Gamification in Software Testing: A Characterization Study*

Gabriela Martins de Jesus
DC/UFSCar - Brazil
gabriela.jesus@ufscar.br

Fabiano Cutigi Ferrari
DC/UFSCar - Brazil
fcferrari@ufscar.br

Daniel de Paula Porto
DC/UFSCar - Brazil
daniel.porto@ufscar.br

Sandra Camargo Pinto Ferraz Fabbri
DC/UFSCar - Brazil
sfabbri@ufscar.br

ABSTRACT

Context: Testing is a fundamental activity in the software development cycle. Revealing software faults is its main objective. Despite that, testing is considered unpleasant, dull and tedious. As a result, there is a lack of expertise among professionals while many projects fail. Gamification is a promising way to address testing issues; it is a new trend being used mostly to increase motivation, engagement and performance with the use of game elements in non-game contexts. *Objective:* To describe results of a study that aimed to characterize how gamification has been explored to support software testing. *Method:* The studies that compose our baseline for analysis and discussion were obtained through a systematic mapping carried out following a research protocol. To retrieve relevant literature, we applied automatic search and backward snowballing. At the end, we selected 15 studies that we analyzed and classified according to six perspectives: application context, used gamification elements, gamification goals, testing techniques, testing levels, and testing process phases. *Results:* The most used gamification elements are points, leader boards, and levels, and unit testing and functional testing are the level and technique most addressed in the studies, respectively. *Conclusion:* Gamification is a rising research topic, especially in the software testing field. The increasing interest for gamification has the potential do lead to positive outcomes. The map presented in this paper can be a useful resource for the identification of gaps and for triggering new research initiatives.

CCS CONCEPTS

• **Software and its engineering → Software verification and validation**; **Software testing and debugging**;

KEYWORDS

Gamification; game elements; software testing; automated testing; systematic mapping; characterization study.

ACM Reference Format:
Gabriela Martins de Jesus, Fabiano Cutigi Ferrari, Daniel de Paula Porto, and Sandra Camargo Pinto Ferraz Fabbri. 2018. Gamification in Software Testing: A Characterization Study. In *III Brazilian Symposium on Systematic and Automated Software Testing (SAST '18), September 17–21, 2018, SAO CARLOS, Brazil.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3266003.3266007

1 INTRODUCTION

Developing reliable software still remains a challenge. To increase reliability, software testing plays a key role. In general, testing consists in executing the software in any executable abstraction model, with the use of specific input data, with the intent of revealing faults [18]. It then checks whether the observed behaviour is in agreement with the expected behaviour. Despite this, testing is a very time- and effort-consuming factor in the software development process. Testing may consume up to 40% of project budgets, as annually reported by a worldwide survey with IT managers [4]. As such, testing is considered a tedious and low-motivating task. As a consequence, many projects produce artifacts with poor quality, or overruns the time or cost [8]. Among other reasons, this occurs due to human factors present throughout the development of software and, more specifically, during the software testing process. Due to this, motivation and discipline have become crucial elements for good software development [8]. One way to introduce and maintain these two key elements is by using *gamification*.

Gamification is a subject widely discussed in areas other than Software Engineering [12]. It uses game elements and mechanics in non-game contexts to stimulate behavior, improve people's motivation and engagement in their tasks [11]. It can bring joy aspects to software development. Thus, unpleasant tasks for the development team, such as writing unit test and easy-to-do maintenance, are stimulated with rewards and joy obtained with gamification.

In this context, this paper characterizes the use of gamification as a technique to support (and enhance the quality of) software testing. It reports on a systematic mapping study that analyzes and discusses gamification applied to software testing based on six perspectives: application context for gamification; used gamification elements; gamification goals; software testing techniques; software testing levels; and software testing process phases. We retrieved 817 references, from which we selected 15 primary studies to compose our final set (details in Section 3). Our contributions include an overview of gamification initiatives to support software testing (Section 4), as well as a classification of studies based on the six perspectives (Section 5). We also point out limitations of our study

*Produces the permission block, and copyright information and summarize related research (Section 6), and present our conclusions and future work in Section 7. Notice that, to the best of our knowledge, this is the first study that focuses on a broad characterization of the state-of-the-art of gamification in the software testing context. Next, we provide some key concepts of gamification.

2 BACKGROUND

Consider you have a system where your employees must log their work activities following a prespecified set of rules[1]. Every single day. Besides this, imagine that your Human Resources Department faces huge challenges because the employees do not do that, at least not every day, or not as expected to. So, you decided to embed into you system a list of features aiming to motivate the employees to do what you want they do. You used a point system to reward them when they log their activities every day, and give extra points when they do it correctly. If they log for five consecutive days, so they earn a badge, which is visible in their profile for anyone to see; if they do it again for 3 times, they earn another badge, and so on. Besides this, a leader board is placed visible in your company ranking the employees by their earned points. The first three leading the ranking for more time, in rounds of two months, are recognized and earn real prizes. Well, what you did in this example was gamifying your system. You added game elements (points, badge, leader board) in a context that is not a game.

To understand the term *non-game contexts*, consider the current example. The company is a "non-game context" because the employees' purposes are not only fun. A classroom is another example, though from a different context: educational. Even though we can add entertainment or fun (as we usually do it when using gamification), it will be to support the teaching and learning processes, not to become a game.

Game elements can be defined as small pieces, like Lego, that compose the games [28]. Using several Legos, we can build a car, a house, or many things our imagination allow us. Similarly, bringing the smaller pieces together, we develop a game, a serious game, or a gamified system, depending on our intention. We go further in game elements in Section 3. For now, we anticipate that gamification includes three classes of elements: game dynamics, game mechanics, and game components. That said, in this paper we use the term *game element* as synonym of *game component*.

2.1 Gamification in Software Testing

Gamification in software development, in general, is not the focus of our research, but we address one of its development process phases: software testing. In our search, four selected studies [7, 8, 11, 21] did not present approaches to gamify, specifically, software testing activities. In fact, they were selected for one reason: they encompass the software testing activity within the software development.

In software development, testing is performed with the aim of revealing faults [18]. However, it is not a simple or easy activity; not even intrinsically motivational for everyone, especially for developers. Sometimes, testing a new software release may take several rounds of back and forth between developers and quality assurance teams, until getting tedious and boring. This is the time to *gamify*, and the state-of-the-art is discussed in the next sections.

[1]This scenario was adapted from the work of Werbach and Hunter [28]

3 METHOD AND DATABASE

To achieve our goal, we analyzed the literature of gamification applied to software testing. We performed a systematic mapping (SM) [23], which is a type of secondary study that provides an overview of the investigated area and indicates research opportunities. It is guided by a predefined and evolvable research protocol [9]. Key elements of the protocol are next presented.

