Critical Failure: A Card Game to Support the Learning of Non-Functional Testing

Exploring Software Quality Attributes through Competition and Engagement

Andre L. S. e Oliveira
Centro de Informática
Universidade Federal da Paraíba
João Pessoa, PB, Brazil
sousa.andrelopes@email.com

Matheus Nascimento
Centro de Informática
Universidade Federal da Paraíba
João Pessoa, PB, Brazil
edutheu_dev@hotmail.com

Antonio Filho
Centro de Informática
Universidade Federal da Paraíba
João Pessoa, PB, Brazil
antoniorochalimafilho@gmail.com

Ana Cecilia Nascimento
Centro de Informática
Universidade Federal da Paraíba
João Pessoa, PB, Brazil
acsdn@academico.ufpb.br

Erik Freire Ramos
Centro de Informática
Universidade Federal da Paraíba
João Pessoa, PB, Brazil
erik.freire.ramos@gmail.com

Christopher A. T. da Silva
Centro de Informática
Universidade Federal da Paraíba
João Pessoa, PB, Brazil
chris.atsilva@gmail.com

Yuska Paola Costa Aguiar
Centro de Informática
Universidade Federal da Paraíba
João Pessoa, PB, Brazil
yuska@ci.ufpb.br

ABSTRACT

Teaching non-functional testing and software quality attributes presents persistent challenges due to the abstract and theoretical nature of the content. To address this, we developed Critical Failure, a card game designed to support learning through playful competition and active engagement. Inspired by poker mechanics and adapted for educational use, the game encourages players to evaluate and critique each other's hands as software systems, simulating the evaluation of quality attributes such as performance, usability, and security. The game was created as part of a Software Testing course to offer a more interactive alternative to traditional lectures. The game was played by a group of 17 students who provided highly positive feedback on its clarity, engagement, and educational value. Suggestions for improvement were incorporated into the final version, refining both gameplay balance and alignment with course content. The activity demonstrated strong potential as a didactic tool, helping students not only recognize various non-functional requirements but also understand how they interact and impact the testing process. This article presents the design of the game and reports on its initial classroom use as an exploratory activity, intended to gather impressions, refine mechanics, and assess its potential as a complementary tool for teaching non-functional testing concepts.

KEYWORDS

non-functional testing, software quality attributes, active learning, Critical Failure, original game design

1 Introduction

Software testing plays a pivotal role in ensuring the quality and reliability of software systems [1]. Particularly, non-functional testing—which encompasses attributes such as performance, security, usability, maintainability, and scalability—is often underrepresented in undergraduate curricula, despite its increasing relevance in industrial contexts [11]. Unlike functional testing, which verifies whether the system meets specified requirements, non-functional testing focuses on how the system operates under various conditions and constraints, making it more abstract and challenging to teach [19].

Traditional pedagogical approaches to teaching non-functional testing often rely on lectures, theoretical definitions from textbooks, and isolated exercises [28]. However, these methods frequently fall short in fostering deep understanding or lasting engagement, especially when dealing with complex and subjective quality attributes. This issue is exacerbated by the lack of interactive tools that allow students to effectively simulate the impact of non-functional requirements on real software systems [8]. Moreover, many students struggle to connect theoretical models of software quality with practical decision-making scenarios encountered in software development [29].

To bridge this gap, educational research in computing has embraced active learning methodologies. These approaches emphasize student participation, creativity, and collaboration, transforming them from passive recipients of information into active agents in knowledge construction. Within this pedagogical movement, game-based learning has emerged as a promising strategy to foster motivation, critical thinking, and the ability to apply abstract concepts in realistic [5].

Educational games like GAMUT, which employs game-based learning for teaching unit testing [12], and experiences with continuous gamification in testing courses [23], highlight the potential of this approach. Previous studies have demonstrated that games can contribute to a deeper understanding of content and greater knowledge retention. Games such as Sojourner under Sabotage, for instance, have shown how playful environments enhance students'

comprehension of unit testing and debugging [24]. However, most of these efforts focus on functional testing, neglecting the important area of non-functional requirements.

Furthermore, the adoption of educational games offers additional benefits beyond content mastery. According to Blanco et al. [4] and systematic reviews like that of Tonhão et al. [26], the software industry increasingly demands interpersonal skills such as teamwork, adaptability, communication, and problem-solving. Previous experiences with educational games in undergraduate courses, as reported by Jesus et al. [15], reinforce that gamification can enhance both engagement and the assimilation of software testing concepts.

In this context, we developed Critical Failure, a card game designed to support the learning of non-functional testing through competitive, scenario-based gameplay. The game aims to simulate realistic interactions between quality attributes, encouraging players to evaluate and critique hypothetical software systems based on combinations of quality factors. By incorporating game mechanics inspired by poker and debate rounds, Critical Failure transforms abstract quality attributes into tangible and discussable elements.

The game was developed as part of a Software Testing course at the Federal University of Paraíba (UFPB), within the Center for Informatics. Software Testing is a mandatory 60-hour course offered in the seventh semester of the Bachelor of Computer Science program. This discipline builds directly on the foundations laid in the prerequisite Software Engineering course and covers both theoretical and practical aspects of test planning, design, execution, and evaluation. Topics include test policies, strategies for functional and non-functional test design, metrics and defect taxonomies, and tools for test automation and reporting. Embedding Critical Failure within this curriculum provided an opportunity to reinforce core syllabus elements—particularly those related to non-functional requirements—while offering students a hands-on, interactive means to contextualize and apply the knowledge acquired in lectures and laboratories.

The game's structure promotes engagement and learning by making players both stakeholders and evaluators. In each round, players are presented with a "system," represented by quality attribute cards, and must collectively determine the configuration that offers the most balanced or suitable quality profile for a specific context. This process stimulates not only the memorization of theoretical content but also skills in negotiation, justification, and strategic thinking—valuable competencies in software development.

