Journal on Interactive Systems, 2025, 16:1, doi: 10.5753/jis.2025.5432

Practices, Process Stages and Examples of an Extreme Programming Proposal in a Playable Mode

Victor Travassos Sarinho [**State University of Feira de Santana** | ***vsarinho@uefs.br***]

Digital Applied Entertainment Lab. (LEnDA), State University of Feira de Santana, Av. Transnordestina, s/n, Novo Horizonte - BA, 44036-900, Brazil.

Received: 16 January 2025 • **Accepted:** 13 September 2025 • **Published:** 21 September 2025

Abstract: *Background:* There are several studies focused on identifying and defining gamification strategies in software development processes. These strategies are also applied by agile methods, which can create a context of recognition and reward for the completion of activities in a software project. *Purpose:* This paper presents a reinterpretation of the Extreme Programming (XP) practices and process stages in order to provide a "playable mode" for the XP development. *Methods:* XP practices and process stages are linked to terms and activities applied in digital games, enabling a reinterpretation from a playable and gamified perspective. *Results:* Gamified XP practices and process stages are explained and exemplified, demonstrating the feasibility of the proposed gamified reinterpretation for the XP software development. *Conclusion:* A software development methodology based on agile gameplays obtained by the XP reinterpretation was proposed, becoming a possible solution to improve the flow state in XP developers.

Keywords: Agile Methods, eXtreme Programming, Practices and Process Stages, Software Development, Playable Mode, Flow State

1 Introduction

When considering the term "game", associations with leisure, entertainment, or time-wasting often arise. However, this perception of waste is an example of how time tends to reshape old ideas and thoughts. This becomes especially evident with the gamification idea, which seeks to apply successful game mechanics and dynamics in real-world situations [Costa and Marchiori, 2015].

Gamification concepts and their applications can be seen as a consequence of a growing trend aimed at engaging employees and collaborators, seeking to help them achieve a "state of flow" [Csikszentmihalyi *et al*., 2005] in their tasks. The *flow* state refers to a mental state of complete absorption in an activity, marked by intense immersion, loss of time and space awareness, and sustained focus on task execution [Csikszentmihalyi *et al*., 2005].

Regarding software development approaches, there are interesting works focused on the identification and definition of gamification strategies in Software Engineering (SE) processes [García *et al*., 2017]. These works proposed the inclusion of *ranks*, *badges*, *missions*, and other gamification components to create reward systems. As a result, they enhance engagement and participation in system development [Pedreira *et al*., 2015], an activity where the human factor is still of great relevance in the productive process.

Considering the human factor, agile methodologies have emerged in recent years as an attempt to improve the quality of delivered software, through the recognition of the importance and limitations that human beings have in the production of desired systems [Al-Saqqa *et al*., 2020]. In this sense, many agile practices have been proposed [Krancher, 2020], highlighting both the importance of distributing the knowledge of system production among its collaborators, as well as the need to produce small system releases directly integrated with the client at different development moments.

Drawing a parallel between gamification and the agile world, it is possible to perceive that gamification practices can also be applied on agile methods, as both create a context of recognition and reward for the completion of activities allocated in the process. However, there is a world of possibilities in the universe of games that can be directly applied or serve as a guide to stimulate the flow of people involved in the production of software systems.

In this sense, and making a direct allusion to *eXtreme Programming* (XP) [Anwer and Aftab, 2017], this work presents a reinterpretation of the execution of XP practices and process stages in a "*playable mode*" [Sarinho, 2024], together with examples and suggestions about the application of playable production practices and process as an extended approach. As a result, with the definition of an agile playable mode, it is expected to take the gamification use to another level, providing a software development methodology based on agile *gameplays* for software production steps. Thus, by incorporating other existing game elements, this approach seeks to establish a coherent and engaging software process to support the development of desired projects in a playable manner.

For organizational purposes, Section 2 presents the theoretical foundation of this work, covering key aspects of agile development, the flow state, and gamification concepts. Section 3 describes the potential reinterpretation of XP production and process practices in a playable mode. Section 4 provides a practical example demonstrating how playable XP practices and processes could be implemented in a software project following a unified game theme. Finally, Section 5 presents the conclusions and future work of this project.

2 Theoretical Foundation

Among the various methodologies encompassed by agile software development, *eXtreme Programming* (XP) [Anwer and Aftab, 2017] stands out for its focus on enhancing software quality and responsiveness to evolving customer requirements through frequent releases in short development cycles. XP promotes a collaborative environment where developers, clients, and managers work closely and continuously, creating an interesting environment for integrating gamified elements and promoting engagement and flow during software development. To support this perspective, the following subsections present the conceptual foundations of agile software development, gamification, the flow state, and their intersections, offering the necessary theoretical basis for understanding how these elements can be integrated in a playable XP context.

2.1 Agile Software Development

Software development methodologies aim to increase the productivity of development teams, accelerating the time to market for solutions, reducing development costs, and improving customer satisfaction [Lee and Chen, 2023]. To achieve these objectives, agile software development has gradually generated public discussions since the 1990s, which can be defined as a collective term for collaborative work based on a set of values and principles that has become a standard approach for the software industry [Naik and Jenkins, 2019].

Agile software development emerged as a solution for the software crisis, where most software development projects failed to meet user requirements [Bera *et al*., 2023]. It promotes adaptive planning and evolutionary development, sharing the same values of the software process and encouraging rapid and flexible responses to changes through early delivery and continuous improvement of the produced systems [Sutherland and Sutherland, 2014]. Furthermore, agile development emphasizes moderate planning, people-oriented cooperation, face-to-face communication, self-organization, self-management, and rapid development of the desired systems [Williams, 2010].

2.2 Gamification

Gamification broadly encompasses technological, economic, cultural, and social developments in which reality becomes increasingly game-like, whether by deliberate design or as an emergent transformation [Koivisto and Hamari, 2019]. It involves the application of game-based mechanics and dynamics to various processes, aiming to establish reward systems that capture the attention and engagement of the individuals involved.

Digital games, on the other hand, make use of the concept of entertainment in highly empathetic interfaces to conquer the attention and interaction of the target audience. For that, they explore the application of important aesthetic elements, such as narrative, challenge, socialization, among others [Hunicke *et al*., 2004], with the aim of engaging and emotionally connecting with their players, so that they remain continuously in a magic circle [Juul, 2008] of gameplay.

By incorporating well-balanced and contextualized game elements through empathetic interface design, gamification enables the creation of meaningful user experiences that promote engagement and immersion for their players. In this sense, gamification acts as a catalyst within agile software development teams, stimulating motivation and focused attention, as well as promoting the necessary conditions to reach the *flow state* that increases both personal satisfaction and productivity at work [Coutinho *et al*., 2021].