Research Question: Our research question is: *How has gamification been investigated to support software testing?* As shown in Sections 4 and 5, we draw the answer to this question based on six perspectives, which are addressed by the following sub-questions (all regarding gamification as a way to support software testing):

(1) *In which context has gamification been applied?*
(2) *Which gamification elements have been used?*
(3) *Which goals have been pursued?*
(4) *Which testing techniques have been addressed?*
(5) *Which testing process phases have been covered?*
(6) *Which testing levels have been addressed?*

Search string and surveyed database: The base search string was "`(gamif*) AND (test*)`". The idea was to define a generic search string that would fetch a broad range of results[2], so that we could carefully analyze in order to reach our final set of studies. The string was applied in April, 2018, to the Scopus search engine[3]. In total, 540 studies were retrieved. The study selection steps (next described) resulted in the following numbers: Pre-selection: 47 studies; Final selection: 9 studies [1, 7, 11, 14, 15, 19, 24, 26, 29]; and Backward snowballing: 6 studies [3, 6, 8, 21, 25, 27]. The final set is composed of 15 studies [1, 3, 6–8, 11, 14, 15, 19, 21, 24–27, 29]. Notice that the snowballing step required the analysis of additional 277 – *i.e.* non-duplicated – references; it was performed in a single round. Also notice that the final set does not include 4 studies [2, 5, 17, 20] that were subsumed (*i.e.* updated or extended) by more recent studies. The subsuming and subsumed studies are:

- Anderson et al. [1] subsumes Anderson et al. [2]
- Yujian Fu and Clarke [29] subsumes Clarke et al. [5]
- Dal Sasso et al. [7] subsumes Mastrodicasa [17]
- Parizi [19] subsumes Parizi et al. [20]

Study selection criteria and procedures: We defined the inclusion **(i)** and exclusion criteria **(e)** listed next. We selected studies that passed **i1** or **i2**, and did not pass any of the exclusion criteria.

i1. Proposes or applies a technology (approach, tool, framework, method etc.) to gamify software testing education.
i2. Proposes or applies a technology (approach, tool, framework, method etc.) to gamify the practice of software testing.
e1. Does not address gamification of software testing.
e2. Is not written in English.
e3. Is a secondary study (Section 6 discusses these studies).

Data extraction procedures: Data extraction basically consisted of elaborating a summary of each selected study, with special attention to the support provided by gamification to the testing activity (either for education or practice). While elaborating the summary, we applied the six classifications associated with each sub-question.

[2]Notice that the term "`test*`" led to the matching of studies from areas other than Computing (*e.g.* studies from medical sciences that addressed a gamified software which has somehow been "tested").
[3]http://www.scopus.com - last accessed on 19-June-2018.

Table 1: Classifications and categories applied to selected studies.

Category	Description
APPLICATION CONTEXT	Contexts in which gamified approaches for software testing may be applied.
Educational	Aims to use gamification to teach software testing in either academic or industrial context.
Industrial	Aims to use gamification in software testing in an industrial context for business purpose.
Any	
USED GAME ELEMENTS	Concrete pieces of games embedded into a gamified environment.
Achievement	Aims to define objectives to be reached. May be used to stimulate harder work.
Avatar	Aims to provide a visual representation of characters. May be used to stimulate engagement and motivation.
Badge	Aims to provide a visual representation of achievements. May be used to stimulate *e.g.* engagement and motivation.
Duel	Aims to provide battles. May be used to stimulate competition and engagement.
Leader Board	Aims to provide a public visual display of performance. May be used to enhance competition, stimulate motivation.
Level	Aims to define steps to reach. May be used to stimulate engagement, competition, and provide a sense of progress.
Points	Aims to quantify progress. May be used *e.g.* to encourage harder work, and to provide personal feedback.
Quest	Aims to define challenges to reach a goal. May be used to drive actions, stimulate engagement.
Social Graph	Aims to represent a social network into the game. May be used to stimulate collaboration, engagement, motivation.
Team	Aims to define groups to work together. May be used to encourage collaboration, motivation, competition.
Virtual good	Aims to provide assets with virtual- or real-money value. May be used to stimulate *e.g.* engagement and motivation.
GAMIFICATION GOALS	Expected results from the use of gamification.
Increase awareness	Aims to increase the people' awareness regarding their performance and results.
Boost adoption	Aims to boost the adoption of software testing.
Develop creativity	Aims to motivate the development of creativity to perform tasks.
Ease the fixing process	Aims to motivate people to perform their tasks minimizing the effort in the fixing process.
Encourage testing habits	Aims to encourage developers to perform testing until it become habit.
Increase engagement	Aims to engage people in testing activities.
Improve skills	Aims to stimulate the improvement of student's knowledge, efficiency, performance, among other skills.
Increase enjoyment	Aims to increase enjoyment while learning/performing software testing.
Increase motivation	Aims to increase motivation to learn/perform software testing.
Enhance monitoring	Aims to enhance the monitoring of all people involved in the development of complex software artifacts.
Increase persuasion	Aims to persuade people to have expected behaviours.
Stimulate collaboration	Aims to stimulate collaboration among people involved in an activity.
Improve training	Aims to improve training to perform software testing or other activities related to software development.
TESTING TECHNIQUES	Approaches that rely on varying underlying software artifacts to derive the test requirements.
Functional Testing	Relies on software specification documents to derive the test requirements (also known as *black-box* testing).
Structural Testing	Relies on implementation details of the software to derive test requirements (also knows as *white-box* testing).
Fault-based Testing	Relies on recurring, documented software faults (*i.e.* fault models and/or taxonomies) to derive test requirements.
Any	
TESTING PROCESS PHASES	Phases included in typical, comprehensive testing process models.
Planning	Comprises the definition of how testing will be performed and what will be tested.
Data / environ. configuration	Addresses the prioritization and implementation of test environment requirements established in the test plan.
Test cases design	Definition of test classes and conditions, thus requiring access to planning and configuration artifacts.
Execution and evaluation	Execution of tests and eventual reporting of noticed failures, as well as assuring test goals were achieved.
Monitoring and control	Aims to organize, consolidate and provide rapid access to information produced during the process execution.
Maintenance	Aims to maintain the test suites (specially the automated ones) during the software evolution.
Any	
TESTING LEVELS	Take into account the granularity of the portion of the software which is under testing.
Unit Testing	Aims to test each software unit in isolation, with the intent of revealing faults related to the implemented logic.
Integration Testing	Aims to identify problems related to the interface between combined units (*i.e.* integrated units) of a given software.
System Testing	Aims to assess the software functionalities and performance in general, when executed in its final infrastructure.
Any	

The full set of categories is shown in Table 1. It is important to highlight that the lists of categories which are specific to gamification (namely, *used gamification elements* and *gamification goals*) evolved during the analysis and data extraction steps. In other words, we did not have comprehensive lists of categories beforehand. Moreover, we do not claim these lists are complete and definite, since more studies can be added to our final set in the future.

