari, F., Tosun, A., Fucci, D., . . . others. (2021). A family of experiments on test-driven development. *Empirical Software Engineering, 26*(3), 1-53.

Santos, E. D., & Oliveira, S. R. B. (2018). *Gamification and Evaluation the Use of the Function Points Analysis Te chnique in Software Quality Subjects: The Experimental Studies*.

Santos, V., Goldman, A., & De Souza, C. R. B. (2015). Fostering effective inter-team knowledge sharing in agile software dev elopment. *Empirical Software Engineering, 20*(4), 1006-1051.

Sarwar, M. I., Tanveer, W., Sarwar, I., & Mahmood, W. (2008). *A comparative study of mi tools: defining the roadmap to mi tools stan dardization*.

Schaufeli, W. B. (2013a). What is engagement. *Employee engagement in theory and practice, 15*, 321.

Schaufeli, W. B. (2013b). What is engagement? In *Employee engagement in theory and practice* (pp. 29-49): Routledge.

Schaufeli, W. B., & Bakker, A. B. (2003). UWES–Utrecht work engagement scale: test manual. *Unpublished Manuscript: Department of Psychology, Utrecht University, 8*.

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

Shi, L., & Cristea, A. I. (2016). *Motivational gamification strategies rooted in self-determination theory for social adaptive e-learning.* Paper presented at the Intelligent Tutoring Systems: 13th International Conference, ITS 2016, Zagreb, Croatia, June 7-10, 2016. Proceedings 13.

Shihab, E., Bettenburg, N., Adams, B., & Hassan, A. E. (2009). *On the central role of mailing lists in open source projects: An explo ratory study*.

Shull, F., Basili, V., Boehm, B., Brown, A. W., Costa, P., Lindvall, M., . . . Zelkowitz, M. (2002). *What we have learned about fighting defects*.

Simões, J., Redondo, R. D. a., & Vilas, A. F. (2013). A social gamification framework for a K-6 learning platform. *Computers in Human Behavior, 29*(2), 345-353.

Singer, L., & Schneider, K. (2012). *It was a bit of a race: Gamification of version control*.

Singleton, R. A., Straits, B. C., Straits, M., & McAlister, R. J. (1999). Approaches to Social Research: Oxford University Press. *New York and Oxford, 9*.

Sisomboon, W., Phakdee, N., & Denwattana, N. (2019). *Engaging and motivating developers by adopting scrum utilizing gamific ation*.

Siutila, M. (2018). *The gamification of gaming streams*.

Sommerville, I. (2015). Software engineering 10th Edition. *ISBN-10, 137035152*, 18.

Stapel, K., & Schneider, K. (2014). Managing knowledge on communication and information flow in global sof tware projects. *Expert Systems, 31*(3), 234-252.

Stellman, A., & Greene, J. (2005). *Applied software project management*: " O'Reilly Media, Inc.".

Stock, J. H., & Watson, M. W. (2008). Heteroskedasticity-robust standard errors for fixed effects panel data regression. *Econometrica, 76*(1), 155-174.

Storey, M.-A., Zagalsky, A., Figueira Filho, F., Singer, L., & German, D. M. (2016). How social and communication channels shape and challenge a participat ory culture in software development. *IEEE Transactions on Software Engineering, 43*(2), 185-204.

Suh, A., Cheung, C. M. K., Ahuja, M., & Wagner, C. (2017). Gamification in the workplace: The central role of the aesthetic experience. *Journal of Management Information Systems, 34*(1), 268-305.

Sukale, R., & Pfaff, M. S. (2014). QuoDocs: Improving developer engagement in software documentation thro ugh gamification. In *CHI'14 Extended Abstracts on Human Factors in Computing Systems* (pp. 1531-1536).

Svensson, H., & Höst, M. (2005). *Views from an organization on how agile development affects its collab oration with a software development team*.

Tahmasbi, N., & Fuchsberger, A. (2018). Gamification for Achieving Sustained Engagement in Programming Classes.

Thomas, C., & Berkling, K. (2013). *Redesign of a gamified software engineering course*.

Tizard, J., Rietz, T., Liu, X., & Blincoe, K. (2022). Voice of the users: an extended study of software feedback engagement. *Requirements Engineering, 27*(3), 293-315.

Toda, A. M., Valle, P. H., & Isotani, S. (2018). *The dark side of gamification: An overview of negative effects of gamification in education.* Paper presented at the Higher Education for All. From Challenges to Novel Technology-Enhanced Solutions: First International Workshop on Social, Semantic, Adaptive and Gamification Techniques and Technologies for Distance Learning, HEFA 2017, Maceió, Brazil, March 20–24, 2017, Revised Selected Papers 1.

Tosun, A., Ahmed, M., Turhan, B., & Juristo, N. (2018). *On the effectiveness of unit tests in test-driven development*.

Tosun, A., Dieste, O., Fucci, D., Vegas, S., Turhan, B., Erdogmus, H., . . . others. (2017). An industry experiment on the effects of test-driven development on ex ternal quality and productivity. *Empirical Software Engineering, 22*(6), 2763-2805.

Towey, D., Chen, T. Y., Kuo, F.-C., Liu, H., & Zhou, Z. Q. (2016). *Metamorphic testing: A new student engagement approach for a new softw are testing paradigm*.

Treude, C., & Storey, M.-A. (2010). Work item tagging: Communicating concerns in collaborative software de velopment. *IEEE Transactions on Software Engineering, 38*(1), 19-34.

Trinidad, M., Ruiz, M., & Calderón, A. (2021). A bibliometric analysis of gamification research. *IEEE Access, 9*, 46505-46544.

