FROM RATIONALISM TO EMPIRICISM IN EDUCATION OF SOFTWARE TESTING USING GAMIFICATION

N. Doorn[1,3], T.E.J. Vos[1,2], B. Marín[2]

[1]Open Universiteit (NETHERLANDS)
[2]Universitat Politècnica de València (SPAIN)
[3]NHL Stenden University of Applied Sciences (NETHERLANDS)

Abstract

In the industry, software testing is considered a *de facto* technique for assessing the quality of software. However, many Computer Science lecturers struggle to integrate software testing into their courses. This is due to a myriad of factors, one of which is the lack of a fully developed body of knowledge on the didactics of teaching software testing. In our previous work, we identified that most students employ a so called 'developer approach' when modelling test ideas for a software system. This approach shows that students primarily use conceptual knowledge gained from programming courses in their testing strategy. These students see testing as a form of problem solving and apply an approach stemmed in rationalism. We believe software testing should not be taught solely from rationalism but should include much more empiricism. Tests should be small scientific studies where students use heuristics and exploration to formulate hypotheses about the expected workings of the system, and then conduct an experiment to find answers to these hypotheses and evaluate how the system responds. To shift the student's mental model to empiricism, we started with the application of gamification and the development of a serious game, where students devise a testing strategy for a system under test through dynamic gameplay. To check the impact of this approach, we have conducted a pilot study featuring the testing concepts of the game with computer science students of the NHL Stenden University of Applied Sciences and the Open Universiteit during the second semester of 2023. Results show that students perceived an improvement in the quality of their testing strategies. Based on these experiences, we are developing a serious game to include empiricism in software testing education. The objective of the game is to achieve goals related to the system's key quality attributes and to mitigate development risks. As the main game mechanic, we want to incorporate ways to enhance critical thinking using socrative questioning.

Keywords: Software Testing Education, Gamification, Empiricism.

1 INTRODUCTION

Software testing, a critical component of the software development lifecycle, is a highly sought-after skill in the industry. Lecturers of Computer Science programs struggle to include software testing in their courses. Consequently, extensive research on the teaching practices of software testing in higher education has emerged, as observed in [1], [2].

In previous work [3], [4], we have explored students' sensemaking processes when testing in order to develop didactic approaches that improve undergraduate testing education. We find that students mainly use a 'developer approach' when modeling test ideas for a software system, which reflects that testing is predominantly taught within a rational design paradigm that emphasizes algorithmic problem solving, planning, and methods, similar to the approach used in teaching programming. This approach often results in inappropriate and deficient testing practices because there is a failure to explore and experiment with the software to generate new knowledge, which is essential for making informed judgments about software quality. To address this, there is a need to shift the teaching of testing toward the empirical design paradigm, emphasizing reflection-in-action. Empiricism emphasizes experience, experimentation, and evidence-based learning. In software testing, this might involve practical, hands-on experiences, exploring different testing scenarios, and learning from actual outcomes.

Gamification, defined as the use of game elements in non-gaming contexts [5], has been used extensively to improve the teaching of complex topics such as testing [6]–[9]. The main benefits of using game-based learning to enhance critical thinking include: **enhanced engagement**, game-based learning provides an immersive and interactive environment, keeping learners engaged and motivated; **practical application**, Games offer a safe space for applying and practicing critical thinking skills; **immediate feedback**, game-based learning allows for instant feedback, helping learners understand

This article will be included in the INTED2024 Proceedings (ISBN: 978-84-09-59215-9)
It will be fully citable as soon as it appears in IATED Digital Library (library.iated.org)
This version should not be distributed since it may change prior to final publication

the consequences of their decisions and refine their thinking process; **collaborative learning**, many games encourage collaboration, fostering teamwork and collective problem-solving; **complex problem-solving**, games often present complex scenarios, requiring players to analyze, evaluate, and create solutions, thereby sharpening their critical thinking skills. These benefits can be helpful in mitigating the current problems in teaching software testing in computer science education, including improved intrinsic motivation and learning performance [10]–[14].

Socrative questioning [15] is a technique of questioning or leading discussion that aims to expose and unravel deeply held values and beliefs. It is spontaneous, exploratory, and issue-specific. This method involves a disciplined and thoughtful dialogue between two or more people, where the Socratic educator listens to and considers the alternative points of view of the student. It's often used in teaching and counseling to expose and unravel deeply held values and beliefs that frame and support what we think and say. By using a series of focused yet open questions, the method aims to unpack our beliefs and those of others, enhancing critical thinking and understanding. Socratic questioning encourages critical thinking, exploration, and reflective inquiry, which are key components of empiricism.