We also highlight that the inclusion of the option *any* in some classifications (namely, *application context*, *testing techniques*, *testing levels*, and *testing process phases*) means that a given study could have been classified with any of the specific categories within that classification. Some notes regarding the classifications come next.

Used gamification element: According to Werbach and Hunter [28], the game elements are small pieces we can use to build a game; or a toolkit. They are divided into three types: *dynamics*, *mechanics*, and *components*. *Dynamics* and *mechanics* are more abstract concepts, not directly instantiable into the games (or in the gamified systems, in our case), and not always and easily noticed by the "players". For example, we can notice that points and levels (game components) give a sense of progression (game dynamics); however, points and levels are visible for whom they are specific to, but progression is not (it is abstract). Thus, we decided to choose the small pieces of the games the "players" can really see – *i.e.* the *components* – and to which we refer to as *elements* along this paper.

Gamification goal: We highlight that educational/business objectives, and gamification goals might not be the same. For example, the CODE DEFENDERS game ([6, 24–26] has the educational objective of teaching mutation testing. On the other hand, its gamification goals are, mainly, to increase enjoyment throughout the learning process, to increase students' engagement and motivation, and to improve students' skills such as their knowledge, performance in testing activities, and creation of stronger tests and mutants.

4 OVERVIEW OF SELECTED STUDIES

This section presents an overview of how gamification was proposed and used by the authors of selected studies.

García et al. [11] proposed the GOAL (Gamification on Application Lifecycle Management) framework with the aim of guiding a process that supports the introduction of gamification in any software development phase. The authors performed a case study in a real software development company to investigate the feasibility of applying GOAL to integrate gamification in other software engineering environment. Three process areas – requirements management, project management, and software testing – were gamified following a methodology present in the framework. The game elements chosen to be used were: points, levels, badges, leader board, social graph, and challenge. Gamification goals were not defined in the case study, since the object of study was the GOAL framework itself, but the methodology aims to gamify systems to engage users and improve their performance. Unit testing was the only testing level mentioned in the case study. Despite this, we understand that any testing technique, level and process phase can me gamified by using GOAL. The outcomes were positive, revealing benefits provided by the framework evaluated in a industrial context.

CODE DEFENDERS[4] is a gamified system used to teach software testing based on mutation testing principles; besides this, it can also be used as a crowdsourcing approach, or an evaluation framework for educators. The game was presented in four studies **[6, 24–26]**. The common gamification goals in the four studies were: increasing engagement, motivation, and enjoyment to learn mutation testing. Other goals were mentioned in one or more of those studies: producing stronger tests [26], improving student's skills [6, 25], improving student's understanding [25], fostering the adoption among developers and practitioners [24], and improving tester's performance [24]. The used game elements in the four studies were: *points*, *duels*, and *levels*. *Leader board* was used only in one study [25], and *team* was used in two studies [25, 26]. Next, we are going to present varying characteristics of these four studies.

Rojas and Fraser [24] presented an early prototype of the game with the aim of exploring ways to make mutation testing fun to learn and apply. The authors' hypotheses were that the players will create stronger tests and mutants, and will perform better at testing tasks; no results for the hypotheses were presented because an empirical evaluation was intended to be performed in future versions of the game. In another study, **Rojas and Fraser [25]** explored the "use of gamification to teach mutation testing concepts". For this, they described manners to provide practical experience (*e.g.* puzzles, duels, self-tutoring, rewards) in an educational context. The research hypothesis presented was: "*through the use of a mutation testing game, students will be able to grasp all relevant mutation testing concepts while having fun, and in the end become better software developers and testers, who produce higher quality software*". The authors presented some educational material to provide practical experience. They also mentioned that the game may serve as a evaluation tool for educators to extract feedback related to student. No evaluation was performed.

More recently, **Rojas et al. [26]** presented an approach based on gamification and crowdsourcing to produce and get stronger tests and mutants from players who use the game, which includes the possibility of industrial application and outsource some of the developer's work. This game version comprehends a single-player mode, uses the automated tools EvoSuite[5] to automatically generate tests, Randoop[6] to answer one research question, and Major[7], a tool to create mutants automatically. The authors also investigated the feasibility of gamifying software testing using CODE DEFENDERS, and evaluated the game as a crowdsourcing approach. For the first investigation, the authors evaluated if testers produce better tests using the game, and if they prefer writing tests while playing a game. For the second, they evaluated if crowdsourcing leads to stronger test suites and mutants than automated test generation, and if mutation scores on crowdsourced mutants correlate with scores on traditional mutants. The results were positive for all questions; besides this, the authors noticed the need of incorporating new game elements to improve player engagement and enjoyment.

Finally, **Clegg et al. [6]** proposed an approach to systematically teach testing concepts, and presented some testing concepts to demonstrate how they can be used in CODE DEFENDERS. In this study, the single-player mode was introduced in the game; duels still exist, but automated players replaced the second player present in multiplayer mode; and puzzles were added to the game. Two automated tools were mentioned (as in [26]): EvoSuite and Major. EvoSuite plays the defender role generating test cases automatically; on the other hand, Major plays the attacker role, depending on which role the human player chooses. No evaluation was performed.

Anderson et al. [1] presented a cloud-based learning environment called Learn2Mine[8]. Its goals are providing an environment where instructors can teach programming in data science courses, and providing students (either in academe or industry) with an enjoyable and useful system that would scaffold them through the learning process. Lessons are included in the system as evolving subproblems; the students should solve them until, incrementally, reach the lesson's goal. Although software testing is not the focus in the course, the subproblems comprehend unit testing concepts. Thus, students end up learning and applying unit testing in their code to solve the main problem and earning a reward concluding the lesson. Learn2Mine gives immediate feedback grading automatically the students' submissions, and also allows them to re-submit the problems many times they want to. No details about testing techniques, levels (except unit testing) or process phases were given, but we understand that any of those may be taught using Learn2Mine, since more lessons can be added to it. The environment is gamified using points, badges, leader board, and duels as its game elements to increase enjoyment, engagement, and motivation. Students not interested in gameplay experience can simply ignore it because the elements were designed to be unobtrusive. A survey was applied to evaluate the pedagogical usability and usefulness of Learn2Mine's characteristics; all questions received positive response, and the retry-system received the highest positive result. Other experiments were performed to compare results from two groups of students: one who had access to the gamified scaffolding learning, and other

[4]http://code-defenders.org/ - last accessed on 19-June-2018.

[5]http://www.evosuite.org/ - last accessed on 19-June-2018

[6]https://randoop.github.io/randoop/ - last accessed on 19-June-2018

[7]http://mutation-testing.org/ - last accessed on 19-June-2018

[8]http://learn2mine.appspot.com/ and https://github.com/Anderson-Lab/Learn2Mine-Main – accessed on 17-June-2018

who did not (but both had access to the retry-system, automatic grading, and immediate feedback). The results are that the performance of the first group was significantly better in overall task completion, supporting the authors' hypotheses that scaffolding and gamification increase completion rate. Some students provided feedback about their use of the system, but none of them who used the gamified version had only negative opinions. Therefore, Learn2Mine was concluded as a beneficial environment to teach data science courses, which is clearly extensible to software testing.