Tsunoda, M., & Yumoto, H. (2018). *Applying gamification and posing to software development*.

Unkelos-Shpigel, N., & Hadar, I. (2015). *Gamifying software engineering tasks based on cognitive principles: Th e case of code review*.

Uskov, V., & Sekar, B. (2014). *Gamification of software engineering curriculum*.

van Hal, S., Post, M., & Wendel, K. (2019). Generating commit messages from git diffs. *arXiv preprint arXiv:1911.11690*.

Velmourougan, S., Dhavachelvan, P., Baskaran, R., & Ravikumar, B. (2014). *Software development life cycle model to improve maintainability of so ftware applications*.

Veltsos, J. R. (2017). Gamification in the business communication course. *Business and Professional Communication Quarterly, 80*(2), 194-216.

Voas, J., & Agresti, W. W. (2004). Software quality from a behavioral perspective. *IT professional, 6*(4), 46-50.

Weinberg, G. M. (1971). *The psychology of computer programming* (Vol. 29): Van Nostrand Reinhold New York.

Wen, M., Wu, R., & Cheung, S.-C. (2016). *Locus: Locating bugs from software changes*.

Werbach, K., & Hunter, D. (2020). *For the Win, Revised and Updated Edition: The Power of Gamification an d Game Thinking in Business, Education, Government, and Social Impact*: University of Pennsylvania Press.

Williams, L., Kudrjavets, G., & Nagappan, N. (2009). *On the effectiveness of unit test automation at microsoft*.

Williams, L., Maximilien, E. M., & Vouk, M. (2003). *Test-driven development as a defect-reduction practice*.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in software engineering*: Springer Science & Business Media.

Wongso, O., Rosmansyah, Y., & Bandung, Y. (2014). *Gamification framework model, based on social engagement in e-learning 2.0*.

Yagüe, A., Garbajosa, J., Díaz, J., & González, E. (2016). An exploratory study in communication in Agile Global Software Develop ment. *Computer Standards & Interfaces, 48*, 184-197.

Yusifoğlu, V. G., Amannejad, Y., & Can, A. B. (2015). Software test-code engineering: A systematic mapping. *Information and Software Technology, 58*, 123-147.

Zahra, K., Azam, F., Ilyas, F., Faisal, H., Ambreen, N., & Gondal, N. (2017). *Success factors of organizational change in software process improveme nt: A systematic literature review*.

Zelkowitz, M. V. (1978). Perspectives in software engineering. *ACM Computing Surveys (CSUR), 10*(2), 197-216.

Zhu, L., Shan, M., & Hwang, B.-G. (2018). Overview of design for maintainability in building and construction re search. *Journal of Performance of Constructed Facilities, 32*(1), 04017116.

Zichermann, G., & Cunningham, C. (2011). *Gamification by design: Implementing game mechanics in web and mobile apps*: " O'Reilly Media, Inc.".

Zielinski, K., & Szmuc, T. (2005). Preliminary analysis of the effects of pair programming and test-drive n development on the external code quality. *Frontiers in Artificial Intelligence and Applications*, 113.

Appendix

A. Variables Used in Studies

a. Variables Used in Observational Study:

Type	Variables	Data Type
TDD-Sequence	SEQ	Dummy
TDD-Iteration	The number of development cycle (Cycle)	Continuous
TDD-Iteration	Cycle created following TDD style (TC)	Continuous
TDD-Iteration	Cycle created following other styles (GC)	Continuous
Engagement-Commit number	Number of Commits (NC)	Continuous
Engagement-Commit number	Number of Product Commits (NPC)	Continuous
Engagement-Commit number	Number of Test Commits (NTC)	Continuous
Engagement-Commit number	Number of New Test Commits (NNTC)	Continuous
Engagement-Commit number	Number of Maintenance Test Commits (NMTC)	Continuous
Engagement-Commit frequency	Frequency of Update Code (FEQ)	Continuous
Engagement-Commit frequency	Frequency of Update Product Code (FP)	Continuous
Engagement-Commit frequency	Frequency of Update Test Code (FT)	Continuous
Engagement-Commit frequency	Frequency of Update New Test Code (FNT)	Continuous
Engagement-Commit frequency	Frequency of Update Maintenance Test Code (FMT)	Continuous
Engagement	New versus Maintain (N_M)	Dummy
Maintainability	Cyclomatic Complexity (CC)	Continuous
Maintainability	Maintainability Index (MI)	Continuous
Control Variable-Readability	Halstead Difficulty (HD)	Continuous
Control Variable-Different Language	Object-Oriented Language (lan_OO)	Dummy
Control Variable-Different Task	LCA_DAG	Dummy
Control Variable-Size	Line of Code (LOC)	Continuous

Appendix Table A Variables Description

b. Variables Used in Group Experiment:

Type	Variables	Data Type
TDD-Iteration	The number of development cycle (Cycle)	Continuous
Engagement-Commit number	Number of Commits (NC)	Continuous
Engagement-Commit frequency	Frequency of Update Code (FEQ)	Continuous
Maintainability	Cyclomatic Complexity (CC)	Continuous
Maintainability	Mean Cyclomatic Complexity (CC)	Continuous
Maintainability	Maintainability Index (MI)	Continuous
Maintainability	Mean Maintainability Index (MI)	Continuous
Control Variable-Readability	Halstead Difficulty (HD)	Continuous
Control Variable-Readability	Comment Ratio (CR)	Continuous
Gamification	Gamification_1	Dummy

Type	Variables	Data Type
Gamification	Gamification_2	Dummy
Gamification	Gamification_3	Dummy

c. Variables Used in Individual Experiment:

Type	Variables	Data Type
TDD-Iteration	The number of development cycle (Cycle)	Continuous
TDD-Iteration	The number of test cases (Test)	Continuous
Engagement-Commit number	Number of Commits (NC)	Continuous
Engagement-Commit number	Number of Product Commits (NPC)	Continuous
Engagement-Commit number	Number of Test Commits (NTC)	Continuous
Maintainability	Cyclomatic Complexity (CC)	Continuous
Maintainability	Maintainability Index (MI)	Continuous
Maintainability	Function with MI (FM)	Continuous
Maintainability	Function with CC (FC)	Continuous
Control Variable-Readability	Halstead Difficulty (HD)	Continuous
Control Variable-Readability	Comment Ratio (CR)	Continuous
Control Variable-Different Language	Object-Oriented Language (lan_OO)	Dummy
Control Variable-Size	Line of Code (LOC)	Continuous
Control Variable-Experience	User Experience (use_exp)	Continuous
Gamification	Gamification	Dummy
Gamification	Leaderboard	Dummy
Gamification	All	Dummy
Gamification	Extra	Dummy
Gamification	Feedback	Dummy
Gamification	Continue	Dummy

2. Ethics Documents

a. Participate Consent

Gamification on Software Engineering Process - Participation Information

I would like to invite you to take part in a learning opportunity regards Test-driven Development (TDD). Before you decide you need to understand why this program is being done and what it would involve for you. Please take time to read the following information carefully. Ask questions if anything you read is not clear or if you would like more information before deciding whether you will take part or not.

If you agree to participate this program, I will engage with team engineers to teach and promote the Test-driven Development method, and I will seek to use gaming strategies to encourage that behaviors. I will ask you to sign up to this program and asking for your agreement to participate. Please read the following content, if you are happy to participate, complete the participation consent form.

WHO I AM AND WHAT THIS PROGRAM IS ABOUT?

I am a PhD candidate of School of Computer Science and Statistics in Trinity College. My research field is social software engineering. In this program, I would like to validate that gamification has positively impacts on software engineers' behaviors and engagement. The process of the program is to establish baseline of behaviors, introducing gamification, and measure the change baseline of behaviors.

The program aims to encourage developer following Test-Driven Development, a cyclic development technique which means writing test case first, writing code with a short duration. The

program is being undertaken as part of the course of study state. The program does not require participant generating additional code.

THE DETAILS OF THIS PROGRAM

This program tries to encourage participants apply TDD during their development by gamification method.

Ideally, the good working process of TDD is in small and rapid iterations, which means each cycle or iteration coincides with the implementation of a tiny feature. So, in this program, the participants will be encouraged to write test case first and development with rapid development iterations. The benefit of applying TDD is that the TDD method has approximately 40% fewer defects than the traditional fashion. There is significant improvement in the code quality by using TDD method, compared to non-TDD method.

Gamification is a concept that apply game elements in non-game context. The game elements include points, leaderboard, feedback and etc. Gamification might be an effective way to encourage behavior change and improve the engagement. In this program, there are five gamification rules to help participants understanding TDD and improve the working performance of software engineering. The gamification rules are shown follow:

1. Finishing a cycle (a unit test and product code), participant will get 2 points.
2. Generating a failing unit test before function code, and participant will get 1 point when unit test pass after finishing function code. (Finishing a cycle with TDD style)
3. Writing one commit of test case, participant will get 0.5 points, if writing a commit of creating new test case, earn extra 2 point.
4. Participant will get 5 points if your repository has one star. Everyone is expected to star your peers (more than one peer).

5. Participants will be ranked and participant position in the leaderboard will be released with gamification point. The points will be accumulated.
    - The highest ranking in total will earn 15 points, but you can get extra bonus (10 points) if your coverage reaches 70%.
    - The second highest will get 14 points, etc.
    - The ranking will be refreshed every week, and the points will be accumulated.

The participants will be noticed that their current points and their position in the leaderboard. At the end of week 12, every five participants will be grouped in one leaderboard based on their point. The participants in different leaderboard will get real-world reward including meal, drink and snack.

Leaderboard 1: 30, 25 and 20 euro (meal voucher).

Leaderboard 2: 15, 12 and 9 euro (drink voucher).

Leaderboard 3: 6, 4 and 2 euro (snack voucher).

The notification which will be released every Monday and Thursday by email, includes current point and participant position in the leaderboard.

The gamification points and the position of leaderboard will not have any **IMPACT** on the module **GRADE**.

The timeline of the process is shown as follow:

Week 8, Tuesday:

1. Introducing TDD and the benefit of applying TDD during the development

2. What is gamification and introduce gamification rules

Week 9, Monday and Thursday morning: release gamification points for individual and the participant's position in the leaderboard. (release point every Monday and Thursday morning)

Week 10: provide some feedback to individual about how to apply TDD more efficient.

Week 12: invited students to answer the questionnaire about their feeling of gamification and TDD and release final leaderboard position.

	Week 8	Week 9	Week 10	Week 11	Week 12
Mon		Score and position	Score and position	Score and position	Score and position
Tue					
Wed					
Thu		Score and position	Score and position	Score and position	Score and position
Fri	Introduce TDD & Gamification		Feedback & suggestion		Questionnaire & Final Leaderboard

WHY HAVE YOU BEEN INVITED TO TAKE PART?

The program focuses on experienced software engineers. The participants have basic programming skill and experience. The participants may adopt various development methods during the project development of the module, and they may have higher motivation of trying TDD than people for the other courses.

DO YOU HAVE TO TAKE PART?

Participating in this program is completely **VOLUNTARY**. There is no obligation on you to take part in this program, it is entirely voluntary. You have the right to refuse to participate, answer the question, and withdraw from the program at any time without any consequence whatsoever. Your participation or non-participation has **NO IMPACT** on your grading in this module.