Our **hypothesis** is that using gamification with Socratic questioning can improve the learning of software testing in computer science education in different educational contexts, emphasizing reflection-in-action. Our first step in this development was to design a game with a board and cards using the concrete elements of TestSphere [16] with the addition of a set of socrative questions. In the proposed game, students formulate a test strategy for a system under test through interactive gameplay. The primary goal of the game is to achieve specific objectives related to the quality attributes of the system and the risks to be avoided in the development process. During the gameplay, we use the socrative questioning to facilitate critical thinking.

Our main contribution in this paper is an in-depth literature review specifically on gamification of software testing education and the application of socrative questioning or other means to improve critical thinking skills, together with a presentation of the design of our game in terms of game mechanics and the reporting on our experiences with the application of gamification in software testing workshops.

The rest of this article is structured as follows. Section 2 discusses existing literature on software testing education and the role of serious games. Section 3 presents our preliminary outcomes of using games in an educational setting. This is followed by Section 4, which presents the approach for the development of the serious game including socrative questionning. Finally, Section 5 presents our conclusions and an insight into our future work. Each section contributes to the overarching theme of transitioning from a rationalist to an empirical approach in teaching software testing.

2 RELATED WORK

For this game's development, we reviewed literature on serious gaming in computer science education, specifically focusing on software testing. We also searched for socrative questioning in games applied in higher education using Google Scholar. We used both academic sources and gray literature for completeness. We explain the search and selection process, and a summary of the results obtained in the sub-sections below.

2.1 Search method: mapping review and gray literature

We conducted a mapping review [17] to gather serious games and gamified approaches to testing education in the major computer science digital libraries, and we also included gray literature that often provides approaches used in testing practice.

In our search approach we included often uses synonyms for completeness. The search process used the ACM Digital Library[1], and the IEEE Xplore digital library[2] to search for scientific publications about gamification or serious games for testing that are relevant for the computer science educational domain. Moreover, we use Google Scholar[3] database to search for publications of Socratic questioning.

To limit the results to relevant publications, we used filters based on the following CRAAP [18] criteria: *currency*, only publications from the last five years are considered; *relevance*, by using well defined

[1] https://dl.acm.org/

[2] https://ieeexplore.ieee.org/

[3] https://scholar.google.com/

This article will be included in the INTED2024 Proceedings (ISBN: 978-84-09-59215-9)
It will be fully citable as soon as it appears in IATED Digital Library (library.iated.org)
This version should not be distributed since it may change prior to final publication

queries and relevant search engines only the most relevant publications are selected, with additional gray literature for completeness; *authority*, only peer reviewed publications of high quality journals and conferences are considered by using the aforementioned publishers; *accuracy*, publications that use methodologies that can be replicated and supported by data sets are preferred; *purpose*, publications focused on the use and development of serious gaming for higher education are selected, preferably with application in computer science or software engineering education.

The following queries have been used:

- **ACM Digital Library on serious gaming in CS education:**

  [[**All**: "serious gaming"] **OR** [**All**: "serious game"] **OR** [**All**: "gamifying"]] **AND** [**All**: "education"] **AND** [**All**: "software testing"] **AND** [E-Publication Date: Past 5 years]

- **IEEE Xplore on serious gaming in CS education:**

  ("Serious Gaming" **OR** "Serious Game" **OR** "Gamifying") **AND** "Education" **AND** "Software Testing" **AND** ("Publication Year": 2018 - **Current**)

- **Google Scholar on socrative questioning:**

  ("Inquiry-based learning" **OR** "Dialogic teaching" **OR** "Critical questioning" **OR** "Interactive probing" **OR** "Reflective questioning" **OR** "Pedagogical questioning" **OR** "Investigative discourse" **OR** "Exploratory questioning" **OR** "Thought-provoking inquiry" **OR** "Constructive interrogation") **AND** ("Serious Gaming" **OR** "Serious Game" **OR** "Gamifying") **AND** "higher education"

The ACM and IEEE queries resulted in 70 results, containing five duplicates of publications of joined ACM/IEEE conferences. These publications were assessed using the aforementioned criteria based on their abstracts. This let to a selection of 21 publications that were further used in this related literature section. The Google Scholar query on socrative questioning resulted in 92 results. Again, these results where assessed using the aforementioned criteria based on their titles and abstracts. Most of these results are not relevant for this study, 31 results target a specific technology such as virtual reality, 23 results are studies specifically related to a particular subject such as mathematics or science education, five studies are targeted at non-higher education, three studies are specific to game design, fourteen studies are on general serious gaming, three results were not related to education, and five studies where written in a language not understood by the authors. This led to a selection of four publications that were further used in this related literature section.