Yujian Fu and Clarke [29] proposed, assessed, and analyzed the effectiveness of WReSTT-CyLE[9] (Web-Based Repository of Software Testing Tutorials - a Cyberlearning Environment). It is a cyber enabled learning environment that uses gamification as a strategy to engage and motivate students to learn software testing methodologies and tools, and to improve their knowledge. The three key used game elements were points, badges, and leader boards, but the presence of levels and possibility of playing in teams were also identified. Regarding software testing, the authors mention that WReSTT-CyLE contains tutorials of tools based on cross-platform IDE (Integrated Development Environment), and content to teach functional and structural techniques. Although no other information was given about testing level and process phase, we understand that all of those are applicable to be taught using WReSTT-CyLE. Yujian Fu and Clarke evaluated if there is a relationship between student's virtual points and real grades (*a*), earned points and number of posts (*b*), and virtual points and course access frequency (*c*). The results revealed that there is a relationship in *a*, but two biases were identified. In *b*, little or no correlation was found. Finally, *c* was positive, but also with two biases. Therefore, the authors concluded that WReSTT-CyLE is an efficient learning environment, and that there is a relationship between gamification and an online environment to engage and motivate students to learn software testing. An interesting observation the authors mention is that the most efficient students were the most motivated ones.

Dal Sasso et al. [7] made several contributions. First, they performed a review of the state of the art to investigate how gamification elements can be used in software engineering. Them, they presented some definitions related to games, gamification, behaviourism, and cited an example of a failure of gamification by discussing if gamification is a lost cause. Second, the authors discussed principles, promises and perils of gamification. Third, the authors proposed a framework to systematically gamify any software engineering activity, including software testing (the reason that led us do include their study). After that, they presented a set of building blocks as the frameworks' foundation. Finally, [7] demonstrated how to use the framework. For this, they gamified two systems and proposed five methods to assess the performance of the gamification. The framework proposed by Dal Sasso et al. supports the design of a gamification layer integrated to existing workplace tools, both in educational and industrial contexts. It comprehends four sections (i.e. Activity, Analysis, Implementation, Testing) that support the design process. The evaluation performed by the authors involves the Testing section. The evaluation aimed to check how successful the game elements are when applied in each activity. Concrete examples were provided to demonstrate how to gamify two systems: The Myth and De-Bug, and The Empire of Gemstones. The first one is a bug tracking system in which the gamification goals were improvement of efficiency, development of creativity, improvement of the quality of bug reports, stimulus for the participation of the community, and easing the fixing process. The used game elements were avatar, badges, leader board, points, and virtual goods. The second system is in the software engineering context of modern code review. The main difference regarding to game elements is the introduction of levels to stimulate competition among users. The gamification goal in this example is to improve the quality of the code. Even though the authors have proposed assessment methods, no evaluation was performed by them.

Parizi [19] defended that approaches of traceability between tests and code artifacts need to be created or improved. So, they proposed to change the "existing post-mortem recovery of trace links to proactive construction of traceable software systems with highly engaged human factors". GamiTracify, a gamified conceptual framework, was created to engage and motivate developers in traceability tasks, resulting on the improvement of quality tracing links. It can be integrated to other tools, and applied in an industrial context. The idea is that test developers must incorporate in the code, or maintain, traceability information while developing the tests, thus avoiding post-verification of recovered links in later phases. The authors performed an empirical assessment to verify how GamiTracify impact the ability to have accurate traceability information compared to SCOTCH, a peer approach. They consider the metrics recall (the number of correct links) and precision (percentage of false positives) metrics to quantify the accuracy. The results revealed that recall and precision metrics was higher using GamiTracify than using SCOTCH. This suggests that gamification can have positive impact on team's motivation to reach accuracy. The used game elements to engage and motivate test developers were points, leader board, avatar, and quests. Structural testing technique and unit testing level were addressed in this study. Besides this, GamiTracify comprehends the development and maintenance phases, in which traceability information can be incorporated or extracted, respectively. This is the only selected study that clearly defined which process phases were gamified.

Liechti et al. [15] discussed and presented an industrial case study about test analytics, which comprehend continuous improvement, feedback mechanisms, and automated testing. The authors defined test analytics as "analytics on test-related data in order to give actionable insights about product quality and agile practices, with the goal to support a continuous improvement process". The team involved in the case study set up its own goal: getting better and consistent at testing, which would bring other benefits. The authors used the Fogg Behaviour Model (FBM) to trigger two behaviour changes: the adoption of Behaviour-Driven Development (BDD), and the rigorous maintenance of an automated test suite. Liechti et al. also developed an open source test analytics platform[10] to make the lessons learned from the projects available to other teams. This platform works integrated to other tools by capturing their produced test results. Based on these results, the team members are rewarded. The used game elements were the PBL triad (points, badges, leader board). Gamification was introduced

[9]http://wrestt.cis.fiu.edu/ – Acessed on 12, April 2018

[10]http://probedock.io/ – Accessed on 13, April, 2018

in the platform as an extrinsic motivator to increase enjoyment to perform the tasks. After the initial enjoyment and motivation caused by the extrinsic motivators (game elements), the developers became intrinsically motivated by having regular and visible feedback where they could see their progress and test results. Regarding software testing, functional and structural testing technique were mentioned; besides this, any testing level (unit, integration, and system) can be used. The process phases gamified are test cases design, its execution and evaluation, monitoring and control, and maintenance. An interesting result observed by the authors is that a developer novice in automated testing became "test addicted". The impacts caused by the experiments in the teams were considered significant and lasting. Although the case study has taken place in a startup company, we believe it is possible to extend both FBM and the platform to an educational context as well.

Laurent et al. [14] proposed a gamified system as a crowdsourcing platform for labeling equivalent mutants and evaluating several parameters involved in the detection process. The prototype is intended to collect information related to the detection of equivalent mutants, and related to the influence of players and mutant attributes in that process. Thus, only mutants that match the player's skill level will be presented to him. The gamification is used to keep the users engaged, and the game elements applied are points and a leader board. Moreover, the gamified prototype is focused in software testing activity, more specifically the fault-based testing technique, at the unit level. The testing process phases in which the users perform their tasks are test cases design, and execution and evaluation. The authors intended to use the system as a crowdsourcing platform, and they mentioned that it can also be used for developers or testers as a non-integrated tool. We understand that the presented system is not integrated to another one; also, it may be used for both educational (academy or industry) and industrial purposes. No evaluation results were presented.