The game was applied with a group of 17 undergraduate students in a classroom environment. Participants engaged in several rounds of gameplay, followed by structured feedback activities. Students reported high levels of satisfaction and acknowledged the game's usefulness in deepening their understanding of non-functional concepts. The collected feedback, obtained through an anonymous questionnaire, was used to enhance both the game mechanics and content alignment, resulting in a more balanced and pedagogically effective tool.

This article describes the design rationale, gameplay structure, and initial application of Critical Failure as a complementary educational resource. Our goal is to contribute to the growing body of research on playful strategies in computing education, specifically addressing the often-neglected area of non-functional testing. Through this work, we advocate for the adoption of more engaging, practical, and student-centered approaches to teaching software quality—preparing students not only to know what to test but also how to test the qualities that matter most in real-world systems.

2 Related Work

The integration of game-based learning into software engineering education has gained traction as an effective strategy to enhance student engagement and comprehension, particularly in the domain of software testing. Several educational games have been developed to address various aspects of software testing, each employing unique mechanics and pedagogical approaches.

One notable example is Debug [6], a card game designed to teach levels and types of software testing. Developed within an undergraduate course, the game encourages collaborative learning by having students create and play games that explore testing concepts. Feedback from participants indicated increased engagement and a deeper understanding of the subject matter .

Similarly, GAMUT (GAMe-based learning approach for teaching Unit Testing) [12] introduces a narrative-driven experience where students engage in unit testing through gameplay. The game employs a medieval fantasy theme to contextualize testing scenarios, thereby making abstract concepts more tangible. Evaluations have shown that GAMUT effectively enhances student motivation and facilitates the learning of unit testing principles .

GreaTest [2] is another card game aimed at motivating students to learn software testing. In this game, players assume the role of test analysts, determining appropriate test types for various scenarios presented on the cards. The interactive nature of GreaTest fosters critical thinking and decision-making skills pertinent to software testing practices .

Expanding beyond card games, Sojourner under Sabotage [24] is a browser-based serious game that immerses players in a narrative where they must identify and fix sabotaged components of a spaceship using unit testing and debugging techniques. The game provides hands-on experience with real-world testing frameworks, such as JUnit [16], thereby bridging the gap between theoretical knowledge and practical application .

In the realm of software ergonomics, JADE [14] is a board game designed to teach principles of software ergonomics to beginners. The game has been successfully implemented in educational settings, offering students an interactive platform to learn and apply ergonomic concepts in software design.

While these educational games have demonstrated effectiveness in teaching various facets of software testing and design, they predominantly focus on functional testing aspects. There remains a relative paucity of game-based learning tools specifically targeting non-functional testing and quality attributes. The development of Critical Failure seeks to address this gap by providing an interactive medium through which students can explore and understand non-functional testing concepts, thereby enriching the educational landscape of software engineering.

Table 1 presents a comparative summary of the educational games discussed. It highlights key aspects of each proposal, such as the type of game, the interaction dynamics among participants,

and the scope of the content addressed. This allows for a clear visualization of the similarities and distinctions in relation to Critical Failure.

Table 1: Overview of Related Educational Games in Software Testing

Name	Type	Mode	Scope
Debug [6]	Analog (Card game)	Cooperative	Levels and types of software testing
GAMUT [12]	Digital (web-based narrative)	Individual	Unit testing
GreaTest [2]	Analog (Card game)	Competitive	Functional testing in general
Sojourner under Sabotage [24]	Digital web-based serious game	Individual	Unit testing and debugging
JADE [14]	Analog (Board game)	Cooperative	Software ergonomics concepts
Critical Failure	Analog (Card game)	Competitive	Non-functional testing topics and quality attributes

3 Methodology

The methodological approach adopted in this study was structured around two interrelated levels: the instructional methodology of the Software Testing course and the specific development methodology applied by the team responsible for designing Critical Failure. This dual framework reflects a dual pedagogical purpose—where both the designing and playing of the game served as learning activities within the course context—and a triple objective for the students involved: (i) to learn the course content, (ii) to propose an original educational game, and (iii) to implement and refine the proposed game based on peer interaction and feedback.

3.1 Course-Level Methodology

The methodology used in the course was designed to promote active student engagement through playful and creative activities, encouraging the assimilation of theoretical knowledge and the construction of original educational materials. Within this context, an activity was proposed in which students would develop a card game focused on the domain of non-functional testing.

Before engaging in the practical aspects of the assignment, students received a theoretical introduction to the topic, supported by access to curated bibliographic materials, reference documents, and lecture notes. The activity was conducted in person, with students organized into teams of approximately six members. It unfolded over three sessions, each lasting 120 minutes, and was structured as follows: 1- Idea exploration and concept definition. 2- Creation of the necessary artifacts for the game. 3- Game implementation and testing within the class, followed by the submission of supporting documentation.

The central theme of the assignment was Quality Attributes and Non-Functional Testing, with the goal of deepening students' understanding of these concepts within the broader field of Software Engineering. Students were also encouraged to incorporate related topics already covered in the course—such as foundational testing principles, levels of testing, and testing types—thereby fostering an integrated and contextualized learning experience.

Each team was tasked with designing an educational game that explicitly addressed both quality attributes—Performance Efficiency, Compatibility, Reliability, Security, Portability, and Usability—and the corresponding non-functional testing types of choice.

A detailed definition of each of these components was required as part of the game documentation, with the intent of consolidating students' theoretical knowledge through practical application. The assignment emphasized interdisciplinarity, critical thinking, and students' ability to synthesize and relate abstract concepts to applied scenarios.

In Lesson 1, students received specific instructions on how to structure their game design. The documentation was expected to clearly specify the following elements: (i) the name of the game; (ii) the scope and covered topics; (iii) details about the cards, including quantity, content, and dimensions; (iv) the minimum and maximum number of players; (v) the type of game (e.g., collaborative, competitive, strategic); (vi) a comprehensive description of the game dynamics, including initial card distribution, game rules, win conditions, and scoring system (if applicable). In cases where the game was inspired by existing titles, the original game and any modifications made were to be documented as well.