Since many innovations in training of software testing skills take place in the software development industry which often do not result in publications in the mainstream academic literature, we conducted an additional gray literature search using Google[4] with the following query:

- **Google on serious gaming in software testing in the industry:**
- ("Serious Gaming" **OR** "Serious Game" **OR** "Gamifying") **AND** "software testing"

We filtered these results to only included results from the last five years. We scanned these results for applications of serious gaming in software testing not present in the academic literature, but relevant or applicable in education. This led to a selection of three sources that where further used in this related literature section.

Other by the authors known software testing communities where gaming is applied are “The Ministry of Testing”[5] and the developers of the Rapid Software Testing Methodology[6] [7]. We searched and included relevant results from those communities in this literature summary as well.

2.2 Summary of existing literature on gamification of software testing

Many **gamification techniques, tools, and implementations** can be found in the literature. The literature highlights several effective gamification methods in software engineering education, such as the use of real-world scenarios, competitive elements, immediate feedback, interactive activities, and collaboration [19]. A gamified tool named ’Testable’ [20] focuses on improving unit testing teaching

[4] https://www.google.com"
[5] https://ministryoftesting.com"
[6] https://rapid-software-testing.com/
[7] https://www.kenst.com/the-testopsy/

through structured and object-oriented programming. Furthermore, the impact of gamification and power posing on coding efficiency is studied [21], suggesting gamification has the potential to enhance developer efficiency. The game to learn functional testing named Testing Maze [22] assists undergraduate students in learning functional testing concepts. CleanGame [23], a gamified tool for teaching code smell identification shows that, on average, participants managed to identify twice as much code smells during with a gamified approach in comparison to a non-gamified approach. Identifying code smells helps to find possible refactoring needs in software, which helps to create maintainable code, leading to less defects. The Code Defenders game [24], [25], applied in university software testing courses on the concept of mutation testing, successfully encourages competition among students. Mutation testing [26] is a way to create new test cases by making small modifications to software and design tests to detect these changes. It has an inherited game mechanism by its approach, making it ideal to use as a basis for gamification. Additionally, we also found the application of gamification in GUI testing of web applications [27].

Gamification can be used with different **educational strategies** and in different **contexts**. The application of gamification in the software measurement process using gamified elements can enhance student engagement and learning [28]. Gamified platforms can be used in software testing education, involving user stories, test cases, and abuse cases within a blended learning approach [29]. Additionally, these platforms can also be used to integrate modern technology stacks and testing practices in software engineering courses, emphasizing automated testing and code coverage tools [30].

Applications of **innovative tools and techniques** can be found withing the computer science educational domain. The development of a chatbot designed to enhance software testing education [31] aims to support the learning process through conversational interaction. The use of serious games in secure programming is investigated in a pilot study [32], focusing the impact of gamification on attitudes and abilities in secure programming by replicating earlier studies, with no results regarding it's effect on education due to the setup of the study.

Studies on the **educational impact and methodologies** show successful interventions in education related to software testing, software quality and software engineer practices. A methodology for evaluating game-based interventions, combining quantitative and qualitative data, is proposed after a tabletop security game was studied [33]. The Scrum Game Challenge [34] uses LEGO4Scrum to simulate city reconstruction and facilitate learning of Scrum methodologies, an agile method often used when developing software, both in industry and education, that has a focus on testing using a definition of done. A tertiary study in software engineering education [35] investigates the application of gamification, especially in software testing and software quality. The challenges of teaching software testing remotely during the COVID-19 pandemic show the need for other forms of active approaches and methodologies to make their teaching and learning process more efficient [36].

Literature about **challenges and future directions** show the importance of assuring meaningful competition, the need for engagement and the risk of oversimplification of the taught concepts [37]. On the other hand, the potential negative impacts of gamification in education, such as decreased intrinsic motivation and fostering unhealthy competition [35]. Lastly, the role of gamification in software engineering, focusing on its general application in the industry, including in software testing, is reviewed [38].