2.3 The Flow State

The *flow state* presents a combination of four elements: intrinsic motivation, maximum concentration, a very positive emotional state, and a high performance rate [Kamei, 2014]. The flow state can be described as a mental state that occurs when a person performs an activity and feels completely absorbed in a sense of energy, pleasure, and total focus on what they are doing [Csikszentmihalyi *et al*., 2005]. In other words, it is a state where, in its essence, it is characterized by complete immersion in what is being done, and by a consequent loss of the sense of space and time. Thus, with maximum focus and concentration, all action, all involvement, and all thought flow in sequence to the action, until the end of the activity itself [Csikszentmihalyi *et al*., 2005].

Despite being desirable, a constant flow state tends to be utopian and practically impossible to be achieved one hundred percent of the time [Rossetti and Ramos, 2022]. In this sense, a closer relationship to reality is presented with the flow state at the intersection between enjoyment and challenge, having the balancing of games on different aspects as a fundamental factor for increasing the chances of achieving the flow state [Rossetti and Ramos, 2022]. In other words, through well-designed game interfaces for their respective users, it is understood that they can help achieve the appropriate balance in order to provide the necessary conditions to promote the flow state [Souza Teixeira and Fonseca Ramos, 2014].

2.4 Games and Agile Development

Initiatives such as the *Planning Game* [Parsons, 2014], source code *Dojo* [Luz and Neto, 2012], and *Game-of-Games* (GoG) [Spil and Bruinsma, 2016], [Sarinho, 2020] demonstrate the potential of incorporating different types of digital game elements into the development of software systems. These approaches go beyond the common inclusion of traditional gamification components (e.g., points, rankings, rewards), introducing new mechanics and dynamics that can be effectively leveraged by software developers.

Developed projects, such as *Planning Poker* [Grenning, 2002], *Extreme Hour*, *XP Lego Game* [Parsons, 2014], and *Red-Green-Go!* [Embury *et al*., 2019] illustrated how agile practices can be enriched through gamification concepts. They enable a more engaging and enjoyable development experience, while demonstrating the feasibility and potential benefits of incorporating game-based dynamics into agile workflows.

In this context, and considering the previously explained convergence of gamification, flow theory, and agile methodologies, these developed initiatives highlight the evolution of game-based strategies in both traditional and agile software engineering processes. Furthermore, by promoting a playful, collaborative, and intrinsically motivating environment, such approaches create the ideal conditions for the emergence of the flow state during software development activities. As a result, these projects provide the necessary inspiration for the creation of new agile development processes based on different types of game elements, such as the playable XP approach presented in this work.

3 Defining a Playable XP

Recent state-of-the-art trends adopting agile development have been explored in the literature, especially in cloud computing, big data, and team coordination [Al-Saqqa *et al*., 2020]. These are studies that reveal new methods and practices capable of increasing the efficiency and quality of software development processes in their respective contexts. Further research also demonstrates how the design and testing of learning games related to agile methodology can serve as a tool for improving the comprehension of agile methods, in addition to giving them the opportunity to develop their creative, organizational, and analytical skills [Parsons, 2014].

In order to define a new agile approach within the perspective of games and gamification, this work seeks to define playable goals able to be applied at the next level of agility for existing software processes. To this end, the aim is to incorporate actions such as "programming by playing", "design challenges", or "programming competitions" into "collaborative multiplayer" development sessions in the agile production of systems based on existing game approaches.

3.1 Playable Production Practices

Taking the XP methodology as a starting point, agile production practices have been listed and successfully accepted by the SE community, such as pair programming, code refactoring, simple design, among others. Based on these practices, a reinterpretation of them was proposed within situations and experiences provided by games in general (Figure 1), such as:

- **Game-Driven Development**: Activities to be carried out in the development process stages should be performed through appropriately defined game mechanics, dynamics, and aesthetics. In this way, the act of playing - with a defined beginning, middle, and end, represented in a fictional experience distant from any existential problem or risk, capable of providing anxiety and hope both in the possibility of losing and also in the expectation of winning, in a voluntary way that generates a feeling of reward - becomes the guiding principle for solving problems in the target domain, aiming to provide a differentiated level of immersion for the collaborators in the development process itself.
- **Simple Challenges**: Developers should play matches divided into phases with small challenges, where the actions performed by the players generate effects that can be identified in the project. Considering that game actions will be used for the production of desired systems, it is important that the definition and achievement of these small game challenges be oriented towards the production of small artifacts capable of being used in the assembly of the final target software.
- **Multiplayer Programming**: The work carried out by people in production processes can be performed individually or collectively depending on the project's characteristics. In the case of games, single or multiplayer matches can be played in both collaborative and competitive modes, as well as in small groups or in a frenetic battle royale [Choi and Kim, 2018]. With the addition of Head-Up Displays (HUDs) that show the performance of each player and their group, there are interesting alternative ways to perform project activities in a playable and gamified way, tailored to each active competitor.
- **Code Replay**: Each match is unique and generates unique solutions. In this sense, nothing prevents players from replaying as many times as necessary, seeking each time to execute better "correct" actions for the respective project. As a result, multiple artifacts and releases with partial and complete solutions can be generated by the collaborators, within the continuous rhythm of effort that players employ when they are immersed in their chosen game matches.
- **Rule Standards**: Just as code patterns can be adopted in software projects, gameplay and action-response patterns can be defined for each game match to be carried out in each process activity. In this way, it becomes necessary to define a set of rules in order to impose on players a respect for the project's quality within the actions and responses defined and expected in the game.
- **Addiction Control**: Players, when they enjoy a game, when they enter the flow state provided by games, they tend to play it at a frenetic pace. In this sense, it is necessary to avoid "overwork" in a specific game, to maintain a sustainable rhythm for the project as a whole.
- **Continuous Victory**: Each phase or completed gameplay within a project is already a victory in itself. The laurels of victory may go to the winners, but the project itself is the main winner since new victorious releases of the project are obtained as a consequence of each completed gameplay.
- **Storytelling**: Each client has a story to be told about a problem. Each story has a beginning, a middle, and an end. Each story can be told in different ways, in order to create a plot capable of immersing players in the system of rules and patterns proposed by the production process for the game in question. It is up to each project manager to define at what point in the story and in which system of rules they will place their collaborators to start their matches.
- **Magic Circle**: A game truly begins when players enter the magic circle, requiring a shared understanding of the game's context. This shared understanding empowers players to take ownership of the rules, actions, and