WHAT ARE THE POSSIBLE RISKS AND BENEFITS OF TAKING PART?

The benefits for the participants include accumulating the experience of using TDD, which helps the developers gain competitive advantage when they come to job hunting. For example, IBM published a Test Automation Engineer job description: As a test engineer work with developers employing TDD to develop desired outcomes. Many software engineering jobs prefer developers to have TDD experience. Another benefit for participants is that, if the student chooses to continue the research study, TDD is also a popular research topic. In google scholar, there are 91,100 papers about Test-Driven Development by 2020.

Although it is more likely to increase code quality by using TDD, it cannot guarantee this improvement. The potential risk of joining this work is that participants may need to put more efforts on testing than others, including writing test cases before functional code, setup testing environment and so on. In addition, the participants may feel unhappy to know their position in the leaderboard, even if their personal information is anonymous.

WHO SHOULD YOU CONTACT FOR FURTHER INFORMATION?

If you have any question, please feel free contact me:

Wei Ren

School of Computer Science and Statistics at Trinity College Dublin

renw@tcd.ie

Supervisor detail:

Professor Stephen Barrett

stephen.barrett@tcd.ie

Tel. +353 (0)1 8962730

THANK YOU VERY MUCH FOR YOUR READING!

b. Data Consent

Gamification on Software Engineering Process - Data Information

You are currently engaging in Test-driven Development (TDD) with Wei Ren, and you are learning to how to do TDD that being done in the form of gamification. We would like to conduct formal research and publish based on this activity. In order for us to do this, we must have your additional consent. So, therefore we are turning now to ask explicitly for your consent to be a subject in a research study. Above beyond giving your consent, ***NO ADDITIONAL EFFORT***, however we do want you to review this document to be sure that you are happy. Consent for Participation the research is entirely optional, and it has ***NO*** impact and will ***NOT*** influence your grading in CSU33D06. If you wish, you may refuse this consent request and continue participate the TDD activity.

If you agree to participate this study, I will ask you to sign up to this study and asking for your agreement to share your data for research purpose ***ONLY***. Please read the following content carefully, ask questions if anything you read is not clear or if you would like more information before deciding whether you will take part or not. If you are happy to share the data, please complete the Data Consent Form.

WHAT WILL INVOLVE WHEN TAKING PART?

We ask for your consent if you agree the data that we are using to do the gamification work and can be used in research studies. So, there is no additional requirements or activities form you. As part of your participation and Test-driven development work. My research is concerned with study in the effective gamification in software engineering process. I intend to use the data that you allow us to include in research on the subject. We expect that a measurable difference in performance based on the application of gamification strategy. The specific data we would gather and the way which we would use these data (store, use and publish) is detailed in appendix 1. We would like to collect following data:

-Your email address and GitHub ID

-Commits history on GitHub which including:
    Commit description
    Commit time
    Contributor ID of the commit

- Source Code on GitHub for analysis code quality which is code quality data including:
    Source code Maintainability Index (MI)
    Source code Cyclomatic Complexity
    Source code Halstead science metrics
    Test cases coverage

-Questionnaire

For consent sharing these data, you ***ONLY*** need to share your ***EMAIL*** address and ***ACCESS AUTHORITY*** of your GitHub repository. To protect your personal data, I will use function to anonymize your email address and GitHub ID. The code quality in this program associates with individual commit and reflect the coding performance of individual person. Thus, the code quality is only connected with GitHub ID which is anonymized to protect your personal data.

I will get your permission to access your GitHub repository via email in week nine. Then, I will access your commits to generate your gamification score. Also, I will analysis your development behaviors to generate development process data which includes writing test cases sequence, development cycle and time consumption of development cycle based on your commit information and time. At the same time, I will analysis your code quality associates with the commits to find out the code quality change. At the end of week 12, you will receive your finally gamification score and your code quality change during this period (week 9 to week 12). At the end of week 12, you will receive a questionnaire (see Appendix 2) which is anonymize about your feeling of applying gamification. These data will help us better understand gamification process and provide analysis that is more insightful for future research.

Your email address, GitHub ID, commit information, commit time, and commit contributor ID will be deleted at the end of week 12. The development process data and code quality data will be retained until the study finished and research paper published. This means all your personal data

will be deleted after week 12 and only population and anonymize data will be retained until the study finished.

The duration of participation from week nine to week twelve, approximately four weeks.

WHAT IS THE DATA USED FOR?

We expect to publish and for this reason, we are seeking separate thing from original your agreement of engaging the project, we are asking if you are willing to allow your data to be used in a research analysis and publication. The data will only be used for research purpose including research paper. If we are to include your work in the research, you must give free independent consent.

WILL TAKING PART BE CONFIDENTIAL?

All the data will be collected online to ensure the confidentiality and anonymity of the participants. Your GitHub id, email address and all personal data will be anonymous. All data will not be shared to others or the third party.

HOW WILL INFORMATION YOU PROVIDE BE RECORDED, STORED AND PROTECTED?

All development related data will be collected from git and the participants need to grant access authority to Wei. The data will be gathered and anonymized to store securely. All data will be anonymous and processing that we were stored only long as necessary to conduct the study.

> ‘Signed consent forms and original data will be retained *securely* and only Wei Ren can access these data. The data will be retained until the study finish. The study finish means the relevant research papers and dissertation published. Under freedom of

information legalisation, you are entitled to access the information you have provided at any time.