Looking at existing **offline games** related to software tesing, we found two card games in the Ministry of Testing community. A gamified approach in the form of a table top game is TestSphere [39] which is aimed at getting (professionals) thinking and talking about software testing and quality. TestSphere consists of a pack of cards featuring 100 cards divided into five categories: **heuristics:** approaches to solve a problem based on previous experiences; **techniques:** ways to test a system; **feelings:** feelings of testers that need to be addressed; **quality Aspects:** possible non-functional aspects of a system of importance; and finally **patterns:** patterns used in testing, but also biases. Each of these 5 categories has 20 cards and each card features a testing concept with 3 examples that put that concept in a different light. By itself, it has no game mechanism, but it is often used to support so called risk storming, a process to identify as many relevant risks and mitigation techniques as possible for a system under test. This process consists of three rounds of game play: first, the **identification of quality aspects**, where a maximum of six most important quality aspects from the deck of cards are identified for a system under test; secondly, **brainstorming of risks**, where the players identify and write down as many potential risks related to the selected quality aspects as possible; finally, **selection of techniques** from the remaining cards of the deck to mitigate the identified risks. This approach supports collaboration among professionals to identify risks and to define testing strategies for a system under test.

This article will be included in the INTED2024 Proceedings (ISBN: 978-84-09-59215-9)
It will be fully citable as soon as it appears in IATED Digital Library (library.iated.org)
This version should not be distributed since it may change prior to final publication

The card deck “Would Heu-Risk It?” [40] (a word play on “would you risk it?” and the term heuristics) has a similar function as TestSphere and provides ample opportunity to use as a gamification instrument. It can be used as part of a teaching strategy, as a way to improve collaboration within a team, or can be used to help in designing test cases. It consists of a deck of cards, divided in three categories: **trap** cards can be used to find weak spots in an existing testing strategy; **tool** cards are cards describing techniques or approaches that can be applied; **weapon** cards are meant to teach about complex areas of coding, to help prioritize or to come up with new test cases. Each card contains a title and a description. The website suggests underdeveloped ways to use this card deck as a game.

Another form of **interactively improving software testing** is used by “The TestOpsy”, which is a concept for dissecting and learning about the testing process [41]. It involves detailed analysis of testing sessions, potentially recorded on video, to uncover heuristics, techniques, and insights. The approach is aimed at building skills in observation, narration, and framing tests, with the potential to discover new techniques and foster discussion on test design. The concept emphasizes the importance of understanding and articulating one’s testing process for deeper learning and skill development.

Blackbox Puzzles [42] is a tool for testing software. It provides puzzles that are designed to improve problem-solving skills for software testers. These puzzles offer a hands-on experience in tackling software testing challenges without having full knowledge of the system (akin to black-box testing). The puzzles serve as an engaging, practical method to enhance understanding and application of black-box testing techniques in real-world scenarios.

A systematic review on the impacts of **game-based learning on argumentation skills** [43] presents a comprehensive analysis of the use of game-based learning to enhance argumentation skills. The study underscores the potential of games as powerful tools for improving students’ argumentation abilities. It systematically reviews 29 publications from 2000 to 2019, focusing on the effectiveness of game-based learning in fostering argumentation skills. The study explores various elements like instructional supports, game elements, learning theories, and game genres in these environments, and their impact on learning outcomes related to argumentation. The results provide valuable insights for designing environments to support the development of argumentation skills.

The literature review presented in [43] highlights the importance of instructional supports, the role of various game elements, and the relevance of learning theories and game genres in game-based learning environments. These insights are critical for designing effective game-based learning experiences that specifically target the development of argumentation skills. These game elements include game design aspects like story lines, characters, and challenges, which are instrumental in engaging learners and fostering critical thinking. Additionally, interactive features and feedback mechanisms within games play a key role in developing argumentation skills. The article also mentions the development of questioning skills as a part of enhancing argumentation abilities through game-based learning. This involves nurturing the ability to formulate questions, probe assumptions, and critically evaluate arguments, which are essential components of effective argumentation. The use of game-based learning in this context aids in building these questioning skills by providing interactive and engaging scenarios for learners to practice and refine their abilities. Three other systematic literature reviews provide some more insights into teaching methods, immersive learning experiences, and the impact of gamification in educational contexts [44]–[46]. These studies show that that gamified learning environments enhanced students’ attitudes towards learning and improved their academic performance. They can also contribute to understanding the application of serious games in software testing education and the enhancement of critical thinking skills through various pedagogical approaches.

The review of existing literature on gamification in software testing, the use of serious games in computer science education, and the application of socrative questioning establishes a foundation for our approach to develop a serious game valuable in software testing education. Our results show that there are few applications of **serious gaming with concrete elements** applied in software testing education and even fewer of the application of **socrative questioning in serious games**. These two key aspects are taken into account for the development of our game.