During the semester, a total of eight educational games were created by different student teams, each addressing various aspects of non-functional testing. Notably, some of the authors of this paper are undergraduate students who participated in the development of Critical Failure. This game was selected as the focus of the study due to the authors' direct involvement in its creation, which provided in-depth familiarity with its conceptual foundations, design rationale, and iterative refinement process. This insider perspective enabled a more comprehensive and accurate account of both the pedagogical methodology and the development experience.

3.2 Game Development Methodology

Within the framework of the course activity, the team responsible for Critical Failure—composed of six undergraduate students enrolled in the Bachelor's degree program in Computer Science and taking the Software Testing course—adopted a specific internal methodology to guide the creation and refinement of the game. This process began during Lesson 1 with a structured brainstorming session aimed at generating and evaluating potential game concepts. Several initial ideas were discussed and informally tested to assess their viability and educational relevance. From this process, a promising concept was selected, and a preliminary ruleset was developed and refined outside class time. This version served as the foundation for the physical prototype.

To develop the game cards, Dextrous [7] was used for modeling, with the support of artificial intelligences such as Gemini [13] and ChatGPT [18] for creating the images used to illustrate the cards, which were produced outside regular class hours. During Lesson 2, time was allocated to the manual construction of the cards using basic materials such as scissors, cardstock, and glue.

Initial playtesting rounds were conducted with classmates to evaluate the gameplay experience and identify areas for improvement. The feedback obtained during these sessions was crucial to refining the balance, engagement, and educational coherence of the game mechanics.

In Lesson 3, the game was formally implemented in the classroom, where the team observed others playing the game and recorded notes regarding player behavior, engagement, and rule clarity. In addition, participants completed a standardized feedback form evaluating the experience. This data, alongside team observations, was used during the remaining class time to adjust the game's final version.

The final submission included the game's description along with the game materials themselves, such as the instruction manual, the complete set of cards in high quality and in a print-ready format, and a credits section listing the names and student IDs of all team members. All files were required to be submitted in PDF format via the SIGAA platform [27]. A post-activity anonymous questionnaire was also administered to all students, prompting them to reflect on their experience playing the developed games.

Figure 1 presents a flowchart that summarizes the design and development process of Critical Failure, outlining the main activities carried out in each of the four sessions.

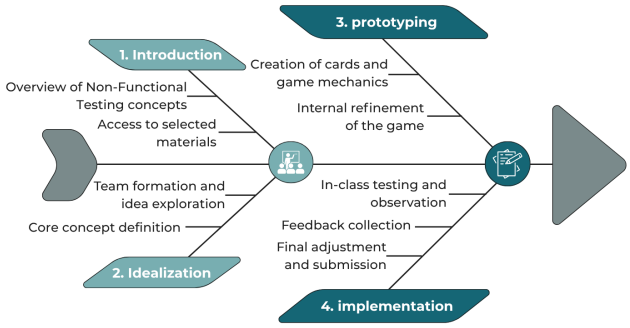

Figure 1: Critical Failure development steps flow

4 Implementation

4.1 The idea and characterization

Non-Functional Testing plays a role in evaluating characteristics of systems and software that, although not directly related to functionality, have a direct impact on the end user's experience. In other words, these tests analyze how the system behaves under different usage conditions.

These application characteristics are known as software quality attributes. In the game Critical Failure, the following quality attributes are addressed: Performance Efficiency, Compatibility, Reliability, Security, Portability, and Usability. In addition to the characteristics, each attribute in the game is associated with two or three types of non-functional tests. The rationale for selecting these specific attributes is discussed in detail in subsection 4.2.

In Critical Failure, players will act both as "developers," building their own systems, and as "testers," trying to find flaws in other developers' systems. The only way to defend against a flaw is if the "developer" has in their system a non-functional test corresponding to the quality attribute being tested by the "tester." Thus, when building their system, each player should prioritize including a variety of characteristics, aiming to make it as resistant as possible to failures. However, just like in a real system, having all quality attributes does not make it immune to flaws, which is why the game includes a special card to test and uncover flaws in systems that appear to be "invincible."

To reinforce the concepts of non-functional tests and quality attributes, the game also includes a bluffing mechanic inspired by poker. When being tested by another player, if the "developer" does not have in their system a non-functional test related to the tested quality attribute, they can choose one of their system's tests and read its description pretending it belongs to the tested attribute. The "tester" must then identify which quality attribute that description actually belongs to before accusing the "developer" of bluffing.

The game is primarily intended for undergraduate computing students who are beginning to learn about software testing and wish to develop a deeper understanding of non-functional aspects of system quality. It addresses the common learning challenge posed by the abstract and theoretical nature of non-functional testing concepts, which often makes them difficult to grasp through traditional instruction alone [29]. Critical Failure is designed to provide an engaging and experiential means of exploring these ideas, remaining enjoyable even for players with no prior knowledge of the subject. By the end of the game, it is expected that players will comprehend the importance of non-functional testing in software development, recognize the definitions of various test types, and identify the quality attributes associated with each. Table 2 provides a summary of the main features of the Critical Failure game.

Table 2: Critical Failure Game Overview

Game name: Critical Failure
Game Type: Competitive **Scope**: Non-Functional Testing; Software Quality Attributes **Number of Cards**: 54 cards **Number of Players**: 3 to 8 players **Cards Dimensions**: 63.5mm x 97.89mm

4.2 Cards of the game

The cards used in the game were selected based on the quality attributes defined in the ISO/IEC 25010 standard [21]. However, given the extensive number of quality attributes and corresponding test types described in the standard, including all of them would have resulted in an excessive number of cards and an overly complex gameplay experience.

Therefore, only a subset of attributes and associated tests was chosen for inclusion. The selection prioritized quality attributes that could be more easily represented and understood by beginners, while the total number of cards was determined through internal hypotheses and playtesting sessions conducted by the developers to ensure a balanced, engaging, and time-efficient game experience.