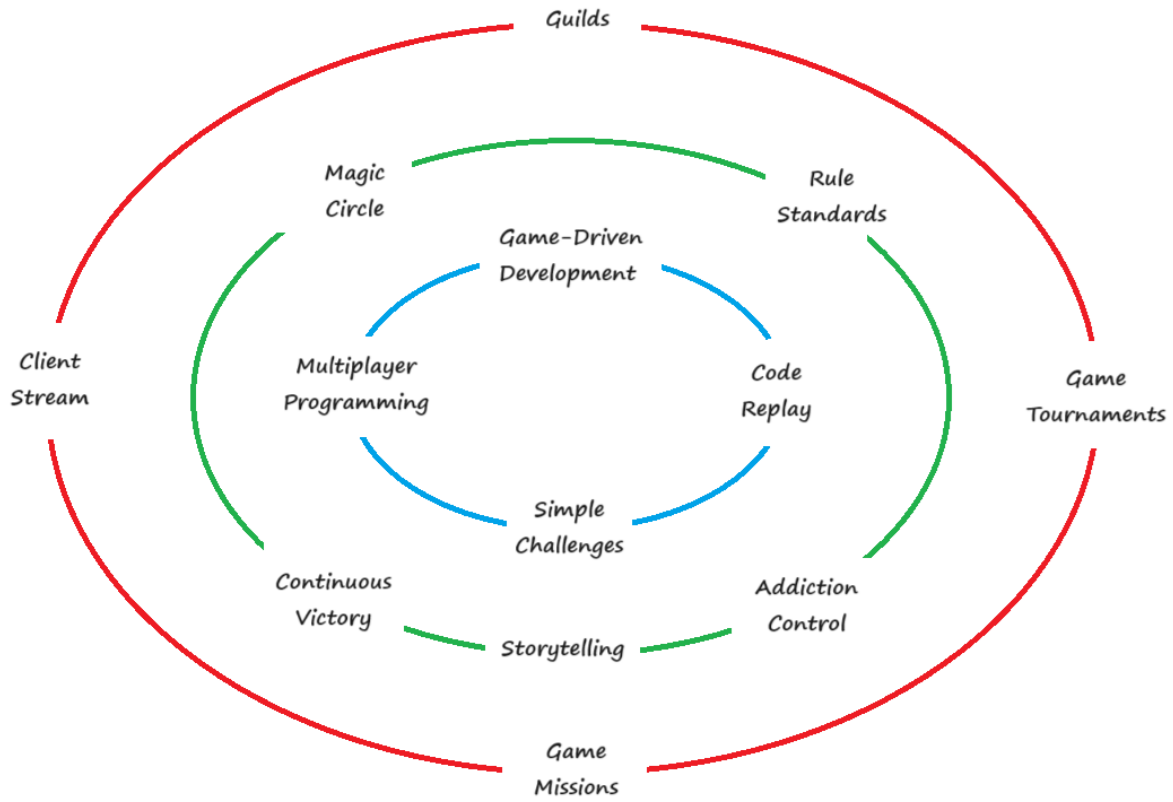

Figure 1. Suggested practices for a playable XP, where the smaller inner circle (blue) represents the core technical practices to be applied, the middle circle (green) represents the support practices that ensure effective team functioning, and the larger outer circle (red) aligns development with customer needs, creating a continuous cycle of feedback and improvement. Source: Author's Own.

patterns, allowing them to master the game over time.

- **Guilds**: Groups formed by players, each specializing in a particular role, strategically plan their actions to achieve the game's objectives. Successful guilds understand the value of every team member, with each member knowing when and how to contribute to ensure the group's success in completing assigned tasks.
- **Game Tournaments**: There are moments in project production where it is necessary to define which matches and in which phases will be played by the developers, which will bring the best results that will advance to the next stage until the final delivery of the project. In this sense, it is up to the managers, in agreement with the players and the project goals, to decide when the tournaments will be defined and held, generating trophies and recognition from the entire team for the best players who followed the rules defined in the game, as well as generated good results for the project.
- **Client Stream**: People like to watch and give opinions about game matches, and this would be no different with customers. In this sense, it is important to create different forms of customer participation in real time with the matches held, whether through live viewing and interaction during the matches, or by issuing relevant comments and suggestions that should be analyzed by the game teams.
- **Game Missions**: Every game has objectives, has missions to be fulfilled, whether they are primary or secondary. It is up to the project team to choose which and when certain challenges should be fulfilled in order to complete the great mission that is to complete the game as a whole and finalize the project planned by the team.

3.2 Playable Production Process

Using the previously indicated practices, some activities were identified, according to the XP production process. They can be performed based on these principles, in order to define a possible execution cycle of steps for an agile development process with "playable" milestones (Figure 2):

1. *Presentation and adaptation of the Storytelling to be used in games* (**Game plan**): At this moment, a planning is carried out to define which game style will be used in the developed project, modeling how they will appropriate the client's *Storytelling* to generate results in future *Continuous Victories*. It is also decided how the players will become aware of the client's problems and the gameplay results for the project, in a way adapted for each applied game approach in question.
2. *Definition of Simple Challenges according to the Rule Standards that will be used in the chosen games for the project* (**Gameplay plan**): Depending on the mission, the game context, and the players in the match, some challenges and rules may or may not be applied to each game, such as: number of possible word matches for the creation of dynamic *user stories* [Sarinho, 2019]; re-

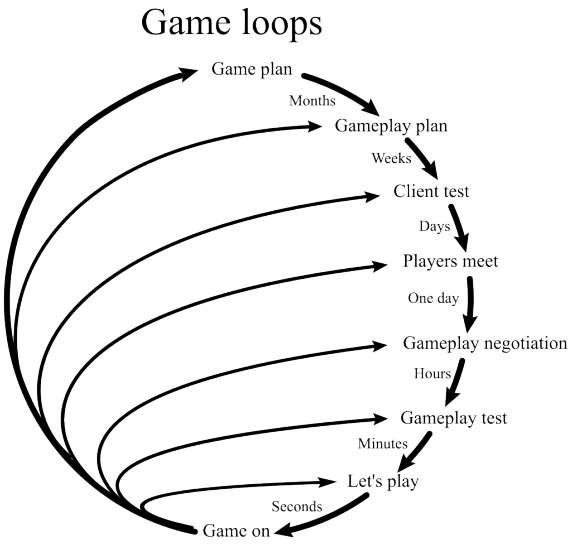

Figure 2. Suggested process steps for a playable XP, showing a continuous and gamified flow among proposed technical practices, team support activities, and customer-aligned strategies. Source: Author's Own.

sponse time to choose a mutation rule to be applied in software tests presented at a "frenetic" pace; or choice of actions that the player can perform for the simplification or composition of interfaces according to User eXperience (UX) in an atomic interface design [Frost, 2016], [Odushegun, 2023]. It is worth mentioning that possible tutorial missions may also be defined at this stage, thus improving the performance and initial immersion of players in the first matches of each game.