3 EXPERIENCES WITH SOFTWARE TESTING GAMES IN EDUCATION

To study the effects of gamification, we conducted four sessions in which we used the risk storming [47] approach together with the TestSphere card set to create a test strategy for a system under test. Each session was started with a presentation on software testing in general and the use of risk storming.

This article will be included in the INTED2024 Proceedings (ISBN: 978-84-09-59215-9)
It will be fully citable as soon as it appears in IATED Digital Library (library.iated.org)
This version should not be distributed since it may change prior to final publication

Three of the four session took place with part-time students from different years from the Open Universiteit in The Netherlands. These were students from the “Informatics” and “Information Science” Bachelor programs, and students from the “Business Process Management & IT”, “Software Engineering”, and “Computer Science” master programs, The participants typically have day-to-day jobs in IT, resulting in diverse levels of conceptual knowledge and experience. Some of the participants worked as software testers. For these students, the following system under test, namely “Travel Agency”, was introduced:

You work for a travel company. The sales department wants to know what the average age is of the people who booked their holidays with your company. One of the developers in your team has developed a program to calculate the average age for a hundred people at the time. The program can handle up to a hundred dates of births and calculates the average age in years. It gets its data from a remote server as a .txt file, where each line contains the name and the age.

The remainder session took place with full-time second year students from the “Informatics” Bachelor program on the NHL Stenden University of Applied Sciences in The Netherlands. These students have a similar background and have far less experience in software testing and in system development in general. For this group of students, we did not use the “Travel Agency” system under test, they used the system from the project they where working on.

For the workshop the students where divided in similar size groups depending on the total number of participants. The group size across all these sessions ranged from four to six participants. The risk storming workshop was divided in three parts, first the participants identified the six most important quality aspects of the system under test. The second part was to identify risk related to these quality aspects and write them down on sticky notes. The third and final part was the matching of appropriate techniques to mitigate these risks.

During one of the sessions students could attend both online and in person. The other sessions where in person. Figure 1 shows a photograph of such a session. For the online session we created a digital version of the table top game.

Figure 1. Overview of the experience using the game with students, faces are blurred for privacy reasons.

To evaluate the use of gamification in these sessions we gathered anecdotal evidence. After the three parts of the workshop, the results were presented to the other groups. The results varied between the

This article will be included in the INTED2024 Proceedings (ISBN: 978-84-09-59215-9)
It will be fully citable as soon as it appears in IATED Digital Library (library.iated.org)

sessions and the groups. Since the selected quality aspects differed, the identified risks, and the chosen techniques to mitigate them differed as well. The time the group used for the three parts took also differed. Some groups discussed for a long time over the quality aspects, other groups quickly selected the aspects they found important for the system under test. This was also the case for the other phases. This might be due to the nature of the system under test, it is presented without much contextual information and the students could not ask for other information than that what was presented to them.

After each session, the participants were asked to discuss their experiences. This was very valuable as it provided us with insights into the use of the game. In general, all participants enjoyed the sessions. The use of the structured approach, together with the different insights from the other participants was considered to be a fun way to come up with aspects for a testing strategy. The use of the cards with the quality aspects and the techniques was considered by many to be insightful. For some participants they provided new information, for other participants with more relevant experience there were a good way to refresh their memory. The wide selection of techniques also broadened the horizons of the participants.

The full-time students who used their own system under test noted that the use of this approach made it easier for them to come up with a more complete testing strategy and they believed that this approach improved their testing strategy.

4 GAME DESIGN

Although the literature does not specifically address socrative questioning in game-based learning, it underscores the importance of argumentation skills in educational settings, closely linked to critical thinking and inquiry-based learning. Socrative questioning, known for stimulating critical thinking through questioning, could potentially be an effective tool in such learning environments. Its emphasis on inquiry and dialogue aligns well with the objectives of game-based learning in fostering deep understanding and analytical skills. Therefore, we hypothesis that there is a potential effectiveness of socrative questioning in conjunction with game-based learning to enhance argumentation skills. These skills are essential for an approach to software testing rooted in empiricism. Using this application of gamification can be helpful to change the student's mindset and contribute to a shift towards a mental model using empiricism.

Our game design's **educational objective** is to improve the student's conceptual knowledge of software testing using an empirical approach, their attitude towards software testing in general, their intrinsic motivation, and their awareness of the importance of software testing by developing a test approach for their system under test using a gamified approach. To maintain the student's interest, the game is high paced and **engaging**.