The study of **Sheth et al. [27]** is a poster that presents the environment HALO, a prototype they developed to evaluate its benefits to students and feasibility. As the environment is discussed both by Sheth et al. [27] and Bell et al. [3], we decided to present its details when we summarized Bell et al.' work. In the following, we are limited to summarize only the results from a pilot study conducted by Sheth et al. [27]. The pilot study was conducted with undergraduate students. The students were offered to use the HALO prototype as optional in programming assignments. The results revealed that the students who used HALO showed greater improvement than the others who did not. Besides this, some of them felt themselves more secure and comfortable to submit their assignment after having testing their code. A detail we highlight from Sheth et al.'s work is that they did not use achievements and a leader board as game elements in the prototype.

Bell et al. [3] presented a methodology called "secret ninja", and proposed to combine it to HALO (Highly Addictive, sociaLly Optimized) Software Engineering, an engaging and collaborative environment that uses MMORPG (Massively Multiplayer Online Role Playing Game) design. This environment can be integrated to an IDE as a plugin. Combining HALO to an adaptation of secret ninja, the authors aimed to motivate students to test their code, but disguising the teaching of software testing concepts into stories and quests. Other goals of using HALO are increasing engagement throughout the software development lifecycle, and improving students' learning. The gamification goal is to add enjoyment to the learning process; thus, they hope to increase productivity and satisfaction. The used game elements are points (experience points), levels, quests, achievements, leader board, and teams. The authors concerned to investigate in the future how to avoid students' discouragement due to a reward tracking system (*e.g.* leader board). Also, they defend the beneficial controlled addiction to increase productivity. As HALO is a tool to teach software testing, we understand that any testing technique, level, and process phase can be taught, thus encouraging "a solid foundation of testing habits, leading to future willingness to test in both coursework and in industry", as mentioned by them. Their future work aims to evaluate the impact and benefit the environment can cause to students.

Passos et al. [21] proposed and evaluated an approach to gamify software development process by incorporating RPG-like mechanics in everyday activities. The authors differed their work from previous research arguing that their proposal is not to incorporate gamification in a interactive media or teaching, but in real-world activities instead. Besides this, they highlight that their "proposal is not intended for classroom use". The gamification goal was to make the developers more aware of their progress, aiming to engage them in their tasks. The approach proposed is based on challenge-punishment-reward loop, in which the used game elements were points, badges (medals), levels, teams, quests, and achievements. A case study was done in a company to evaluate a set of designed achievements. Both individual and team achievements were set up and, based on the results, the developer (or team) earns a medal (which we classified as a kind of badge). The evaluation revealed that the achievements element helped to increase engagement, stimulate competition among developers, served as a manner to monitor and control the development process, and can be used as a metric to measure performance comparing estimated with actual time spent to accomplish tasks. The authors also proposed to gamify a task-management tool incorporating DevRPG elements, which goals are to give immediate feedback aiming to engage the developers in the challenges, and stimulate competitiveness among them. Notice that this study [21] indirectly addresses software testing because the proposal is applicable to gamify software development process, which includes software testing activities. Moreover, we understand that any testing technique, level, testing process phase can be gamified by applying this approach.

Dubois and Tamburrelli [8] compared their proposal with the approach proposed by Passos et al. [21] and argued that the main limitation of Passos et al.'s approach was the lack of explanation on *how* to use gamification. This limitation is addressed in Dubois and Tamburrelli [8]'s study. The authors proposed an approach to gamify phases of the software engineering lifecycle aiming to improve software engineering education, and improve quality of development activity. Other gamification goals applied to this study are to increase engagement and motivation, improve training, and enhance monitoring of people involved in software development activities. The approach can both be integrated with other tools or approaches, and applied to educational or industrial contexts. Cheating behaviours were discouraged by the identification of manners to prevent them. The authors discussed a possible way to apply their approach presenting how to use gamification in an educational context; however, their example was generic in terms of game elements; they cited the reward mechanism, but did not define which concrete elements could be used to reward an student. The authors also observed positive outcomes in courses taught with the gamification approach. So, they performed an experiment to understand the effect of competition between groups in two sections. The authors presented the results and affirmed that there was not enough evidence to conclude the role of competition. Even though Dubois and Tamburrelli [8]'s study does not focus on software testing, its generic nature led to its inclusion in our final set. Moreover, we also understand that any testing technique, level, testing process phase can be gamified by applying this approach.

5 ANALYSIS AND DISCUSSION

The following analysis and discussion are based on the perspectives listed in Section 3. Table 2 summarizes the overall classification.

5.1 Classification: *Application context*

We considered two contexts in which gamification was applied in the selected studies: *educational* (both academic and industrial), and only *industrial*. The *educational* context was considered in 10 studies: [1, 3, 6–8, 24–27, 29]. The *industrial* context was considered in 8 studies: [7, 11, 14, 15, 19, 21, 24, 26]. With a few exceptions, all approaches can be applied to both *educational* and *industrial* contexts. The exceptions are the approach of García et al. [11] which, as we understand, cannot be applied for educational purposes due to the fact that their framework was created to assist designers to gamify a workplace, not to teach software testing.; and the approaches proposed by Anderson et al. [1] and Yujian Fu and Clarke [29], which are meant for education because their systems are applicable to teach software testing, not to other industrial purposes such as to improve the employees' performance or to engage them to test their code improving the product's quality.

Despite the fact that some authors have mentioned a specific context to apply their proposals (academic education, for example), most of them can also be used in other contexts (industrial education, or even in industry for commercial purposes). For example, Passos et al. [21] affirm that their proposal is not intended to classroom use, but we see a real possibility to use it to teach software testing. The authors proposed an approach to gamify the software development process, highlighting that they gamified the developers' everyday activities in their real world. Passos et al. defined achievements that the developer (or the teams) needed to reach aiming to earn medals. If someone reached an impressive code coverage in his automated testing, an extra reward was given. Converting this to an educational context, we can consider to encourage students to learn software testing by gamifying the classroom. We can create quests they should solve, and define how many times they need to reach at least 80% of code coverage to earn a medal.

5.2 Classification: *Used gamification elements*

Points, *leader board*, and *levels* were the most commonly used elements. Only Dubois and Tamburrelli [8] did not use *points*. Both *levels* and *leader board* were each used in 10 studies. *Badges*, which composes the PBL triad (*Points*, *Badges*, *Leader board*), were mentioned in 6 studies. Werbach and Hunter [28] discuss about the PBL triad and affirm that it has its benefits, but highlight that they are superficial game elements. The authors also suggest that PBL can be used as an initial extrinsic motivator, as done by Liechti et al. [15], but they advise that it is necessary to go further in the other game elements to achieve the maximum gamification may offer. As another example, to reach the gamification goals intended in CODE DEFENDERS, Rojas and Fraser [24] incorporated elements into the game (*e.g.points*, *levels*, *duel*). In CODE DEFENDERS, *Points* are given to a defender based on how many mutants he/she killed; they are given to a attacker based on how many mutants survived. *Levels* are placed into the game to give to the players two options of difficulty (easy, hard). *Duels* occur when a defender labels a mutant as equivalent; so, the attacker has to prove that it is not equivalent by creating a test case that kills the mutant, or has to agree.