3. *Transmission and continuous monitoring of comments in the Client Stream* (**Client test**): This is a stage continuously carried out by the clients in the project, through the monitoring of HUDs and the status of each game. It is also a decision-making stage performed together with the development team, looking for possible changes in the game mode as a whole, since the clients essentially act as the game master of each match played in the development process.
4. *Storytelling and Game Mission updates* (**Players meet**): Through repeated playtesting of the selected games, performing their *Simple Challenges* according to the defined *Rule Standards*, new game states are regularly captured. Based on these states, new rules and challenges can be defined and released to players, along with updates to the game's *Storytelling*, in order to guarantee variability and surprise for the players involved. It's important to note that since multiple distinct games may be incorporated in the final system, the state of each game after playtesting can influence other applied games with the available playtesting data.
5. *Guild and Game Mission selection* (**Gameplay negotiation**): The development of a system has several parallel missions that can be carried out in each available game, such as screen production, architecture definition, code programming, test writing, etc. It is up to the players to choose their partners and which initial missions they will venture into.
6. *Continuous Code Replay with Addiction Control monitoring* (**Gameplay test**): With the definition of *Simple Challenges* according to the defined *Rule Standards*, multiple matches are repeatedly played by the players, in order to generate multiple artifacts capable of being used in the system project. In this sense, it is necessary to define and carry out the monitoring of the game pace of each player, in order to prevent potential "burnout" or "overwork" by the continuous and addictive act that the games can provide to them.
7. *Search for the Magic Circle for the players involved according to the chosen Multiplayer Programming mode* (**Let's play**): Whether collaborative or competitive, whether individual or in groups, it is up to the manager and the teams to decide how they want to play each proposed game, in order to guarantee fun and pleasant moments of system production with them.
8. *Tracking Continuous Victory and organizing Game Tournaments* (**Game on**): For each match played, artifacts generated from the system in production need to be documented and organized in prototypes and releases of the system. Internal tournaments may also be held at this stage of the project, depending on project urgency and the team's current development velocity for upcoming client releases.
9. *Repetition from the initial steps to the completion of the Game Missions* (**Game loops**): Based on the performance of the players in the matches, the comments of the clients and the progress of the project development, it is up to the development team to decide which readaptations will be necessary for the purpose of obtaining the desired final project successfully.

4 Applying a Playable XP

Considering the 9 proposed steps for the XP-inspired process from a gamified perspective, some possible examples and suggestions can be provided for each process stage to develop a desired system.

4.1 Game Plan

Regarding the presentation and adaptation of the *Storytelling* in the Game Plan stage, it involves defining the project's overall narrative, as well as its possible outcomes, which can be considered *Continuous Victories*. Thus, within the general context of an Automated Teller Machine (ATM) control system to be produced, a control system for Gringotts Wizarding Bank's Magic Teller Machines (MTMs) will be developed, according to the context of the Harry Potter[1] saga. This system will be developed by "witches" and "wizards" distributed in *Guilds* linked to the Hogwarts Houses.

To structure the proposed narrative in a way that the system functionalities are represented as tasks, different strategies can be adopted. Two viable alternatives are: (A) the use of Problem-Based Learning (PBL) layout [Hung *et al*., 2008], or (B) the adoption of Behavior Driven-Development (BDD) story formats [Farooq *et al*., 2023].

[1] https://www.harrypotter.com

PBL emphasizes the presentation of real-world problems as the starting point for learning and task execution [Wood, 2003]. This approach supports immersion, critical thinking, and active problem-solving—key aspects for maintaining player engagement throughout the gamified process. On the other hand, BDD is a software development technique that expresses system behavior in a structured, human-readable format, typically using the Gherkin syntax (Given-When-Then syntax) [Farooq *et al*., 2023]. This format enables clear communication between developers and stakeholders while providing testable and verifiable acceptance criteria.

In the context of Playable XP, PBL can be framed as challenges that require collaborative analysis, iterative exploration, and solution development, aligning naturally with the storytelling and mission-based elements. For the BDD format, scenarios can be reinterpreted as "magical challenges" to be solved through spells (i.e., code), enhancing the thematic integration between narrative and technical specification. As a result, both strategies enable the representation of functional requirements through concrete tasks and provide a basis for defining evaluation criteria. They can define functional elements to be implemented, along with corresponding preconditions and postconditions for evaluation. Thus, through these conditions, it is possible to demonstrate and verify the success or failure of the outcomes achieved through the Continuous Victories performed during each gameplay.

4.1.1 (A) Storytelling for the Gringotts Magic Teller Machines (MTMs) following PBL

Storytelling: Imagine that the world has been hit by a wave of intermittent "digital blackouts". The communication infrastructure and banking systems are unstable. Traditional banks are having difficulty keeping their services online, causing panic and confusion among the population. A small local bank, "Gringotts Bank", known for its innovative approach and use of advanced technology, decides to hire a team of "digital magicians" to develop a new MTM system resilient to these failures and that offers a more interactive and secure experience for its customers.

The Problem: Gringotts Bank needs a new ATM system that continues to operate during digital blackouts and provides the following functionalities:

1. *Withdrawal*: The customer inserts the "Magic Wand" (card) and pronounces the "Vault Password" (PIN). Due to the blackouts, the system must have an offline mode that allows limited withdrawals based on an "Energy Crystal" (offline balance) that will be managed by Gringotts' magical network.
2. *Deposit*: The customer can deposit "Gold Coins" (money) into the MTM. In offline mode, the deposit is registered locally and synchronized with Gringotts' magical network as soon as the connection is re-established.
3. *Balance Inquiry*: The customer can check the balance online and offline (Energy Crystal). The interface displays the balance with a "Mana" (magical energy) visualizer, representing the available funds.
4. *Inter-Account Transfer*: The customer can transfer "Gold Coins" to other customers, when both accounts are online. The transfer is represented as a "Transfer Spell".
5. *Leveling Up*: The user can complete small tasks at the MTM, such as "checking the balance 3 times a week", to gain "experience" and "rewards" from the bank.
6. *Security Level*: As the customer levels up in the bank, the associated account can unlock additional features, for example, increasing the offline withdrawal limit.
7. *Extra Items*: Depending on the customer's level, the MTM can offer extra virtual "magic items" to its customers, such as "Balance Boost Potion" (temporary bonus) or "Security Amulet" (extra protection against fraud).

4.1.2 (B) Storytelling for the Gringotts Magic Teller Machines (MTMs) following BDD

General Context: Gringotts Bank is implementing more modern and secure MTMs, as witches and wizards need to access their galleons, sickles, and knuts easily and efficiently.

Story 1: Galleon Withdrawal - Retrieving Vault Funds

Narrative: As a wizard needing galleons to buy potion ingredients in Diagon Alley, I want to withdraw money from my vault at Gringotts using an MTM.

Scenario 1.1: Successful Withdrawal - Transaction Approved by the Goblin;

Given that my Magic Wand is registered and authorized with Gringotts Bank and my vault contains 50 Galleons, And the MTM is connected to Gringotts' magical network, **When** I insert my Magic Wand and pronounce the Vault Password (PIN) correctly, And I cast the withdrawal spell for 20 Galleons (withdrawal command), **Then** 20 Galleons should be magically withdrawn from my vault, And the MTM should display a message with glowing runes confirming the transaction: "Transaction Approved - 20 Galleons withdrawn from Vault [Vault Number]", And my new vault balance should be 30 Galleons.

Scenario 1.2: Insufficient Funds - Withdrawal Spell Failed - Empty Vault;

Given that my Magic Wand is registered and authorized with Gringotts Bank and my vault contains only 10 Galleons, And the MTM is connected to Gringotts' magical network, **When** I insert my Magic Wand and pronounce the Vault Password correctly, And I cast the withdrawal spell for 20 Galleons, **Then** an error message, accompanied by black ravens flying around, should be displayed on the MTM: "Withdrawal Spell Failed - Insufficient Funds for the Spell", And my vault balance should remain at 10 Galleons.