The game used a table-top format rather than a digital format to enhance the **interactivity** between the players. By using an existing or the student's own system under test, the game reflects a **real world situation**, this is important for the transfer of this knowledge and skills of this testing approach in real life. **Feedback** is an inherent part of software testing. By applying the developed test approach to a real system under test, the feedback is given by the behaviour of the system. Also, during game play, feedback is part of the interactivity among the players.

The use of socrative questioning together with the presentation of testing techniques and with the system under test provides a **story line** for the students. Since software testing by nature is a potential endless activity where is remains ever unclear if the system under test still contains potential failures, the game can be played indefinitely. The players should, together with a lecturer, determine if the resulting test approach is 'enough' for the moment. This can also be time boxed. This resembles a real life situation, and an important part of software testing in practise. By incorporating this aspect in the game, we provide ample opportunity for the players to **assess** their approach together.

The game is a strategic card game designed for 2-6 players, where each player takes on the role of a Software Testing Engineer. The game is a table-top game consisting a System Under Test (SUT), various Testing Techniques, a wheel of socrative questions (based on a wheel of fortune). Each player takes a turn to improve the strategy by spinning the wheel of socrative questions and asking the question selected to the other players. Socrative questioning in the game can lead students to question their assumptions, explore different testing scenarios, and learn from these explorations, aligning with the principles of empiricism. The asking of socrative questions between the students can help to re-evaluate the testing scenarios and chosen techniques, leading to a better testing strategy. The aim of this technique is to shift students' mindset from a purely theoretical or rationalist approach to a more hands-on, empirical approach.

This article will be included in the INTED2024 Proceedings (ISBN: 978-84-09-59215-9)
It will be fully citable as soon as it appears in IATED Digital Library (library.iated.org)
This version should not be distributed since it may change prior to final publication

5 CONCLUSION

In this paper, we explored a shift in teaching software testing in computer science education. We advocate for a more empirical approach, moving away from a purely rationalist mindset.

The paper discusses the development and application of a serious offline game designed to evolve students' mental models towards empiricism in software testing. We conducted a pilot study to evaluate the impact of the game on students' testing strategies. The results of the pilot study show that the students were very enthusiastic about the approach and that the game helped them with more and better test strategies than before.

After, we present the design of game that promotes critical thinking and a hypothesis-driven approach to software testing through the use of socrative questioning. As future work, we plan to develop and make available a fully functional prototype of the game, including the application of socrative questioning. We also plan to conduct empirical studies with undergraduate and masters students in four European countries with our partners in the Innovation Alliance for Testing Education (ENACTEST [48]).

ACKNOWLEDGEMENTS

This work was funded by the ENACTEST - European innovation alliance for testing education (ERASMUS+ Project number 101055874, 2022-2025).

REFERENCES

[1] V. Garousi, A. Rainer, P. Lauvås Jr, and A. Arcuri, "Software-testing education: A systematic literature mapping," *Journal of Systems and Software*, vol. 165, p. 110570, 2020.

[2] L. P. Scatalon, R. E. Garcia, and E. F. Barbosa, "Teaching practices of software testing in programming education," in *Frontiers in education conference (FIE)*, IEEE, 2020, pp. 1–9.

[3] N. Doorn, T. E. Vos, B. Marín, H. Passier, L. Bijlsma, and S. Cacace, "Exploring students' sensemaking of test case design. An initial study," in *21st international conference on software quality, reliability and security companion (QRS-c)*, IEEE, 2021, pp. 1069–1078.

[4] N. Doorn, T. E. Vos, and B. Marín, "Towards understanding students' sensemaking of test case design," *Data & Knowledge Engineering*, p. 102199, 2023.

[5] S. Deterding, D. Dixon, R. Khaled, and L. Nacke, "From game design elements to gamefulness: Defining" gamification"," in *15th international academic MindTrek conference: Envisioning future media environments*, 2011, pp. 9–15.

[6] D. Dicheva, C. Dichev, G. Agre, and G. Angelova, "Gamification in education: A systematic mapping study," *Journal of educational technology & society*, vol. 18, no. 3, pp. 75–88, 2015.

[7] S. de Sousa Borges, V. H. Durelli, H. M. Reis, and S. Isotani, "A systematic mapping on gamification applied to education," in *29th annual ACM symposium on applied computing*, 2014, pp. 216–222.

[8] I. Caponetto, J. Earp, and M. Ott, "Gamification and education: A literature review," in *European conference on games based learning*, Academic Conferences International Limited, 2014, p. 50.

[9] J. Vargas-Enríquez, L. García-Mundo, M. Genero, and M. Piattini, "Análisis de uso de la gamificación en la enseñanza de la informática," in *XXI jornadas de la enseñanza universitaria de la informática*, Univ. Oberta La Salle, 2015, pp. 105–112.