The only study that did not define concrete game elements was the presented by Dubois and Tamburrelli [8]. They gave an example of how to use gamification in an educational context. However, the authors only mentioned rewarding game mechanism, though not defining which concrete elements could be used to reward a student.

5.3 Classification: *Gamification goals*

A variety of gamification goals was identified in the studies. Examples are *boost adoption*, *stimulate collaboration*, and *develop creativity*. The full list and work distribution can be seen in Table 2. Notice that we unified different kinds of improvements, namely: efficiency, performance, creation of stronger tests, quality of bug reports, quality of tracing results, software engineering education, learning outcomes, and testing skills.

The top pursued goals were *increase engagement* [1, 3, 6, 8, 11, 14, 19, 21, 24–27, 29], *improve skills* [3, 6–8, 11, 19, 24–27, 29], *increase motivation* [1, 7, 8, 15, 19, 25–27, 29], and *increase enjoyment* [1, 3, 6, 15, 24–27]. Other goals were pursued by less than four studies.

5.4 Classification: *Software testing techniques*

The selected studies were classified into the following three techniques: *functional testing*, *structural testing*, and *fault-based testing*. Besides this, we included the option *any*, as aforementioned.

Specifically, *fault-based testing* was the most gamified technique. Apart from it, in total, 9 out of 15 studies proposed gamified approaches that, in our understanding, are applicable to any of the three techniques. For example, Yujian Fu and Clarke [29] mentioned that the WReSTT-CyLE tool contains tutorials and content to teach *functional testing* and *structural testing*; however, we classified their study considering that the system can be used to teach any testing technique, since other contents are possible to be added in the system. At the same way, *functional testing* and *structural testing* were cited by Liechti et al. [15], but we also classified that any testing technique can be used in their approach.

Regarding the 9 studies classified as applicable to any technique, Yujian Fu and Clarke [29], Sheth et al. [27], and Bell et al. [3] proposed gamified systems to support software testing teaching/learning. The proposal of Anderson et al. [1] did not focus on teaching software testing concepts, but involved them in the lessons. García et al. [11], Dal Sasso et al. [7], Passos et al. [21], and Dubois and Tamburrelli [8] proposed approaches to gamify any software engineering activity. We considered that those proposals may be applied to any of the software testing techniques for its generic and/or flexible characteristics.

Table 2: Overall study classification ("#" stands for "number of studies").

Category	#	Study IDs	Category	#	Study IDs
USED GAME ELEMENTS			**APPLICATION CONTEXT**		
Achievement	3	[3] [21] [8]	Educational	2	[1] [29]
Avatar	3	[7] [19] [21]	Industrial	1	[11]
Badge	6	[11] [1] [29] [7] [15] [21]	Any	12	[24] [26] [7] [19] [15] [14] [27] [3] [25] [6] [21] [8]
Duel	6	[24] [26] [1] [25] [6] [8]	**TESTING TECHNIQUES**		
Leader Board	10	[11] [1] [29] [7] [19] [15] [14] [3] [25] [8]	Functional Testing	0	n/a
Level	10	[11] [24] [26] [29] [7] [27] [3] [25] [6] [21]	Structural Testing	1	[19]
Points	14	[11] [24] [26] [1] [29] [7] [19] [15] [14] [27] [3] [25] [6] [21]	Fault-based Testing	5	[24] [26] [14] [25] [6]
Quest	5	[11] [19] [27] [3] [21]	Any	9	[11] [1] [29] [7] [27] [3] [21] [8]
Social Graph	1	[11]	**TESTING PROCESS PHASES**		
Team	5	[26] |[29] [27] [3] [25]	Planning	0	n/a
Virtual goods	1	[7]	Data / environ. configuration	0	n/a
GAMIFICATION GOALS			Test cases design	7	[24] [26] [19] [15] [14] [25] [6]
Increase awareness	1	[21]	Execution and evaluation	6	[24] [26] [15] [14] [25] [6]
Boost adoption	1	[24]	Monitoring and control	5	[24] [26] [15] [25] [6]
Develop creativity	1	[7]	Maintenance	6	[24] [26] [19] [15] [25] [6]
Ease the fixing process	1	[7]	Any	8	[11] [1] [29] [7] [27] [3] [21] [8]
Encourage testing habits	2	[27] [3]	**TESTING LEVELS**		
Increase engagement	13	[11] [24] [26] [1] [29] [19] [14] [27] [3] [25] [6] [21] [8]	Unit Testing	6	[24] [26] [19] [14] [25] [6]
Improve skills	11	[11] [24] [26] [29] [7] [19] [27] [3] [25] [6] [8]	Integration Testing	0	n/a
Increase enjoyment	8	[24] [26] [1] [15] [27] [3] [25] [6]	System Testing	0	n/a
Increase motivation	9	[26] [1] [29] [7] [19] [15] [27] [25] [8]	Any	9	[11] [1] [29] [7] [15] [27] [3] [21] [8]
Enhance monitoring	1	[8]			
Increase persuasion	1	[15]			
Stimulate collaboration	3	[7] [27] [3]			
Improve training	1	[8]			

5.5 Classification: *Testing levels*

We classified studies according the following levels: *unit testing*, *integration testing*, and *system testing*. Option *any* was applied to studies that (potentially) address gamification in any testing level.

Authors that explicitly mentioned a testing level were: Anderson et al. [1], García et al. [11], Laurent et al. [14], Liechti et al. [15], Parizi [19], Rojas and Fraser [24, 25], Rojas et al. [26] and Clegg et al. [6]. All these authors mentioned the *unit testing* level. Liechti et al. [15] were the only authors that also explicitly took into account the other two levels (namely, *integration testing* and *system testing*.

Bell et al. [3], Dal Sasso et al. [7], Sheth et al. [27], Yujian Fu and Clarke [29] and Dubois and Tamburrelli [8] did not specify a testing level in their studies. However, considering our reading and understanding of their studies, we classified them as applicable to any testing level. The studies of Anderson et al. [1], García et al. [11] and Liechti et al. [15], which address *unit testing*, are also proposals applicable to any other level.

5.6 Classification: *Phases in the testing process*

At first, notice that *maintenance* was not considered as an option in our list of testing process phases. We added it when we observed that a study [19] clearly defined the phases that the proposed approach would comprehend (*test cases design*, and *maintenance*). Moreover, Liechti et al. [15] mentioned that the *maintenance* of an automated test suite was a reason to use the Fogg Behaviour Model.

Regarding the obtained classification, it is important to note that, even though some studies have the same classification across the process phases [6, 15, 24–26], we could not classify them as applicable to any. It is due to the fact that they are not applicable to the *planning* and *data and environment configuration* phases.