Below is a description of the cards in Critical Failure, including the tests associated with each quality attribute and the definitions of each test. Figure 2 displays the visual design of the cards, while the quantity of each card type is presented later in Table 3.

Figure 2: Cards of the game Critical Failure

Performance Efficiency Attribute - Test: Load Testing. Description: Simulates normal and peak usage of the system to ensure it performs well under the expected number of users. Test: Stress Testing. Description: Pushes the system to its limits to see if it can withstand and recover from maximum pressure. Test: Scalability Testing. Description: Checks if the system can handle a large increase in users or workload without slowing down or crashing.

Compatibility Attribute - Test: Coexistence Testing. Description: Verifies whether a new system works well with existing systems without causing issues or disrupting their functioning. Test: Interoperability Testing. Description: Checks whether different systems can work together and exchange information correctly.

Reliability Attribute - Test: Recovery Testing. Description: Verifies whether the system can return to proper functioning and preserve data after failures (such as power outages or serious errors). Test: Fault Tolerance Testing. Description: Checks if the system can continue to function even when partial failures occur in components or services. Test: Availability Testing. Description: Verifies whether the system remains accessible and operating as expected during its intended period of operation.

Security Attribute - Test: Confidentiality Testing. Description: Ensures that only authorized individuals can access sensitive information in the system. Test: Authentication Testing. Description: Checks whether the system correctly verifies a user's identity before granting access. Test: Non-Repudiation Testing. Description: Ensures the system can prove that a specific action was performed by a specific user, preventing denial of the action later.

Portability Attribute - Test: Adaptability Testing. Description: Verifies whether the software performs well in different environments (systems, browsers, etc.) without major changes. Test: Installability Testing. Description: Ensures the software installs easily and without errors in different environments, ensuring a good user experience. Test: Replaceability Testing. Description: Verifies whether a software component can be replaced by an equivalent one without compromising system operation.

Usability Attribute - Test: A/B Testing. Description: Compares two versions of something (like a web page or feature) to determine which one performs better with real users. Test: Accessibility Testing. Description: Ensures the system is accessible to users with different types of disabilities, such as visual, auditory, or motor impairments. Test: User Error Protection Testing. Description: Verifies whether the system has mechanisms to prevent or minimize user errors and their consequences.

Special Cards - Critical Failure: Based on the principle that no system can ever be entirely immune to defects, the Critical Failure special card introduces an unavoidable failure condition. Mechanically, this card cannot be used to build a player's system. Instead, when used to test another player's system, it automatically causes a failure — allowing the tester (attacker) to take one card from the tested system and add it to their own, without any possibility of defense by the opponent.

Table 3: Card Quantity and Types

QTY	Type of Attribute	Type of Test	QTY
9	Performance Efficiency	Load Stress Scalability	3 3 3
6	Compatibility	Coexistence Interoperability	3 3
9	Reliability	Recovery Fault Tolerance ScalabilityAvailability	3 3 3
9	Security	Confidentiality Authentication Non-Repudiation	3 3 3
9	Portability	Adaptability Installability Replaceability	3 3 3
9	Usability	A/B Stress Scalability	3 3 3
3	Special Card: Critical Failure		
54	**TOTAL OF CARDS**		

4.3 The rules of the game

"Critical Failure" is designed as a competitive card game where players use cards representing different non-functional tests and quality attributes. The objective is to build a robust "system" while

identifying flaws in opponents' systems, accumulating points based on collected cards and formed combinations. The game rules are as follows:

Setup

(1) *Card Distribution:* Each player draws an initial hand of 6 cards from the deck.
(2) *Order of Play:* The order of players is defined (e.g., clockwise).
(3) *System Assembly:* Each player selects 3 of the 6 cards from their hand to form their "defense system." These 3 cards are placed face down on the table in front of the player. The remaining 3 cards stay in the player's hand to be used in the testing phase.
(4) *Testing Phase:* With the systems assembled, players use the remaining 3 cards in their hands to test opponents' systems. On their turn, a player performs one test action. After the action, the turn passes to the next player. This phase continues until all players have used all the cards from their hands (the 3 test cards).

Performing a Test - On their turn, the active player chooses a test card from their hand and selects an opponent's system to test. The outcome of the test depends on the card(s) in the opponent's system (which are face down): *Flaw Discovered:* If the opponent does not have a card in their system with the same quality attribute (same color) as the test card used, the attacking player "discovers a flaw." The attacker then takes one of the cards from the defender's system (their choice from the face-down system cards) and adds it to their own system. *Successful Defense:* If the opponent has a card in their system with the same quality attribute (same color) as the test card used, the defender's system withstands the test. The defender then adds the test card used by the attacker to their own system.

Bluff and Challenge - *Defender's Declaration (Bluff):* When being tested, and before any system card is revealed, the defending player can declare whether or not they have a card with the same quality attribute (same color) as the test card used by the attacker. If they claim to have such a card, they might be telling the truth or bluffing. *Attacker's Doubt (Challenge):* The attacking player can doubt the defender's declaration. If they doubt, the defender must provide a definition of a non-functional test type (associated with the card they claim to have) to convince the attacker they possess the card. *Challenge Resolution:* After hearing the description/definition, the attacking player decides whether they believe the (still unrevealed) card corresponds to the quality attribute needed for defense. *Attacker Accepts Description:* If the attacker accepts that the described (still unrevealed) card has the correct quality attribute, the standard rule from *Performing a Test* applies (Successful Defense or Flaw Discovered, depending on the truthfulness of the defender's claim about the card they actually have for that attribute). The play proceeds based on the attacker's trust in the defender's word. *Attacker Does Not Accept Description (Contests): Revelation:* The defending player then reveals the relevant card from their system (the card they would use for defense). Consequences of Revelation: *Defender Was Correct:* If the revealed card is of the quality attribute the defender initially declared (and which is the same as the attack card), the defense is successful (*Successful Defense:* the defender adds the attack card to their system). Additionally, as a penalty for the incorrect challenge, the defender can take one card (of their choice, without seeing it) from the attacking player's system. *Attacker Was Correct in Contesting (Defender Bluffed or Erred):* If the revealed card is not of the quality attribute the defender claimed (i.e., it does not defend against the attack), the standard rule from *Performing a Test* for Flaw Discovered applies (the attacker takes one card from the defender's system). Additionally, as a reward for the correct challenge, the attacker can take an additional card from the defender's system.