[10] B. Marín, "Lessons learned about gamification in software engineering education," in *Research anthology on developments in gamification and game-based learning*, IGI Global, 2022, pp. 1473–1496.

[11] B. Marín, "Gamification to ignite learning in modern times (keynote)," in *2nd international workshop on gamification in software development, verification, and validation (GAMIFY)*, 2023, pp. 1–1.

[12] M. R. Lepper, "Motivational considerations in the study of instruction," *Cognition and instruction*, vol. 5, no. 4, pp. 289–309, 1988.

[13] T. W. Malone and M. R. Lepper, "Making learning fun: A taxonomy of intrinsic motivations for learning," in *Aptitude, learning, and instruction*, Routledge, 2021, pp. 223–254.

[14] S. Bai, K. F. Hew, and B. Huang, “Does gamification improve student learning outcome? Evidence from a meta-analysis and synthesis of qualitative data in educational contexts,” *Educational Research Review*, vol. 30, p. 100322, 2020.

[15] R. Paul and L. Elder, The thinker’s guide to socratic questioning. Rowman & Littlefield, 2019.

[16] “TestSphere,” Ministry of Testing. Dec. 2022. Available: https://www.ministryoftesting.com/testsphere

[17] B. A. Kitchenham, D. Budgen, and O. P. Brereton, “Using mapping studies as the basis for further research–a participant-observer case study,” Information and Software Technology, vol. 53, no. 6, pp. 638–651, 2011.

[18] S. Blakeslee, “The CRAAP test,” Loex Quarterly, vol. 31, no. 3, p. 4, 2004.

[19] P. Rodrigues, M. Souza, and E. Figueiredo, “Games and gamification in software engineering education: A survey with educators,” in Frontiers in education conference (FIE), IEEE, 2018, pp. 1–9.

[20] M. Marabesi and I. F. Silveira, “Towards a gamified tool to improve unit test teaching,” in *2019 XIV latin american conference on learning technologies (LACLO)*, IEEE, 2019, pp. 12–19.

[21] M. Tsunoda and H. Yumoto, “Applying gamification and posing to software development,” in *25th asia-pacific software engineering conference (APSEC)*, IEEE, 2018, pp. 638–642.

[22] J. Severo and V. Lelli, “Testing maze: An educational game for teaching functional testing,” in *XXXVII brazilian symposium on software engineering*, in SBES ’23. Campo Grande, Brazil: ACM, 2023, pp. 407–415.

[23] H. M. dos Santos, V. H. S. Durelli, M. Souza, E. Figueiredo, L. T. da Silva, and R. S. Durelli, “CleanGame: Gamifying the identification of code smells,” in *XXXIII brazilian symposium on software engineering*, in SBES ’19. Salvador, Brazil: ACM, 2019, pp. 437–446.

[24] G. Fraser, A. Gambi, M. Kreis, and J. M. Rojas, “Gamifying a software testing course with code defenders,” in *50th ACM technical symposium on computer science education*, in SIGCSE ’19. Minneapolis, MN, USA: ACM, 2019, pp. 571–577

[25] G. Fraser, A. Gambi, and J. Rojas, “Teaching software testing with the code defenders testing game: Experiences and improvements,” in *International conference on software testing, verification and validation workshops (ICSTW)*, IEEE, 2020, pp. 461–464.

[26] R. A. DeMillo, R. J. Lipton, and F. G. Sayward, “Hints on test data selection: Help for the practicing programmer,” *Computer*, vol. 11, no. 4, pp. 34–41, 1978,

[27] T. Fulcini and L. Ardito, “Gamified exploratory GUI testing of web applications: A preliminary evaluation,” in *International conference on software testing, verification and validation workshops (ICSTW)*, IEEE, 2022, pp. 215–222.

[28] L. Furtado and S. R. B. Oliveira, “A teaching proposal for the software measurement process using gamification: An experimental study,” in *Frontiers in education conference (FIE)*, IEEE, 2020, pp. 1–8.

[29] D. Zurita-Gaibor, M. Escobar-Sanchez, and X. Lopez-Chico, “Application of ‘design thinking’ in the development of virtual platforms with gamified elements,” in *3rd international conference on information systems and software technologies (ICI2ST)*, IEEE, 2022, pp. 122–129.