Appendix 1 Data Gathering Information

ID	Type of Data	Justification: Why do we need the data?	Data Format	Technical and Organisational Controls	Will it be stored	Identifiable coded, or anonymised	How long will the data be retained?
1	Consent Form	Evidence of consent from an ethical perspective	Word Form	Stored in encrypted equipment. Accessible to PI only.	Yes	Identifiable	Until the study finished
2	Commit	Analysis development process, which includes ID 3, ID 4 and ID 5.	Word Text	Stored in Participator's GitHub repository, PI can only access it.	No	Anonymised	Not retain
3	Commits Information	Analysis development process	Word Text	Stored in Participator's GitHub repository, PI can only access it.	No	Anonymised	Not retain
4	Commits time	Analysis time consuming of development process	Word Text	Stored in Participator's GitHub repository, PI can only access it.	No	Anonymised	Not retain
5	Commits contributor ID	Identify commits belong to	Word Text	Stored in Participator's GitHub repository, PI can only access it.	No	Anonymised	Not retain
6	Writing test cases sequence (0-test last, 1-test first)	Analysis development behavior, this data is generated based on ID 3.	Numerical Value	Stored in encrypted equipment. Accessible to PI only.	Yes	Anonymised	Until the study finished
7	Number of Development cycle	Analysis development behavior, this data is generated based on ID 3.	Numerical Value	Stored in encrypted equipment. Accessible to PI only.	Yes	Anonymised	Until the study finished
8	time consumption of development cycle	Analysis development behavior, this data is generated based on ID 3 and ID 4.	Numerical Value	Stored in encrypted equipment. Accessible to PI only.	Yes	Anonymised	Until the study finished
9	Source Code	Analysis code quality change	Code	Stored in Participator's GitHub repository, PI can only access it.	No	Anonymised	Not retain
10	Maintainability Index	Analysis code maintianability, this data is generated based on ID 9.	Numerical Value	Stored in encrypted equipment. Accessible to PI only.	Yes	Anonymised	Until the study finished
11	Cyclomatic Complexity	Analysis code complexity, this data is generated based on ID 9.	Numerical Value	Stored in encrypted equipment. Accessible to PI only.	Yes	Anonymised	Until the study finished

12	Halstead science metrics	Analysis code quality, this data is generated based on ID 9.	Numerical Value	Stored in encrypted equipment. Accessible to PI only.	Yes	Anonymised	Until the study finished
13	Test cases coverage	Analysis code quality, this data is generated based on ID 9.	Numerical Value	Stored in encrypted equipment. Accessible to PI only.	Yes	Anonymised	Until the study finished
14	Questionnaire	Collecting feedback of applying gamification	Numerical Value	Stored in encrypted equipment. Accessible to PI only.	Yes	Anonymised	Until the study finished

Commit: a snapshot of your repository at certain time.
Code quality: Software functional quality reflects how well it complies with or conforms to a given design, based on functional requirements or specifications

Appendix 2 Questionnaire

Questionnaire of Applying Gamification

Each question is optional. Feel free to omit a response to any question; however, the researcher would be grateful if all questions are responded to.

- *Do you feel your skills improving?*
  1_____ 2_____ 3_____ 4_____ 5_____ Don't know_____
- *Do you feel gamification strategy fair?*
  1_____ 2_____ 3_____ 4_____ 5_____ Don't know_____
- *Do you easily understand how point is computed?*
- *1_____ 2_____ 3_____ 4_____ 5_____ Don't know_____*
- *Does gamification have positive motivating effects in your group?*
- *1_____ 2_____ 3_____ 4_____ 5_____ Don't know_____*
- *Does gamification provide you additional effort?*
- *1_____ 2_____ 3_____ 4_____ 5_____ Don't know_____*
- *Do you engage in code testing?*
- *1_____ 2_____ 3_____ 4_____ 5_____ Don't know_____*

5 means most likely, and 1 means least likely.

Illicit activity: *In the extremely unlikely event that illicit activity is reported I will be obliged to report it to appropriate authorities.*

You have right to not submit or exit without submitting at any time.

c. Participate Consent Form

Participant Consent Form

[*Gamification on Software Engineering Process*]

Consent to take part in program

x I voluntarily agree to participate in this program.

x I understand that even if I agree to participate now, I can withdraw at any time or refuse to answer any question without any consequences of any kind.

x I have had the purpose and nature of the program explained to me in writing and I have had the opportunity to ask questions about the program.

x I understand that participation involves development process and codebase being analyzed.

x I understand that I will not benefit directly from participating in this program.

x I agree to my development process and codebase from GitHub being analyzed.

x I understand that all information I provide for this program will be treated confidentially and anonymous.

x I understand that if I inform the researcher that myself or someone else is at risk of harm, they may have to report this to the relevant authorities - they will discuss this with me first but may be required to report with or without my permission.

x I understand that under freedom of information legalisation I am entitled to access the information I have provided at any time while it is in storage as specified above.

x I understand that I am free to contact any of the people involved in the program to seek further clarification and information.

Wei Ren

PhD Candidate

renw@tcd.ie

Trinity College Dublin, School of Computer Science and Statistics

Signature of program participant

---------- ----------

Signature of researcher

I believe the participant is giving informed consent to participate in this program

---------- ----------

d. Data Consent Form

Data Consent Form

[*Gamification on Software Engineering Process*]

Consent to share Data

x I voluntarily agree to share the data in this research study.

x I understand that even if I agree to share data now, I can withdraw at any time or refuse to share my data without any consequences of any kind.

x I understand that I can withdraw permission to use data from my code repository (GitHub) within two weeks after study finish, in which case the material will be deleted.

x I understand that in any paper on the results of this research my identity will remain anonymous. This will be done by deleting my name, code repository ID and disguising any details of my data which may reveal my identity or the identity of people I speak about.

x I understand that analysis result extracts from my data may be quoted in dissertation, conference presentation, published papers etc.

x I understand that signed consent forms and data recordings will be retained in encrypted equipment securely until the study finish.

x I understand that under freedom of information legalisation I am entitled to access the data I have provided at any time while it is in storage as specified above.

x I understand that I am free to contact any of the people involved in the research to seek further clarification and information.