End of Round and Scoring - A round ends when all players have used all their test cards (the initial 3 cards from their hand). Players then reveal all cards in their respective systems and calculate their individual scores based on the following table: Each card in the system at the end of the round: 1 point. Pair of cards of the same color (same quality attribute) and different types (different non-functional tests): +2 bonus points (in addition to points for each individual card). Three cards of different colors (different quality attributes) and different types (different non-functional tests): +2 bonus points (in addition to points for each individual card). It is possible to score for multiple pairs of the same color(quality attributes), as long as each pair consists of cards of different types (tests). The player with the highest total score wins the round.

Table 4 summarizes the scoring methods and the points awarded for each combination, while Figure 3 illustrates the overall gameplay flow of the game. The diagram outlines the turn-based sequence starting from card distribution and system assembly, followed by testing actions where players attempt to identify flaws in opponents' systems. In the flowchart, green ellipses represent start and end points, blue rectangles indicate procedural steps, and pink diamonds denote decision points throughout the gameplay.

Table 4: Scoring Table

Scoring method	Points
Each individual cart	+1
Pair of cards of the same quality attribute and different non-functional tests	+2
Three cards of different quality attributes and different non-functional tests	+2

4.4 Application of the Game in the Class

The in-class game session, dedicated to the practical implementation and experimentation with the developed educational games, was conducted during the third lesson of the activity. The classroom was organized into multiple groups of approximately six students, arranged in a circular layout to facilitate rotation between the various games created by the teams. This structure allowed each group to engage with different games throughout the session. Among the games presented, Critical Failure was made available for play and evaluation by the participants.

Each group that interacted with Critical Failure participated in two consecutive rounds, resulting in a total of six complete playthroughs over the course of the session.

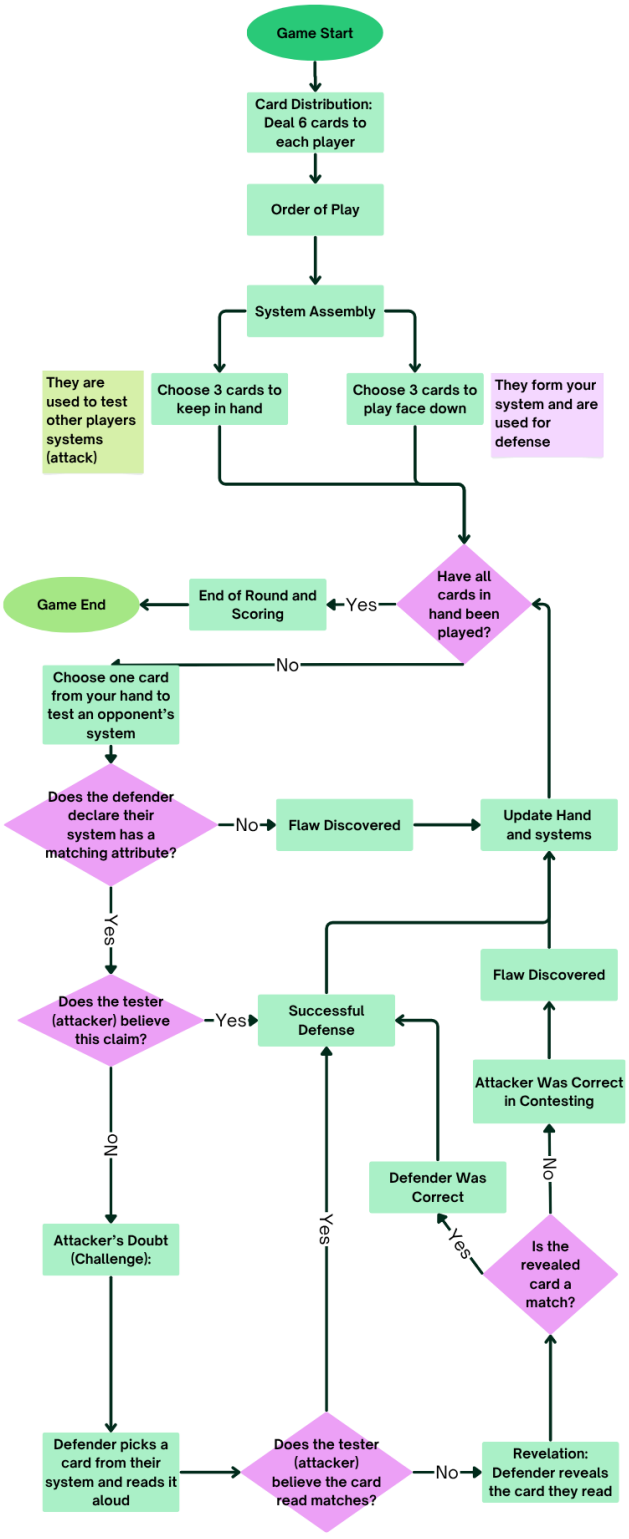

Figure 3: Gameplay Flow of Critical Failure

Upon arrival at the game station, each new group received a brief explanation of the game rules provided by a member of the development team, ensuring that players were adequately prepared to begin the activity without significant delays.

Throughout each match, a designated team member acted as a supervisor, observing the gameplay process, clarifying any doubts raised by the players, and taking detailed notes on the participants' reactions, strategies, and overall engagement.

At the conclusion of each session, players were asked a series of informal questions to gather immediate feedback about their experience with the game. This informal feedback was notably positive. One of the most frequently praised aspects of Critical Failure was the bluffing mechanic, which was highlighted as a key element that enhanced player engagement and made the experience more dynamic and enjoyable.