[30] S. P. Chow, T. Komarlu, and P. T. Conrad, “Teaching testing with modern technology stacks in undergraduate software engineering courses,” in *26th annual conference on innovation and technology in computer science education*, in ITiCSE ’21. Virtual Event, Germany: ACM, 2021, pp. 241–247.

[31] L. N. Paschoal, L. F. Turci, T. U. Conte, and S. R. S. Souza, “Towards a conversational agent to support the software testing education,” in *XXXIII brazilian symposium on software engineering*, in SBES ’19. Salvador, Brazil: ACM, 2019, pp. 57–66.

[32] M. Maarek, L. McGregor, S. Louchart, and R. McMenemy, “How could serious games support secure programming? Designing a study replication and intervention,” in *European symposium on security and privacy workshops (EuroS&PW)*, IEEE, 2019, pp. 139–148.

This article will be included in the INTED2024 Proceedings (ISBN: 978-84-09-59215-9)
It will be fully citable as soon as it appears in IATED Digital Library (library.iated.org)
This version should not be distributed since it may change prior to final publication

[33] M. Gutfleisch, M. Schöps, S. Sayin, F. Wende, and M. A. Sasse, “Putting security on the table: The digitalisation of security tabletop games and its challenging aftertaste,” in *ACM/IEEE 44th international conference on software engineering: Software engineering education and training*, in ICSE-SEET ’22. Pittsburgh, Pennsylvania: ACM, 2022, pp. 217–222.

[34] E. T. S. Masson, A. T. S. Calazans, I. N. Bandeira, G. R. S. Silva, and E. D. Canedo, “Scrum in practice: City reconstruction as a pedagogical game challenge,” in *XXII brazilian symposium on software quality*, in SBQS ’23. Brasília, Brazil: ACM, 2023, pp. 321–331.

[35] S. Tonhão *et al.*, “Gamification in software engineering education: A tertiary study,” in *XXXVII brazilian symposium on software engineering*, in SBES ’23. Campo Grande, Brazil: ACM, 2023, pp. 358–367.

[36] I. Elgrably and S. Oliveira, “Remote teaching and learning of software testing using active methodologies in the COVID-19 pandemic context,” in *Frontiers in education conference (FIE)*, IEEE, 2021, pp. 1–9.

[37] T. Fulcini, R. Coppola, L. Ardito, and M. Torchiano, “A review on tools, mechanics, benefits, and challenges of gamified software testing,” *ACM Comput. Surv.*, vol. 55, no. 14s, Jul. 2023.

[38] C. Barreto and C. Franca, “Gamification in software engineering: A literature review,” in *IEEE/ACM 13th international workshop on cooperative and human aspects of software engineering (CHASE)*, IEEE, 2021, pp. 105–108.

[39] “TestSphere,” *Ministry of Testing*. Jan. 2024. Available: https://www.ministryoftesting.com/testsphere

[40] “Would Heu-Risk It?” Sep. 2023. Available: https://pejgan.se/wouldheu-riskit.html

[41] Chris Kenst, “The TestOpsy,” Chris Kenst, Jun. 2022, Available: https://www.kenst.com/the-testopsy

[42] “Black Box Puzzles.” Feb. 2022. Available: https://blackboxpuzzles.workroomprds.com

[43] O. Noroozi, H. Dehghanzadeh, and E. Talaee, “A systematic review on the impacts of game-based learning on argumentation skills,” Entertainment Computing, vol. 35, p. 100369, 2020.

[44] M. A. Kuhail, A. ElSayary, S. Farooq, and A. Alghamdi, “Exploring immersive learning experiences: A survey,” *Informatics*, vol. 9, no. 4, 2022.

[45] Å. Hirsh, C. Nilhom, H. Roman, E. Forsberg, and D. Sundberg, “Reviews of teaching methods – which fundamental issues are identified?” *Education Inquiry*, vol. 13, no. 1, pp. 1–20, 2022.

[46] B. S. Tan and K. S. Chong, “Unlocking the potential of game-based learning for soft skills development: A comprehensive review,” *Journal of ICT in Education*, vol. 10, no. 2, pp. 29–54, 2023.

[47] “Risk-storming.” Apr. 2020. Available: https://riskstorming.com

[48] B. Marín, T. E. J. Vos, M. Snoeck, A. C. Paiva, and A. R. Fasolino, “ENACTEST project-european innovation alliance for testing education,” CAiSE Research Projects Exhibition, pp. 91–96, 2023.

This article will be included in the INTED2024 Proceedings (ISBN: 978-84-09-59215-9)
It will be fully citable as soon as it appears in IATED Digital Library (library.iated.org)
This version should not be distributed since it may change prior to final publication