The studies of García et al. [11], Anderson et al. [1], Yujian Fu and Clarke [29], Dal Sasso et al. [7], Sheth et al. [27], Bell et al. [3], Passos et al. [21], and Dubois and Tamburrelli [8] were classified as applicable to *any* testing process phase following the same rationale for testing technique: they are either generic or flexible approaches.

Parizi [19]'s work is an example of a study that we classified with multiple testing phases. The author presented a new approach to create trace links between test and code artifacts. As aforementioned, it addresses the following phases: *test case design*; and *maintenance*. The trace links are embedded into the test code in the moment the test developers are designing the tests. In the *maintenance* phase, the traceability information can be extracted and an automatic Link Retriever tool specifies the validity of the trace links' status. With his proposal, the effort to verify the links would be reduced, and only invalid status would need to be analyzed and fixed.

5.7 Further Discussion

Several gamified approaches proposed in the studies mentioned automated software testing activities, most of those in unit testing level. For example, Rojas et al. [26] used three automated tools with CODE DEFENDERS: EvoSuite, Randoop, and Major. The first two automatically generates unit tests for Java classes in JUnit format. Major, on the other hand, is a mutation testing tool. These tools were used to support the single-player mode of the game, and to compare the created tests and mutants.

Parizi [19] listed some approaches of automatic traceability. He also discussed that empirical experiments revealed some poor results in precision and recall, despite their high applicability. Another important observation he made was that results from automated techniques (specially if applied in critical systems) should have

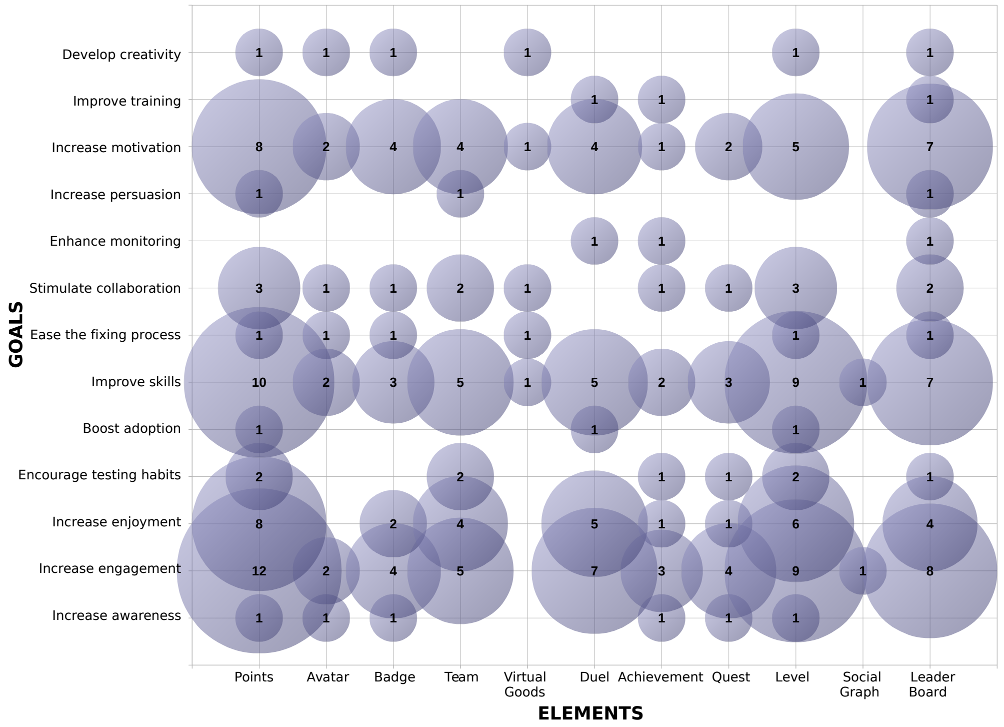

Figure 1: Number of studies that relate gamification elements to gamification goals.

human inspection. The gamified approach proposed by the author revealed better results in accuracy than a non gamified system.

Liechti et al. [15] stated that automated testing increases the software quality significantly. In their work, they presented a concept of test analytics, which is composed by key practices in agile methodologies: continuous self-improvement, feedback mechanisms, and automated testing. The authors used the Fogg Behaviour Model to drive employees' behavior, aiming to make them rigorous while maintaining automated test suites. Gamification was used to add fun to the process, thus motivating the employees to write tests.

Passos et al. [21] proposed gamification to engage developers in their daily activities. Despite the use of varied game elements for motivation purposes, *achievements* was on focus of the study. They created an *achievement* to reward teams by excellence in automated testing; code coverage was used as a measure. That is, they stressed the importance of performing automated testing appropriately.

5.8 Revisiting our Research Question

We analyzed the selected studies and classified them according to six perspectives. The classification based on the *application context* revealed that almost all proposals comprehend both educational and industrial contexts, intentionally or not. Besides that, the most *used game elements* were points, levels, and leader boards, whereas the main *gamification goals* in software testing activities were increasing engagement, improving skills, and increasing motivation. The bubble chart of Figure 1 shows combined results based on the last two aforementioned classifications: *used game elements* and *gamification goals*. In the chart, considering the vertical reading, the data labels (and the sizes of the bubbles) represent the numbers of times a given game element was addressed in a study that pursued a given goal. For instance, *points* were used in 12 studies that investigated gamification to *increase motivation*.

Werbach and Hunter [28] argue that "gamification is about engagement". Our results corroborate it. Those authors also observe that the PBL triad (points, badges, leader board) are the most basic game elements. Once again, our results corroborates it[11]. Taking into account the other classifications we applied to the selected studies, we draw the following answer to our research question:

Gamification has been investigated to support software testing mostly with the application of *basic game elements (such as points, levels and leader boards)*, and with the aim of *increasing engagement and motivation, and improving skills*, without any clear focus on particular testing technique, level or process phase.

6 LIMITATIONS AND RELATED WORK

The scope of the search for primary studies is key concern in our study, since it is restricted to: (i) a single search engine (namely, Scopus); and (ii) a one-step, backward snowballing round. Moreover, in our search we have used a narrowed list of gamification- and testing-related terms. That said, we argue that the Scopus engine retrieves a broad range of results since it surveys a plethora of indexed databases [13]. Nevertheless, we believe the inclusion of other repositories and search terms (such as "software quality" and

[11]Noticeably, *level* is another very basic game element that stood out in Figure 1.

"development process") may result in the selection of other relevant studies that can enhance and extend our classifications. Besides this, more rounds of backward snowballing, as well rounds of forward snowballing, are also worth to be done.

Regarding related research, Pedreira et al. [22] performed a systematic mapping to characterize gamification applied to the software development process. They analyzed 29 studies published until 2014 and answered questions about software engineering processes that have been gamified, used game elements, and research methods that were used on software gamification quality evaluation. Our goals differ from Pedreira et al.'s, given that we did not focus on all software development activities but software testing.