Scenario 1.3: Incorrect Vault Password (Unauthorized Access Attempt) - Vault Protection Alert - Goblins on Alert;

Given that my Magic Wand is registered, And the MTM is connected to Gringotts' magical network, **When** I insert my Magic Wand and pronounce an incorrect Vault Password three times, **Then** my Magic Wand should be temporarily disabled for use in MTMs (like a temporary "Confundus" Charm), And an audible alert with the sound of moving iron chains should be heard, and the message "Vault Protection Alert - Goblins on Alert - Unauthorized Access Attempt

Recorded" should be sent to Gringotts security.

Story 2: Balance Inquiry - Viewing the Gold in the Vault

Narrative: As a wizard, I want to check my vault balance at Gringotts using an MTM.

Scenario 2.1: Successful Inquiry - Treasure View;

Given that my Magic Wand is registered and authorized with Gringotts Bank and my vault contains 75 Galleons, 12 Sickles, and 5 Knuts, And the MTM is connected to Gringotts' magical network, **When** I insert my Magic Wand and pronounce the Vault Password correctly, And I cast the balance inquiry spell (inquiry command), **Then** the MTM should display a holographic image of my vault with the exact amount of Galleons, Sickles, and Knuts, with the message: "Your Gringotts Treasure: 75 Galleons, 12 Sickles, 5 Knuts".

4.2 Gameplay Plan

For this stage, it is necessary to define *Simple Challenges* according to the *Rule Standards* that will be used in the games chosen for the project. Thus, based on the PBL and BDD examples previously described, some Scrum sprints containing *Simple Challenges* for the development of the MTM system will be presented, focusing on contracts representing evaluation *Rule Standards* defined by constraints (pre- and post-conditions) and Gherkin descriptions (Given-When-Then).

4.2.1 Sprint 1: Authentication and Basic Withdrawal

Sprint Goal: Implement authentication with the Magic Wand and basic Galleon withdrawal.

Stories: As a wizard, I want to withdraw Galleons from my vault using my Magic Wand and Vault Password.

Indicated Tasks:

1. Develop the Magic Wand reading module.
2. Implement the Vault Password verification.
3. Create the vault debit logic.
4. Develop the basic MTM interface for withdrawals.

Contracts (Constraints):

1. *Pre-conditions (Withdrawal)*: The Magic Wand must be registered and authorized, and the vault must exist.
2. *Post-conditions (Successful Withdrawal)*: The vault balance must be reduced by the withdrawn amount, and a transaction record must be created.
3. *Post-conditions (Insufficient Funds)*: An "Insufficient Funds" error message must be displayed, and the vault balance must not be changed.
4. *Gherkin Rules*: Scenarios 1.1 and 1.2 (Successful Withdrawal and Insufficient Funds).

Acceptance Criteria:

1. The MTM must correctly authenticate the Magic Wand and the Vault Password.
2. The withdrawal must be performed successfully if there are sufficient funds.
3. An appropriate error message must be displayed if the balance is insufficient.
4. Unit and integration tests must cover the Gherkin scenarios and must pass successfully upon execution.

4.2.2 Sprint 2: Security and Error Handling

Sprint Goal: Implement security measures and error handling, including temporary Wand blocking in case of incorrect passwords.

Stories: As a wizard, I want my vault to be protected against unauthorized access.

Indicated Tasks:

1. Implement temporary Wand blocking after multiple incorrect password attempts.
2. Create the alert system for Gringotts in case of unauthorized access attempts.
3. Handle other potential errors, such as communication failures with the vault database.

Contracts (Constraints):

1. *Pre-conditions (Withdrawal)*: The Magic Wand must be registered and authorized, and the vault must exist
2. *Post-conditions (Incorrect Password)*: The Wand must be temporarily blocked, and an alert must be sent to Gringotts.
3. *Gherkin Rules*: Scenario 1.3 (Incorrect Vault Password).

Acceptance Criteria:

1. The Wand must be blocked after three consecutive incorrect password attempts.
2. An alert must be generated and logged at Gringotts.
3. The system must handle other exceptions robustly, without presenting unexpected errors.

4.2.3 Sprint 3: Balance Inquiry and Interface Refinements

Sprint Goal: Implement balance inquiry and improve the MTM interface.

Stories: As a wizard, I want to check my vault balance quickly and easily.

Indicated Tasks:

1. Develop the balance inquiry logic.
2. Create a gamified interface to display the balance.
3. Refine the overall MTM interface, improving usability.

Contracts (Constraints):

1. *Pre-conditions (Withdrawal)*: The Magic Wand must be registered and authorized, and the vault must exist.
2. *Post-conditions (Balance Inquiry)*: The current vault balance, including Galleons, Sickles, and Knuts, must be displayed.
3. *Gherkin Rules*: Scenario 2.1 (Successful Inquiry).

Acceptance Criteria:

1. The balance must be displayed correctly in the Galleons, Sickles, and Knuts format.

2. The interface must be intuitive and visually appealing.

More detailed contract options using a formal notation can also be specified, as exemplified by the following reinterpretation of some contracts from Sprint 1:

1. *Pre-conditions (Galleon Withdrawal)*: wand.registered == true and wand.authorized == true and vault.exists(vaultNumber) == true and vault.balance(vaultNumber) >= withdrawalAmount
2. *Post-conditions (Successful Withdrawal)*: vault.balance(vaultNumber) == vault.balance(vaultNumber)@pre - withdrawalAmount and transaction.recorded(vaultNumber, withdrawalAmount, dateTime)
3. *Post-conditions (Insufficient Funds)*: message.displayed == "Insufficient Funds" and vault.balance(vaultNumber) == vault.balance(vaultNumber)@pre

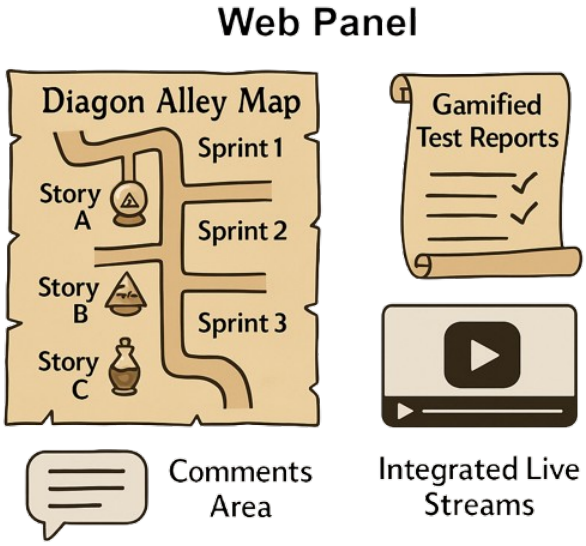

Figure 3. Conceptual illustration of a gamified Client Stream Panel for tracking project progress in Playable XP. Source: Generated by ChatGPT and edited by the author on July 31, 2025.