Wei Ren

PhD Candidate

renw@tcd.ie

Trinity College Dublin, School of Computer Science and Statistics

Signature of research participant

---------- ----------

Signature of researcher

I believe the participant is giving informed consent to participate in this study

---------- ----------

3. Gamification Document

a. Screen Shot of Gamification Structures

1. Finishing a cycle (a unit test and product code), participant will get 2 points.
2. Generating a failing unit test before function code, and participant will get 1 point when unit test pass after finishing function code. (Finishing a cycle with TDD style)
3. Writing one commit of test case, participant will get 0.5 points, if writing a commit of creating new test case, earn extra 2 point.
4. Participant will get 5 points if your repository has one star. Everyone is expected to star your peers (more than one peer).
5. Participants will be ranked and participant position in the leaderboard will be released with gamification point. The points will be accumulated.
   - The highest ranking in total will earn 15 points, but you can get extra bonus (10 points) if your coverage reaches 70%.
   - The second highest will get 14 points, etc.
   - The ranking will be refreshed every week, and the points will be accumulated.

The participants will be noticed that their current points and their position in the leaderboard. At the end of week 12, every five participants will be grouped in one leaderboard based on their point.

The participants in different leaderboard will get real-world reward including meal, drink and snack.

Leaderboard 1: 30, 25 and 20 euro (meal voucher).

Leaderboard 2: 15, 12 and 9 euro (drink voucher).

Leaderboard 3: 6, 4 and 2 euro (snack voucher).

The notification which will be released every Monday and Thursday by email, includes current point and participant position in the leaderboard.

The gamification points and the position of leaderboard will not have any ***IMPACT*** on the module ***GRADE***.

b. Screen Shot of Gamification Feedback

Hi there,
This is the first time that the gamification score is released!

Your gamification score is: **16** and you are **No.4** in the leaderboard.

A little tips: In this final week, you have done a great job! Maybe you can try to create some unit tests to test your function!

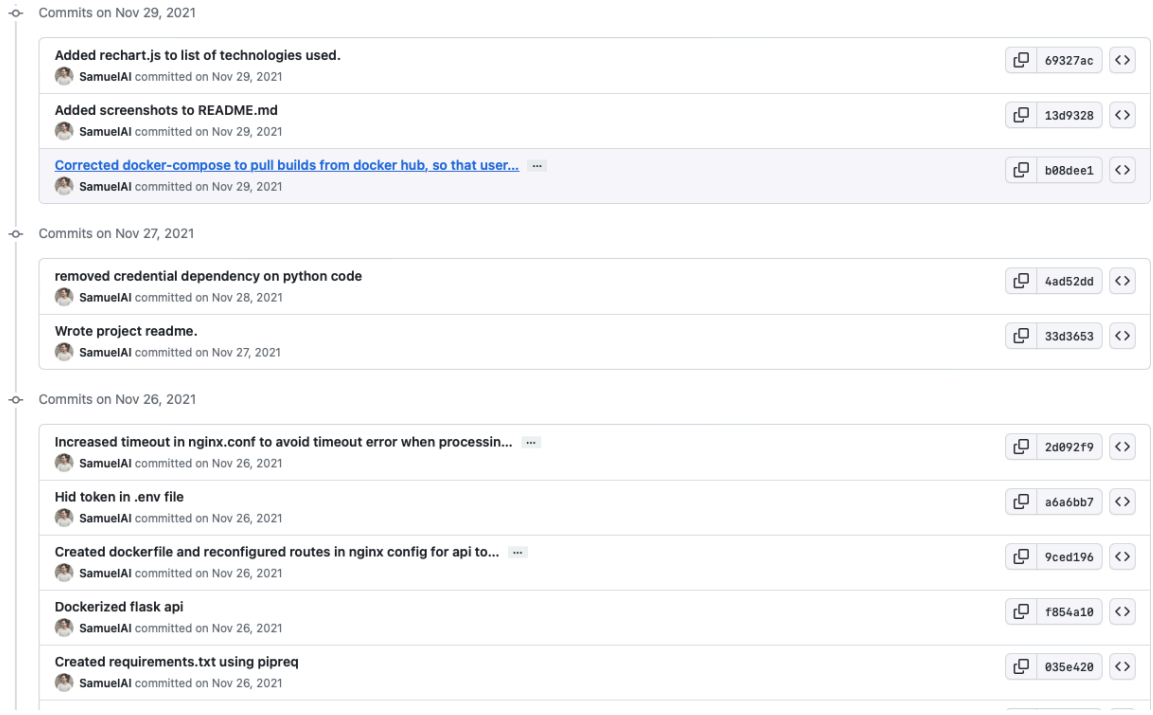
If you have any questions about development methods or gamification, please feel free to contact me without hesitation.

Thank you again for your participation and have a good day!