Mäntylä and Smolander [16] presented a very brief report on a literature review they carried out about gamification of software testing. They used the study of Pedreira et al. [22] to perform forward and backward snowballing. After this, they searched for more studies using the Google Search Engine. In total, they selected 20 items[12] and quantified results regarding types of testing, systems under test, roles of individuals, used game elements, studies that performed empirical evidence, studies that presented support constructs to gamification, and challenges to gamify software testing. In our work, we also analyzed the retrieved papers based on types of testing (more specifically, *testing techniques*), and used game elements. Differently from their research, we provided deeper information; we classified the selected studies considering three perspectives about software testing (e.g. techniques, levels, and process phases); besides this, we gave more fine-grained details about the applied classifications and results.

Fraser [10] discussed issues software testing has been facing, and introduced gamification as a solution to address them. He presented applications of software testing in three domains: education, practice, and crowdsourcing. Differently from our work and from prior work [16, 22], Fraser did not map, review, or characterized studies found in the literature; not even tried to answer a research question. Instead, he raised and presented the potential of use of some existing gamified tools based on its application domain.

7 FINAL REMARKS

As gamification is a new trend in Software Engineering, we believe it is important and convenient to monitor its increase, specially in the software testing field. Secondary (mapping) studies are meaningful tools to do it; they map, review, and/or characterize the state-of-the-art in a subject to reveal gaps and research opportunities. Besides the direct contributions of our work (*i.e.* the characterization of gamification as a supporting technique for software testing), it is also intended to support the design of new gamification approaches. We believe it may be useful due to the characterization we have done, for instance, for researchers and practitioners that aim to explore combined perspectives such as "*which game elements were used to increase student's engagement?*"; or "*which game elements were used to motivate students to learn mutation testing?*". Both educational and industrial contexts may benefit from our results.

[12]Notice that from the 20 *items* selected by Mäntylä and Smolander [16], only 2 could have been potentially included in our dataset. From the remaining ones, 2 items are studies also selected by us, 14 are not proper literature material (*e.g.* web sites and slides) and 2 did not pass our selection criteria.

For future work, we plan to enlarge our set of selected studies by running rounds of forward snowballing, as well as additional rounds of backward snowballing. Moreover, we intend to analyze more deeply some studies that were not focused in software testing, but were included in our dataset due to their general characteristics regarding the software development process and activities.

ACKNOWLEDGMENTS

The authors thank the financial support received from CNPq (grants #167513/2017-6 and #306310/2016-3), CAPES and Federal University of São Carlos.

REFERENCES

[1] P. E. Anderson, T. Nash, and R. McCauley. 2015. Facilitating Programming Success in Data Science Courses Through Gamified Scaffolding and Learn2Mine. In *ITiCSE*.

[2] P. E. Anderson et al. 2014. An Extensible Online Environment for Teaching Data Science Concepts Through Gamification. In *IEEE FIE Conference*.

[3] J. Bell, S. Sheth, and G. Kaiser. 2011. Secret Ninja Testing with HALO Software Engineering. In *SSE Workshop*.

[4] Capgemini Group, Sogeti, and HP. 2016. *World Quality Report 2015-16*. Tech. Report. Cap Gemini S.A., Sogeti and Hewlett-Packard. https://www.capgemini.com/thought-leadership/world-quality-report-2015-16 - last accessed on 19/06/2018.

[5] P. J. Clarke et al. 2014. Integrating Testing into Software Engineering Courses Supported by a Collaborative Learning Environment. *Trans. on Comp. Education* 14, 3 (2014).

[6] B. S. Clegg, J. M. Rojas, and G. Fraser. 2017. Teaching Software Testing Concepts Using a Mutation Testing Game. In *ICSE SEER Track*.

[7] T. Dal Sasso et al. 2017. How to Gamify Software Engineering. In *SANER Conf.*

[8] D. J. Dubois and G. Tamburrelli. 2013. Understanding Gamification Mechanisms for Software Development. In *ESEC/FSE*.

[9] S. C. P. F. Fabbri et al. 2013. Externalising Tacit Knowledge of the Systematic Review Process. *IET Software* 7, 6 (2013).

[10] G. Fraser. 2017. Gamification of Software Testing. In *AST Workshop*.

[11] F. García et al. 2017. A Framework for Gamification in Software Engineering. *Journal of Systems and Software* 132 (2017).

[12] J. Hamari, J. Koivisto, and H. Sarsa. 2014. Does Gamification Work? – A Literature Review of Empirical Studies on Gamification. In *HICSS Conference*.

[13] B. A. Kitchenham, D. Budgen, and P. Brereton. 2016. *Evidence-based Software Engineerg and Systematic Reviews*.

[14] T. Laurent et al. 2017. Towards a Gamified Equivalent Mutants Detection Platform. In *ICST Conference - Posters Session*.

[15] O. Liechti, J. Pasquier, and R. Reis. 2017. Supporting Agile Teams with a Test Analytics Platform: A Case Study. In *AST Workshop*.

[16] M. V. Mäntylä and K. Smolander. 2016. Gamification of Software Testing - An MLR. In *PROFES Internl. Conference – Poster Session*.

[17] E. S. Mastrodicasa. 2014. *A Gamification Framework for Software Engineering*. Master's thesis. Faculty of Informatics, Università della Svizzera Italiana.

[18] G. J. Myers, C. Sandler, and T. Badgett. 2011. *The Art of Software Testing*.

[19] R. M. Parizi. 2016. On the Gamification of Human-centric Traceability Tasks in Software Testing and Coding. In *SERA Conference*.

[20] R. M. Parizi, A. Kasem, and A. Abdullah. 2015. Towards Gamification in Software Traceability: Between Test and Code Artifacts. In *ICSOFT Conference*.

[21] E. B. Passos et al. 2011. Turning Real-World Software Development into a Game. In *SBGAMES Symposium*.

[22] O. Pedreira et al. 2015. Gamification in Software Engineering - A Systematic Mapping. *Information and Software Technology* 57 (2015).

[23] K. Petersen et al. 2008. Systematic Mapping Studies in Software Engineering. In *EASE Conference*.

[24] J. M. Rojas and G. Fraser. 2016. Code Defenders: A Mutation Testing Game. In *Mutation Workshop)*.

[25] J. M. Rojas and G. Fraser. 2016. Teaching Mutation Testing using Gamification. In *ECSEE Conference*.

[26] J. M. Rojas et al. 2017. Code Defenders: Crowdsourcing Effective Tests and Subtle Mutants with a Mutation Testing Game. In *ICSE Conference*.

[27] S. Sheth, J. Bell, and G. Kaiser. 2012. *Increasing Student Engagement in Software Engineering with Gamification*. Tech. Report CUCS-018-12. Columbia University.

[28] K. Werbach and D. Hunter. 2012. *For the Win: How Game Thinking Can Revolutionize Your Business*.

[29] P. E. Yujian Fu and P. J. Clarke. 2016. Gamification-Based Cyber-Enabled Learning Environment of Software Testing. In *ASEE Conference & Exposition*.