4.3 Client Test

Considering the continuous transmission and monitoring of client comments in a *Client Stream*, and focusing on the Gringotts MTM project and their exemplified sprints, three approaches can be adopted: *generic video streaming platforms*, such as private YouTube channels and closed Facebook groups; *self-built platforms*, for tracking the evolution and results of the game sessions; or *assigning responsible professionals* within the project, able to share the session data and perform desired tests with clients.

Regarding generic video streaming platforms, private channels can be created where sprint reviews are live-streamed and remain recorded for later consultation. To do this, it is necessary to create a closed group on these platforms where updates, demonstrations, and discussions are shared. Thus, sprint reviews can be live-streamed within the group, and clients can interact via chat during the broadcast and leave comments on the recorded videos. Among the advantages obtained, it is possible to identify: the good client-developer social interaction; the availability of file sharing and discussion features; and the familiarity of using these platforms for many different users. As disadvantages, this is a less formal method than a dedicated video channel, and the organization of the project information can become a challenge as the project evolves.

Self-built platforms, which are more customized, can also be applied, such as a tracking panel with game elements. For this, the development of an interactive web panel that shows the project's progress in a gamified way can be implemented. Therefore, considering the Gringotts MTM, a panel that includes the following elements can be developed (Figure 3): *Diagon Alley Map*, representing sprints and developed tasks, together with mission status (Stories) represented by themed icons and descriptions; *Gamified Test Reports*, presenting scrolls or other visual elements for the test results; *Comments Area*, integrated into the panel for clients to leave feedback directly; and *Integrated Live Streams*, incorporating a video player for live broadcasts within the panel. Integration with project management tools (e.g., Jira[2], Trello[3], among others) can also be applied at this stage, facilitating the collection of final artifacts from each game session according to the performed Code Replay.

Considering the option of assigning responsible professionals within the project for this activity, a team member can be designated as the "Master of the MTMs" or "Guardian of Gringotts" for communicating progress to clients. This team member is responsible for sending periodic reports, conduct personalized video calls, perform desired tests with or without clients, and answer questions directly. This approach allows for closer and more personalized communication, presenting greater flexibility to address the specific needs of each client. However, in addition to the greater time demand for the person responsible for communication, it may not be scalable for a large number of clients and developers.

It is worth noting that it is not enough to just transmit project information itself. It is essential to create mechanisms to collect client feedback, such as polls, forms, question and answer sessions, and comment areas, as well as maintaining a record of decisions made based on client feedback, thus demonstrating transparency and acknowledging their participation.

It is also important to acknowledge that not all clients may engage with gamified environments in the same way. While game elements can enhance engagement for some, others may prefer more formal, minimalist, or task-oriented interactions. Therefore, the proposed gamification strategies should be positioned as optional layers that enrich communication without becoming mandatory entry points. The core functionalities—such as progress visualization, feedback submission, and sprint validation—must remain accessible through standard interfaces or direct communication with the development team, thus ensuring inclusivity across different client profiles and preferences.

[2]https://www.atlassian.com/software/jira
[3]https://trello.com

4.4 Players Meet

Considering the updates to the Storytelling and the status of the *Game Missions* (stories) based on the execution of process stages, new mechanics and dynamics can be proposed for conducting the project's process. In this sense, and considering the Gringotts MTM project, there are some examples of results presentations, playable mechanics, and gamified dynamics that can be applied to it, such as:

1. *Daily Prophecy*: An inspiring phrase or a tip related to the current mission is presented, like a "prophecy" guiding the day's work. For example: "Precision in the account details will ensure the safety of the vaults".
2. *Wand Pass*: Team members must have possession of the "Wand of Attention" (physical or virtual) during Daily Meetings so that they can "pass" it to other members of each team to report their development status. This creates a gamified flow and prevents anyone from failing to present their contributions about the project.
3. *Daily Prophet Report*: Each member reports their progress as if they were giving news to the Daily Prophet. Phrases like "The authentication potions are brewing and should be ready tomorrow!" or "A digital troll attacked the withdrawal code, but we have already subdued it!" add a touch of immersion in the project's theme.
4. *Status with Magical Creature Cards*: Each member can choose a magical creature card to represent their status on assigned tasks. As possible examples: the "Hippogriff" can represent a *Task in progress, with good progress;* the "Messenger Owl" can represent a *Task with minor problems that needs help*; and the "Troll" can represent a *Task with major difficulties, requiring urgent intervention*.
5. *Distribution of Merit/Improvement Cards*: Themed cards can be used to recognize team members' contributions (e.g., "Spell Champion Card", "Master Builder Card") or to identify areas for improvement (e.g., "Curse of Confusion Card", "Unpredictable Troll Card").
6. *Wizard Performance Scroll:* Each team member can have a personalized "control panel", showing their tasks, experience points, and obtained items. The interface can allow interaction with map elements, such as clicking on representative task icons to see sprint details or on a scroll to view test reports.
7. *Quick Status Spell*: After a Daily Meeting, a brief visual "status spell" is activated on the Client Stream or on a virtual board (e.g., Notice Board or Hall of Fame). This spell can be a simple graph showing the progress of tasks in each "shop" of Diagon Alley.
8. *Magical Progress Board:* Instead of a traditional Kanban board, a stylized map of Diagon Alley is used, where each shop or iconic location can represent a project phase or a sprint (Figure 4). Tasks can be represented by cards with themed illustrations, such as: a wand casting a spell for "Implement the Wand Reading Module", or a bag of galleons for "Create the vault debit logic". The cards can be moved between the "shops" (phases/sprints) as development progresses: "In Preparation" at the "Leaky Cauldron's Ingredients Cauldron" (Backlog), "In Progress" at "Ollivanders Wand Shop" (Sprint Backlog), "In Testing" performing "Sweetness Testing at Honeydukes" (In Review), and "Completed" on the "Weasleys' Wizard Wheezes Success Shelf" (Done).

Figure 4. Illustration of the "Magical Progress Board", a stylized task-tracking board based on a Diagon Alley map. Source: Generated by ChatGPT and edited by the author on July 31, 2025.

9. *Hall of Fame*: A virtual hall of fame is updated at the end of each Sprint, showing the ranking of the Guilds and Houses based on criteria such as: Number of completed tasks; Code quality assessed by reviews and tests; Participation in events and challenges; and Client feedback.
10. *Daily Lightning Challenge*: A small programming or design challenge, related to the current mission, can be proposed immediately after the Daily Meeting. This challenge should be quick (5-10 minutes) and offer a small reward. Example: "Write a small spell (function) to validate a specific date format".
11. *Guessing Mini-Game*: A brief mini-game can be held on the Client Stream, involving both clients and the team. This mini-game can be about the code, the interface, or the narrative. For example: "Guess which Guild implemented this functionality based on a code snippet".
12. *Spell Fitting*: Guilds are invited to include snippets of small code pieces (spells) provided by a code generation tool. Extra points are awarded to those who manage to make insertions accepted by the other guilds.
13. *Scoring and Leveling System:* With each successfully completed task, the team earns points that contribute to their progression. As they accumulate points, they advance in levels (e.g., from "Wizard Apprentice" to "Master Sorcerer"), unlocking new "spells" (tools or functionalities) or "magic items" (guild attack or protection resources). Small virtual or physical rewards can also be offered for achieved goals, such as prize give-

Figure 5. Visual representation of the "Laws of Magic and Magical Challenges" in the Playable XP context. Source: Generated by ChatGPT and edited by the author on July 31, 2025.

aways based on "guild progress".