Regards,
Wei

Reply Forward

c. Screen Shot of Github



d. Screen shot of Data Sample

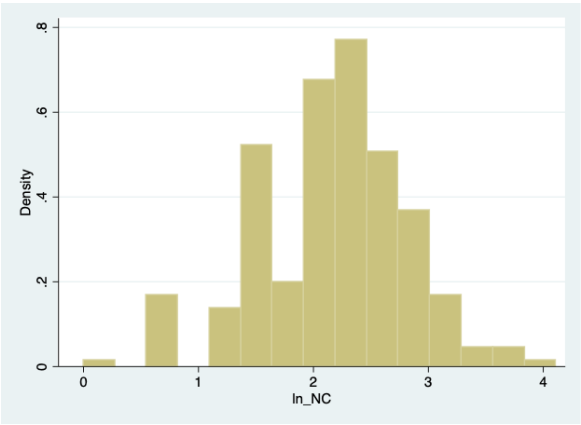
id	gender	language	use_rig_date	use_exp	Cycle	Test	NTC	NPC	NC	function	MI_m	MI	CC	comment_ra~o	loc	halstead_b~p	HD	halstead_e~t	halstead_t~d	halstead_v~e
1	1	haskell	01dec2020	2	17	5	0	13	13	2	54.968858	64.968858	4	19.993767	163	1.2025886	63	227289.25	12627.181	3607.7659
2	1	js	01dec2020	2	5	0	0	17	17	3	70.032717	60.032717	9	3.5627455	732	10.27108	34.119205	1051323.2	58406.846	30813.239
3	1	python	01oct2021	1	19	3	11	36	47	6	34.460925	64.460925	7	25.806183	410	3.0708826	102.98077	948725.55	52706.975	9212.6477
5	1	python	01oct2021	1	22	0	0	23	23	3	70.631639	80.631639	7	20.549828	67	.60644077	24.956522	45403.957	2522.442	1819.3223
6	1	python	01oct2021	1	1	0	0	2	2	0	55.103007	75.103007	16	42.689942	144	1.0141192	31.641975	96266.208	5348.1227	3042.3577
7	0	python	01oct2021	1	6	0	0	15	15	2	59.906117	69.906117	8	29.871352	128	.82789495	23.981481	59562.442	3309.0246	2483.6848
8	0	python	01dec2020	2	0	0	0	10	10	0	71.539122	81.539122	4	44.629057	74	.5816188	18.686441	32605.156	1811.3975	1744.8564
9	0	python	01dec2020	2	4	3	3	51	54	4	38.498781	48.498781	6	48.68937	296	2.0076759	42.095238	253540.79	14085.599	6023.0278
10	0	js	01dec2020	2	6	0	0	25	25	5	28.787071	38.787071	7	2.9183425	368	3.8789044	45.789474	532838.98	29602.165	11636.713
11	1	python	01dec2020	2	3	0	0	6	6	1	44.57812	54.57812	11	31.958164	206	2.016533	35.678808	215842.48	11991.249	6049.599
12	0	js	12jul1905	2	4	0	0	0	25	4	28.787071	18.787071	14	2.9183425	368	3.8789044	45.789474	532838.98	29602.165	11636.713
13	1	js	01oct2021	1	10	0	0	20	20	5	42.429392	32.429392	10	9.070532	242	3.0022634	42.856	385995	21444.166	9006.7901
14	1	js	01oct2021	1	6	0	0	26	26	1	52.638521	42.638521	8	1.7486549	162	1.6078	44.37931	214059.17	11892.176	4823.4
15	1	js	01oct2021	1	1	0	0	3	3	0	63.95141	53.95141	1	9.991765	91	1.5681778	18.092105	85114.915	4728.6064	4704.5335
17	0	python	01oct2020	1	21	0	0	6	6	2	69.994987	59.994987	5	48.516949	91	.39318865	19.444444	22936.004	1274.2225	1179.5659
18	1	python	01jan2021	2	0	0	9	44	53	6	38.074577	28.074577	9	1.2833169	283	4.4525432	117.22353	1565828.5	86990.472	13357.63
20	1	python	10jul1905	4	29	9	0	0	53	6	38.074577	28.074577	9	1.2833169	283	4.4525432	117.22353	1565828.5	86990.472	1357.6297
21	1	js	01dec2020	2	0	0	0	5	5	1	52.900727	42.900727	1	53.006993	165	2.0567439	24.208	149368.97	8298.2761	6170.2318
22	1	python	01jan2019	4	19	2	2	18	20	5	31.649771	21.649771	9	7.0094747	345	5.309817	45.225	720409.42	40022.746	15929.451
24	1	python	01dec2019	3	3	0	0	8	8	1	32.811792	22.811792	9	9.7747918	299	4.2614018	56.636943	724058.31	40225.462	12784.205
25	1	js	12jul1905	2	3	2	3	6	11	4	100	90.656671	3	0	20	.13248605	9.8421053	3911.8248	217.3236	397.45814
27	1	js	01dec2020	2	7	0	0	19	19	2	47.352967	47.352967	8	7.6421739	833	15.561849	62.664016	2925503.9	162528	46685.548
28	1	js	01oct2021	1	7	0	0	13	13	1	42.503009	32.503009	9	0	222	2.7105463	43.713043	356507.8	19805.989	8155.6389
29	1	python	01sep2020	2	8	0	1	10	11	2	46.733297	36.733297	9	8.6963485	180	2.2160134	26	172849.04	9602.7247	6648.0401
30	1	python	01dec2020	2	1	0	0	24	24	1	67.712128	57.712128	10	12.250333	82	.67626423	19.090909	38731.497	2151.7498	2028.7927
32	1	python	01oct2021	1	0	0	0	1	1	0	110	100	12	0	6	.0184503	2.5714286	142.3309	7.9072723	55.350906
33	1	ts	01dec2016	7	12	0	0	53	53	2	63.76846	53.76846	1	12.411010	109	.92571063	15.230095	42318.2	2351.0111	2777.1319
34	1	php	01mar2018	5	42	1	0	64	64	7	12.577282	2.5772816	1	3.8655256	984	18.403695	150.96154	8334750.3	463041.68	5211.0846
35	1	js	01dec2020	2	12	0	1	14	15	10	67.165628	57.165628	7	1.2592801	513	9.0822208	80.843096	2202704.5	122372.47	27246.662
36	1	go	01oct2021	1	37	23	27	51	78	7	25.704965	15.704965	4	8.0545717	1282	22.527216	57.072172	3857031.4	214279.52	6581.6485
37	1	python	01dec2016	6	2	0	0	4	4	1	59.495133	49.495133	10	32.702626	128	.82012779	24	59049.201	3280.5112	2460.3834
38	1	python	01dec2020	2	14	0	0	28	28	2	52.182427	42.182427	7	20.356037	171	1.5501749	30.160194	140260.73	7792.2628	4650.5248
39	1	js	01dec2020	2	28	0	0	55	55	5	30.28487	20.28487	7	3.639289	457	4.8888224	39.509901	579470.67	32192.815	14666.467
40	1	python	01oct2021	1	13	0	0	31	31	2	69.344209	59.344209	6	36.226982	80	.63687422	19.658537	37560.045	2086.6692	1910.6227
41	1	js	01dec2014	8	4	0	0	12	12	2	27.420827	17.420827	19	6.9892473	441	5.5733001	38.994604	651985.89	36221.438	16719.9
42	1	python	01dec2019	3	8	0	0	22	22	3	29.330014	19.330014	13	1.6592268	347	5.2349052	127.40964	2000932.2	111162.9	15704.716
43	1	python	01nov2020	2	6	1	2	61	63	4	56.235644	46.235644	8	30.386052	139	1.0867856	26.088608	85058.173	4725.4541	3260.3569
44	0	js	01oct2021	1	3	0	0	21	21	1	52.76125	42.76125	4	5.2899675	19058	539.37803	316.82965	5.127e+08	28481825	1618134.1
45	1	python	01oct2021	1	7	0	0	21	21	1	81.803831	71.803831	6	17.054846	46	.32521905	15.052632	14686.208	815.90043	975.65716
46	0	python	01dec2020	2	6	0	0	34	34	6	40.89927	30.89927	14	33.265223	226	1.9696884	46.188034	272928.1	15162.672	5909.0652
47	1	ts	01dec2013	8	7	3	1	21	22	2	61.354233	51.354233	2	20.200657	117	1.0010002	20.250345	96461.20	5381.1838	3243.2675