During gameplay sessions, several qualitative observations were made. One such finding was that the rules appeared somewhat confusing for first-time players, particularly due to the distinction between face-down system cards and the cards in a player's hand. Although some confusion was observed, this mechanic was retained in the final version of the game, as it was deemed essential to maintain the game's metaphor of constructing a software system. Nonetheless, a proposed improvement for future iterations would be to provide a printed play area to clearly separate system and test cards, alongside a reference guide summarizing the scoring rules—another area that generated confusion during play.

Over the course of six rounds played during the classroom activity, four were won by players with significantly higher scores compared to others. This suggests a tendency toward unbalanced matches, where players in the lead gain a compounding advantage. Although the game's design assumed that players would target the strongest opponents—those with the most system cards—to prevent runaway leaders, empirical observation showed that players often attacked weaker opponents, reasoning that they were less likely to successfully defend against tests. This, in turn, allowed dominant players to proceed with minimal opposition. A potential direction for future research is to investigate whether this behavior is a characteristic of inexperienced players or an inherent flaw in the game dynamics. Possible balancing solutions include increasing the number of Critical Failure cards or implementing mechanics that penalize players with larger systems—for instance, disallowing test cards used against the leading player from being added to their system even if the test is passed.

5 Lessons Learned and Limitations

5.1 Game Experience

This section presents reflections gathered from both players and creators of the Critical Failure game, based on spontaneous and anonymous responses provided by students through an online questionnaire. The survey was designed as an optional activity to collect insights about the classroom use of educational games. A total of 34 students responded, among whom 17 had played Critical Failure—representing approximately 94.4% of the 18 students who engaged with the game. The following analysis outlines key lessons learned and highlights limitations identified throughout the experience.

Figure 4 presents the results of a multiple-choice question answered by 17 students regarding their experience with the game Critical Failure. Among them, 14 indicated that they enjoyed playing the game, while 2 reported feelings of stress during gameplay. Interestingly, both of the students who mentioned stress also selected that they had fun, suggesting that the competitive nature of the game may evoke both positive engagement and moments of frustration, particularly for players who are falling behind.

Additionally, 9 students reported that the game served as a review of previously learned content. This indicates a moderate degree of effectiveness in reinforcing knowledge of non-functional testing, especially during the bluffing phase, in which players must judge whether their peers' statements about quality attributes are accurate. This mechanic requires players to engage with and recall conceptual definitions actively. Moreover, 5 students stated that they learned new concepts, while 2 acknowledged that they needed to study the topic further. These responses suggest that the game may serve as a practical diagnostic tool, helping students to assess their own understanding and highlighting areas where further study is needed. In this way, Critical Failure has the potential to reinforce theoretical knowledge through experiential learning.

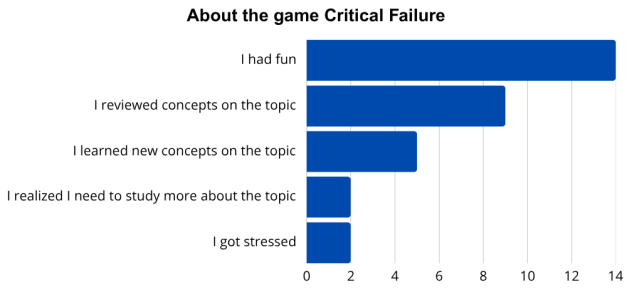

Figure 4: Graph depicting the opinions of students who engaged with the game.

5.2 Creation Process and Pedagogical Limitations

Shifting focus from the gameplay experience to the game creation process, the anonymous questionnaire also inquired about students' experiences in developing Critical Failure. Table 5 summarizes the responses of those involved. Overall, participants reported that the design process significantly deepened their understanding of non-functional testing and software quality attributes. These observations align with the generation effect, originally demonstrated by Slamecka and Graf [22], which posits that actively generating content enhances semantic encoding and retention. Subsequent meta-analytic work [3] and more recent studies in educational settings [17] further support the efficacy of constructivist artifact creation in promoting meaningful learning of technical concepts.

Additionally, analysis of Table 5 responses suggests that the game development activity may foster the acquisition of interpersonal competencies—such as teamwork, communication, and

Table 5: The Experiences of the Students Who Developed the Game

Reflections on the Development of the Game:
"I had to think of ways to connect the concepts with the game rules, and I believe that really helped make the content stick in a broader way. "
"We wanted to make something fun to play no matter how much the player knows about the topic, and when we tested the game ourselves, we actually had fun playing it."
"It was a good experience. When we were discussing what game to make, our ideas really complemented each other, so the planning part went pretty quickly."
"The experience of creating the game helped me learn a lot about non-functional testing, both in theory and in practice, since we had to prototype and test the game several times before reaching the final version."
"I discovered new software for developing card games and also studied different types of quality attributes while I was working on the description for each card, focusing on examples of each test "
"By creating the game, I developed not only skills related to the testing course but also improved social interaction with my friends — it was an amazing experience to have. "
"I used my creativity and naturally saw the need for testing during the development process."

critical thinking—alongside domain-specific testing skills. Although the limited sample size precludes definitive conclusions, this trend gives rise to the hypothesis that engaging students in the design of pedagogical games can simultaneously cultivate both technical and soft skills.

Nevertheless, while these findings are promising, it is essential to consider potential limitations inherent to the game creation approach. The time and effort required to design, prototype, and iterate a game may be disproportionate to the depth of technical content covered, particularly when compared to more traditional instructional formats. Students may become overly focused on the creative or aesthetic aspects of the game at the expense of engaging deeply with the underlying theoretical concepts. Moreover, not all students possess the same levels of creativity, design skills, or comfort with open-ended assignments, which may lead to uneven participation within teams and create frustration for individuals less inclined toward artistic or conceptual work. This can hinder both learning outcomes and group cohesion.

Another challenge is the potential for cognitive overload— a state in which the mental demands of a task exceed the learner's working memory capacity, impairing comprehension and retention [25]. Combining multiple goals—content assimilation, teamwork, design thinking, and game balancing—within a single assignment may overwhelm some students, especially if they lack prior experience with educational game design or the targeted testing concepts.