14. *Laws of Magic and Magical Challenges*: Pre- and post-conditions defined by contracts can be presented as "Laws of Magic" that must be followed for the spells to work correctly (Figure 5). The Gherkin rules (Given-When-Then) can be seen as a "magical challenge" or a "puzzle" that needs to be solved through spells (code) based on Laws of Magic that can be used in different situations. In this approach, similar to those practiced in programming marathons, when a Law of Magic is evaluated from the execution of a spell to solve a proposed magical challenge, the interface can display a themed animation or visual effect indicating: Test Passing (glowing runes appear on the screen); or Test Failing (a mountain troll attack with visual effects appears on the screen). Test reports can be presented as magic scrolls with the results of the "challenges", showing which spells passed and which failed in solving the magical challenges.
15. *Wizard Council Evaluation*: Clients can be represented as a jury of wizards from the Ministry of Magic, evaluating the team's "magical work". They can give grades and feedback using thematic criteria, such as "Spell Precision", "Potion Potency", or "Effectiveness against Dark Creatures". Clients provide feedback and evaluate the product increment using a thematic voting system (e.g., "Approved by the Wizengamot", "Requires Adjustments from the Department of Mysteries").
16. *Progress Report as a Spell Book*: A document or presentation can be formatted as an ancient spell book, with illustrations, runes, and magical descriptions of what was developed. The encountered problems can be described as "curses" that were broken.
17. *Awards Ceremony*: At the end of the Sprint, "trophies" (virtual or physical, such as small themed objects) can be awarded to team members who excelled in different areas, such as "Best Authentication Spell", "Most Effective Anti-Error Potion", or "Digital Troll Tamer".
18. *Next Mission Preview*: At the end of the Sprint Review, a brief "teaser" of the next mission is presented, increasing anticipation and engagement for the next cycle. This teaser can be a short video, an enigmatic image, or a brief description of the new threat to the Gringotts MTM project.

4.5 Gameplay Negotiation

Considering the context of the Gringotts MTM project, the choice of a Guild and the Game Missions (Sprint/Story) should be aligned with the interests of the players, who are represented by both the development team and the stakeholder/client teams, to maximize engagement and motivation. However, in the Harry Potter universe, the choice of a Guild is directly related to a Hogwarts House, representing different values and characteristics of their members. In this sense, the houses can be used as metaphors for different areas of expertise or work styles within the team, such as:

1. *Gryffindor (Courage, Bravery, Chivalry)*: Represents the team focused on innovation, rapid resolution of complex problems, and implementation of challenging functionalities. They can focus on missions that involve calculated risks and creative solutions.
2. *Hufflepuff (Loyalty, Hard Work, Fairness)*: Represents the team focused on stability, reliability, rigorous testing, and attention to detail. They can focus on missions that require precision, organization, and quality assurance.
3. *Ravenclaw (Intelligence, Creativity, Wisdom)*: Represents the team focused on design, architecture, elegant solutions, and code optimization. They can focus on missions that involve strategic planning, analysis, and resolution of complex problems.
4. *Slytherin (Ambition, Cunning, Leadership)*: Represents the team focused on performance, efficiency, automation, and process optimization. They can focus on missions that involve performance improvements, test automation, and rapid value delivery.

Each team member should carry out a self-assessment to identify their strengths and preferences. A team discussion can help align perceptions and define which houses best represent the different groups or individuals. The choice of a house can also be made by voting or direct allocation, depending on the team dynamics. Finally, it is important to allow flexibility in assigning a Guild to a particular house, allowing members to change Guilds if their interests or roles change throughout the project.

Regarding the choice of the Game Mission that the Guild will participate in, it is important to consider some factors, such as: the team's interests, client priorities, the complexity and duration of the missions, and the players' connection to the mission's Storytelling. To facilitate this process, the missions can be categorized or tagged according to the Hogwarts Houses, so that teams can choose missions that fit their values and specialties. Another possibility is to gamify the mission selection, through the use of "magic coins" to purchase the missions that the team considers most important.

The choice of Guild and Game Mission can also be shared with clients in the Client Stream. This increases transparency

and allows clients to better understand the team's decisions. Furthermore, clients can provide feedback on the chosen missions and even suggest new missions. In this way, there is a collaborative process and alignment with client priorities for the choice of Guild and Game Mission, thus contributing to the success of the project and the satisfaction of all stakeholders.

4.6 Gameplay Test

To ensure continuous *Code Replay* with *Addiction Control* monitoring, an interface that offers high playability and supports a fast-paced action-response rhythm for its players is necessary. In this sense, creating interfaces that allow rapid gameplay with visual programming elements, or interaction with tools that generate partial source code, can be helpful at this stage.

The idea is to provide visual and interactive elements capable of representing and manipulating the project's development progress based on playable mechanics and dynamics, in order to provide interesting *Rule Standards* for guiding the project's progression through gameplay. In this sense, and following interesting initiatives such as Primitive[4], VRIDE[5], BlocklyVR [Hedlund *et al*., 2023] and Cubely [Vincur *et al*., 2017], as potential inspirations for this work, the creation of playable interfaces that follow the Scratch[6] or Minecraft[7] style, which allows programming source code in the *drag'n'drop* style, are suggested. The definition of visual components able to be programmed, such as Unit+Bolt Visual Scripting [Knutsen, 2021], in a low-code/no-code perspective [Rokis and Kirikova, 2022], as well as the use of LLMs [Kumar, 2024] generating small "spells" (code) to be fitted into the main project according to the defined gameplay, can also be applied in this perspective.

For a development interface in the Scratch/Minecraft style, the source code would be represented by interlocking blocks in a 2D/3D environment. Each type of block would represent a different code element, such as: blocks of different colors for variables, functions, operators, etc.; blocks with specific textures and formats for control structures (if, for, while); blocks with symbols or icons to represent specific commands, such as "WithdrawGalleons" or "CheckBalance" for the MTM system, among other elements. Players would build the code by stacking, connecting, and organizing the blocks in the 2D/3D space, where the order and connection of the blocks would define the program's logic. When executing the code, the blocks could animate or emit visual effects to show the execution flow, where a variable block could change color upon receiving a value, or a function block could glow when called. Following a generative programming approach [Sarinho and Apolinário, 2009], the visual code built with blocks would eventually be converted to a real programming language (e.g., Java[8], Python[9], etc.) and could finally be integrated into the main MTM system.