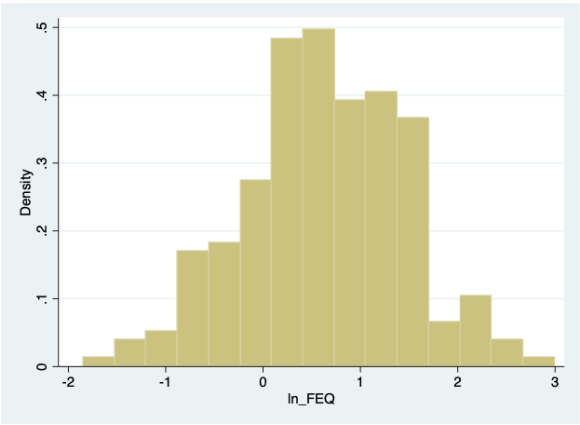
4. Plot Distribution of Variables

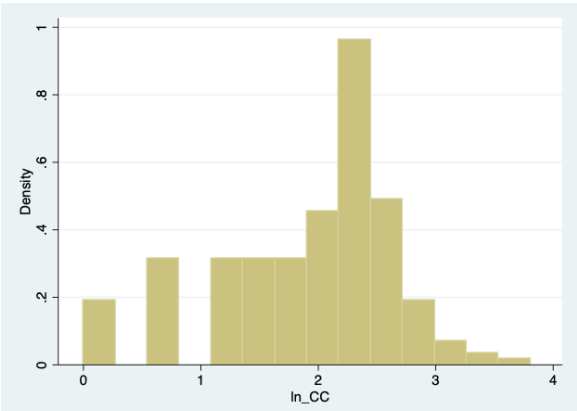
Here are the main variables used in regression.

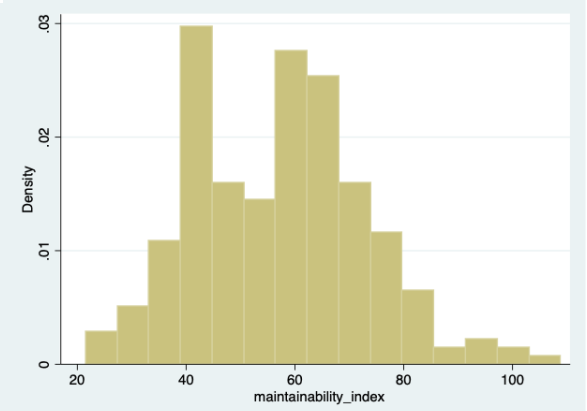
a. Observational Study

Number of Commits, Frequency of update code, Complexity and maintainability index.



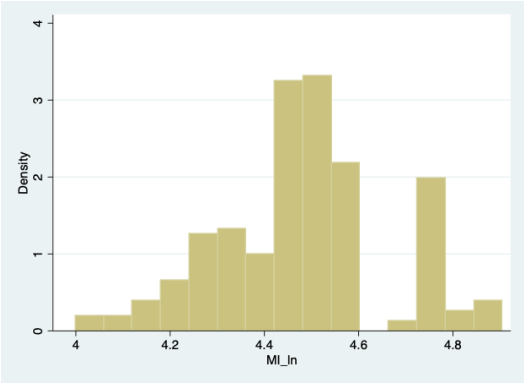






b. Group Experiment

Maintainability index



c. Individual Experiment

Maintainability index and complexity.