Additionally, assessment of learning through the final product may be less precise, as a well-designed game does not necessarily guarantee accurate or comprehensive representation of the intended educational content. Without supplementary evaluative

tools, it can be difficult for instructors to isolate what students truly learned from what was creatively constructed. Finally, the reliance on external tools (e.g., Image generators, card design software) may introduce variability in access and technical proficiency, inadvertently privileging students with prior exposure to such resources.

5.3 Connection Between Game Design and Non-Functional Testing

Despite these challenges, a closer examination of observational notes and qualitative feedback reveals striking parallels between game design tasks and established non-functional testing practices. For instance, just as software must be tested to ensure correctness, a game must be tested for balance and fairness, including identifying overpowered combinations and evaluating the clarity and usability of the rules. Some of these similarities directly parallel concepts found in non-functional testing.

In terms of usability testing, game designers must assess whether the rules are understandable, whether the turn sequence is intuitive, and whether the actions available to players are clearly conveyed—all of which closely resemble evaluations of user experience in software systems. Regarding robustness, games must be able to handle unforeseen scenarios such as repetitive ties, dominant strategies, or players deviating from expected behavior, reflecting the importance of verifying how software responds to adverse conditions or invalid input. Lastly, accessibility testing is mirrored in the need for card games to be approachable by players with different levels of experience, age groups, or cognitive abilities, analogous to the challenge of ensuring software accessibility for diverse user profiles.

Future research could further investigate whether these parallels have significant educational implications, potentially establishing a theoretical foundation for using game design as a pedagogical tool in software engineering courses.

6 Conclusions and Future Work

Based on the evidence collected, we conclude that Critical Failure successfully fulfilled its purpose of facilitating the learning of non-functional testing and software quality attributes for the students who played it. Furthermore, the process of designing educational games proved to be an effective pedagogical strategy for those who participated in the game creation activity, reinforcing theoretical knowledge while also fostering the development of technical competencies and soft skills. These findings support the continued use of Critical Failure in educational contexts and suggest that the broader methodology—incorporating game design into curricula—holds promise as a complementary strategy to traditional instruction in software testing.

Nevertheless, while the results observed among the participants are encouraging, the limited number of students involved in both the design and evaluation phases prevents any generalizable conclusions from being drawn. Although it is reasonable to assert that the game achieved its intended educational impact within this specific cohort, such claims remain speculative in the absence of broader empirical validation. Thus, the observed outcomes should be interpreted more as indicative hypotheses than definitive findings. The promise shown by both the game and the underlying pedagogical approach highlights the need for further research with larger and more diverse student populations, allowing for more rigorous analysis of learning outcomes, methodological effectiveness, and scalability across different educational contexts.

Building upon these considerations, future work could expand upon this study by replicating the activity in different educational contexts to evaluate the generalizability of its results and comparing its outcomes with traditional teaching methods, such as lectures or laboratory activities. Additionally, the adoption of standardized evaluation instruments for educational games, such as the MEEGA+ model [20], could be employed to assess the quality, engagement, and learning impact of Critical Failure in a more structured and measurable way. Such analyses would contribute to strengthening the empirical basis for integrating playful methodologies into computing education.

ARTIFACT AVAILABILITY

The main artifact developed for this study—the educational card game Critical Failure, including its full deck of cards—is publicly available and can be freely accessed and reused via the following links: English version [10]; Brazilian Portuguese version [9]. This artifact is shared to support transparency, reproducibility, and reuse in alignment with Open Science principles.

However, due to ethical considerations related to participant privacy and the presence of sensitive information, the dataset derived from the user feedback forms cannot be made publicly available. These data include potentially identifiable responses and personal opinions collected during the evaluation phase, and no consent was obtained from participants for public disclosure of individual-level data. Therefore, in accordance with privacy protection standards, these data will not be publicly available.

REFERENCES

[1] S. S. Riaz Ahamed. 2010. Studying the Feasibility and Importance of Software Testing: An Analysis. arXiv:1001.4193 [cs.SE] https://arxiv.org/abs/1001.4193

[2] Thiago Beppe, Italo Linhares de Araújo, Bruno Aragão, Ismayle De Sousa Santos, Davi Ximenes, and Rossana Andrade. 2018. GreaTest: a card game to motivate the software testing learning. *SBES '18: Proceedings of the XXXII Brazilian Symposium on Software Engineering*, 298–307. doi:10.1145/3266237.3266254

[3] Sharon Bertsch, Bryan Pesta, Richard Wiscott, and Michael Mcdaniel. 2007. The generation effect: A meta-analytic review. *Memory & cognition* 35 (04 2007), 201–10. doi:10.3758/BF03193441

[4] Raquel Blanco, Manuel Trinidad, María José Suárez-Cabal, Alejandro Calderón, Mercedes Ruiz, and Javier Tuya. 2023. Can gamification help in software testing education? Findings from an empirical study. *Journal of Systems and Software* 200 (2023), 111647. doi:10.1016/j.jss.2023.111647

[5] Felipe Carvalho and Mariano Pimentel. 2020. *Atividades autorais online: aprendendo com criatividade.* http://horizontes.sbc.org.br/index.php/2020/11/atividadesautorais SBC Horizontes.

[6] Carolina de Melo Costa, Yuska Paola Costa Aguiar, Luiz Gusttavo Oliveira de Souza, Jayanne Laysa Cruz Morais, and Pedro Alves da Silveira. 2024. Card Game as an Educational Strategy in Software Testing: Debug a game to explore content on Levels and Types of Testing.. In *Proceedings of the XXIII Brazilian Symposium on Software Quality (SBQS '24)*. Association for Computing Machinery, New York, NY, USA, 605–612. doi:10.1145/3701625.3701685

[7] Dextrous Creative. 2025. Dextrous Creative. https://www.dextrous.com.au/ Accessed on May 15, 2025.

[8] Isaac Elgrably and Sandro Oliveira. 2023. Uma Abordagem para o Ensino de Testes de Software utilizando Metodologias Ativas em Cursos Superiores de Computação. *Anais Estendidos do Congresso Brasileiro de Informática na Educação (CBIE)*, 13–25. doi:10.5753/cbie_estendido.2023.234630

[9] Oliveira et al. 2025. Critical Failure Card Game – Brazilian Portuguese Version. https://drive.google.com/file/d/1wjkkAf6Xjxu9SfTboM5gWrvmXymllVhh/view?usp=sharing Accessed on October 12, 2025.

[10] Oliveira et al. 2025. Critical Failure Card Game – English Version. https://drive.google.com/file/d/1b9PMKSy-uyGCWDf6blTTLbW_X4p5SRuo/view?usp=sharing Accessed on October 12, 2025.
[11] Vahid Garousi, Austen Rainer, Per Lauvås, and Andrea Arcuri. 2020. Software-testing education: A systematic literature mapping. *Journal of Systems and Software* 165 (2020), 110570. doi:10.1016/j.jss.2020.110570
[12] Renata Faria Gomes and Valéria Lelli. 2021. GAMUT: GAMe-based learning approach for teaching Unit Testing. In *Proceedings of the XX Brazilian Symposium on Software Quality* (Virtual Event, Brazil) *(SBQS '21)*. Association for Computing Machinery, New York, NY, USA, Article 27, 11 pages. doi:10.1145/3493244.3493263
[13] Google. 2025. Google Gemini. https://gemini.google.com/app?hl=pt-BR Accessed on May 15, 2025.
[14] Stéphanie Jean-Daubias. 2023. JADE: a board game to teach software ergonomics. *Interaction Design and Architecture(s) Journal* 56 (March 2023), 29–52. doi:10.55612/s-5002-056-002
[15] Gabriela Martins de Jesus, Fabiano Cutigi Ferrari, Leo Natan Paschoal, Simone do Rocio Senger de Souza, Daniel de Paula Porto, and Vinicius Humberto Serapilha Durelli. 2020. Is It Worth Using Gamification on Software Testing Education? An Extended Experience Report in the Context of Undergraduate Students. *Journal of Software Engineering Research and Development* 8 (Aug. 2020), 6:1 – 6:19. doi:10.5753/jserd.2020.738
[16] JUnit Team. 2025. JUnit 5. https://junit.org/junit5/ Accessed on May 15, 2025.
[17] Yasmin B. Kafai and Quinn Burke. 2015. Constructionist Gaming: Understanding the Benefits of Making Games for Learning. *Educational Psychologist* 50, 4 (2015), 313–334. doi:10.1080/00461520.2015.1124022
[18] OpenAI. 2025. ChatGPT. https://chatgpt.com/ Accessed on May 15, 2025.
[19] Leo Natan Paschoal and Simone do Rocio Senger de Souza. 2018. A Survey on Software Testing Education in Brazil. In *Proceedings of the XVII Brazilian Symposium on Software Quality* (Curitiba, Brazil) *(SBQS '18)*. Association for Computing Machinery, New York, NY, USA, 334–343. doi:10.1145/3275245.3275289
[20] Giani Petri, Christiane Gresse von Wangenheim, and Adriano Borgatto. 2017. A Large-Scale Evaluation of a Model for the Evaluation of Games for Teaching Software Engineering. doi:10.1109/ICSE-SEET.2017.11
[21] E. Sheppard. 2025. ISO/IEC 25010:2023 Systems and Software Engineering — Systems and Software Quality Requirements and Evaluation (SQuaRE) — Product Quality Model Characteristics, Sub-characteristics, Definitions, and Their Proposed Application to Electric Vehicle Supply Equipment (EVSE) Software. doi:10.5281/zenodo.14758339 Accessed on October 12, 2025.
[22] Norman Slamecka and Peter Graf. 1978. The Generation Effect: Delineation of a Phenomenon. *Journal of Experimental Psychology: Human Learning and Memory* 4 (11 1978), 592–604. doi:10.1037/0278-7393.4.6.592
[23] Philipp Straubinger and Gordon Fraser. 2024. Gamifying a Software Testing Course with Continuous Integration. arXiv:2401.17740 [cs.SE] https://arxiv.org/abs/2401.17740
[24] Philipp Straubinger, Tim Greller, and Gordon Fraser. 2025. Teaching Software Testing and Debugging with the Serious Game Sojourner under Sabotage. arXiv:2504.19291 [cs.SE] https://arxiv.org/abs/2504.19291
[25] John Sweller. 2011. CHAPTER TWO - Cognitive Load Theory. Psychology of Learning and Motivation, Vol. 55. Academic Press, 37–76. doi:10.1016/B978-0-12-387691-1.00002-8
[26] Simone Tonhão, Marcelo Shigenaga, Julio Herculani, Andressa Medeiros, Aline Amaral, Williamson Silva, Thelma Colanzi, and Igor Steinmacher. 2023. Gamification in Software Engineering Education: a Tertiary Study. In *Proceedings of the XXXVII Brazilian Symposium on Software Engineering* (Campo Grande, Brazil) *(SBES '23)*. Association for Computing Machinery, New York, NY, USA, 358–367. doi:10.1145/3613372.3614193
[27] Universidade Federal da Paraíba. 2025. SIGAA - Sistema Integrado de Gestão de Atividades Acadêmicas. https://sigaa.ufpb.br/ Accessed on May 15, 2025.
[28] Pedro Henrique Valle, Ellen Barbosa, and José Maldonado. 2015. Um Mapeamento Sistemático Sobre Ensino de Teste de Software. doi:10.5753/cbie.sbie.2015.71
[29] Jari Vanhanen, Timo O.A. Lehtinen, and Casper Lassenius. 2018. Software engineering problems and their relationship to perceived learning and customer satisfaction on a software capstone project. *Journal of Systems and Software* 137 (2018), 50–66. doi:10.1016/j.jss.2017.11.021