Regarding the code generated by LLMs, these could be integrated into the playable programming interface through chat commands, by requesting the LLM to generate small snippets of representative code blocks (e.g., "/createBlock WithdrawGalleons amount:10"). If the playable programming environment is not being used, dynamics that seek to fit partial code into the project itself with a limit of attempts or time between players could be applied.

It is worth noting that, due to the pursuit of continuous Code Replay, all player actions in the playable programming approaches must be recorded in a reproducible format, such as the actions of building, editing, and executing the code in the suggested playable programming interface. With the visual reproduction of the code construction itself, the sequence of players' programming actions is presented, thus allowing for a better analysis of the development process, as well as the identification of errors and learning from the strategies applied by other players.

Finally, regarding Addiction Control, usage metrics need to be collected, providing data on interface usage time, frequency of game sessions, and other relevant metrics. The idea is to define usage time limits and display alerts when these limits are reached. As a result, it is expected to reward players for maintaining healthy usage time, providing personalized feedback to players about their usage patterns and offering tips for a more balanced use. Situations where the player needs a stimulus to increase their playtime can also be identified, opening opportunities to offer "focus potions" (various incentives) designed to temporarily boost player productivity, but requiring a rest period after use.

4.7 Let's Play

To achieve the *Magic Circle* for players in the chosen *Multiplayer Programming* mode, it is essential to promote player engagement at this stage. Thus, external strategies beyond the matches can be used, such as organized cheering sections and awarding prizes to the victors, can help create an environment that encourages collaboration, healthy competition, and a sense of belonging.

For this project example, the Magic Circle represents the state in which the players (programmers) are fully immersed in the task of building the MTM system, engaged with the playable programming interfaces, collaborating with other players, and competing in a friendly manner, so that they feel challenged, rewarded, and part of a larger whole. However, for this state to be achieved by the players, several strategies to create and sustain the Magic Circle can be applied, such as:

1. *Contextualized Missions*: Each sprint/story should be presented as a mission within the Harry Potter universe, with a clear objective and a narrative that motivates the players. For example, instead of "Implement authentication", the mission could be "Protect Gringotts Vaults from a Death Eater attack".
2. *Characters and NPCs*: Introduce non-playable characters (NPCs) with relevant roles within the narrative. A Gringotts goblin could give instructions about the mis-

[4]https://primitive.io/
[5]https://github.com/Vito217/VRIDE
[6]https://scratch.mit.edu/
[7]https://www.minecraft.net/
[8]https://www.java.com/
[9]https://www.python.org/

sions, or a famous wizard could request a new functionality for the MTM.

3. *Themed Events*: Create themed events within the game, such as an event where players need to solve logical puzzles and programming challenges to "open" virtual vaults or decipher encrypted messages. This could involve data manipulation, search algorithms, or Boolean logic problem solving, all using the playable interface.
4. *Team Guilds*: Encourage the Guild teamwork by selecting missions that require the collaboration of multiple players. Reward systems for collaboration should be applied, in order to reward players for helping other team members, sharing knowledge, and contributing to the project's success.
5. *Rankings and Leaderboards*: Create rankings and leaderboards to display players' progress within the Guilds and Houses. The rankings can be based on different criteria, such as programming speed, code quality, number of completed missions, or contributions to the community.
6. *Competitions, Tournaments, and Rewards*: Organize regular competitions and tournaments with programming and construction challenges in the playable programming interface. It is important to offer rewards both to competition winners and to players who excel in the rankings. The rewards can be virtual (e.g., in-game items, experience points, titles, public recognition) or real (e.g., prizes, giveaways, etc.).
7. *Public Recognition*: The winners of the competitions and the players who excel in the rankings can be publicly awarded during broadcasts on the Client Stream, thus increasing player recognition and motivation. Interviewing outstanding players, showcasing their projects, and sharing their tips and strategies on the Client Stream can further enhance motivation.
8. *Feedback Collection, Adjustments, and Iterations*: It is necessary to collect player feedback about the game, the missions, the group dynamics, and the gamification strategies, and use the feedback to refine and iterate on the strategies, always seeking to improve the experience and maintain the Magic Circle within the project.

4.8 Game On

To record the *Continuous Victories* of each match played, as well as to organize *Game Tournaments*, it is necessary to use supporting tools to record victories and configure the desired tournaments.

Thus, focusing on the concept of Continuous Victory, a "match" in the MTM project can be defined as either the completion of a Game Mission (Sprint/Story) or the resolution of a specific challenge within the playable programming interface. With each successfully completed mission/challenge, a victory counter should be incremented, and in case of failure, it may either decrease or remain unchanged. Thus, a system for controlling match results should be designed to record the wins and losses of each player or Guild..

Victories can be displayed on an individual player panel, showing their level, experience points, Guild and House performance, and the player's longest winning streak. A global or house-specific leaderboard can also display the longest winning streaks, thus encouraging friendly competition among players. The longest winning streaks can be highlighted during broadcasts on the Client Stream, thus increasing public recognition of the players and winning teams.

Remember that, regardless of whether a win or loss is recorded, each match generates code that must be saved, contributing to the production of programmed artifacts for the final project. The visualization of the generated code depends on the programming tools used in the matches, which can be integrated with available configuration and maintenance systems.

Regarding the holding of Game Tournaments, these can add a more structured layer of competition and offer more significant prizes to project participants. As a tournament example, programming challenges can be held where players must solve specific problems using a playable programming interface, aiming to meet specific programming and delivery deadlines defined in the challenges. Competitions between the Guilds and the Houses can also be held, where teams must complete a series of challenges, accumulating points to win the tournament. For this, tournament formats with single elimination (after one loss), double elimination (second chance after one loss), or group stages followed by playoffs can be applied.

Available tournament management platforms can offer resources to create, manage, and track tournaments, including: bracket creation, match scheduling, results recording, registration management, and integration with existing streaming platforms. It is important that one or more "referees" can supervise the tournaments to ensure compliance with the rules and resolve any disputes. Depending on the playable programming environment used, it may be necessary to develop customized tools to record victories, configure tournaments, show player rankings, and display matches and their results on the Client Stream.

4.9 Game Loop

As the final step of the gamified process, it is necessary to evaluate whether the project tasks have been completed or whether it is necessary to repeat all the other steps from the initial stage until the completion of the listed *Game Missions*. The idea is that the process repeats itself in a continuous cycle, similar to a sprint cycle in agile methodologies, but with thematic and gamified elements to maintain engagement, as previously demonstrated.

At this stage of the process, the collection of match results (Game Missions/Sprints) is crucial for project monitoring and decision-making. It is up to the client and other project stakeholders to decide whether the project tasks are finished, which should be based on objective and transparent criteria.

In this sense, clear and specific acceptance criteria need to be defined for each task before the mission begins. A task is only considered complete when it meets all the acceptance criteria. A final review is also carried out with the clients to ensure that all functionalities have been implemented as expected and that they meet business needs. After client approval, the final product is delivered, and the project is considered complete. A closing ceremony within the project's

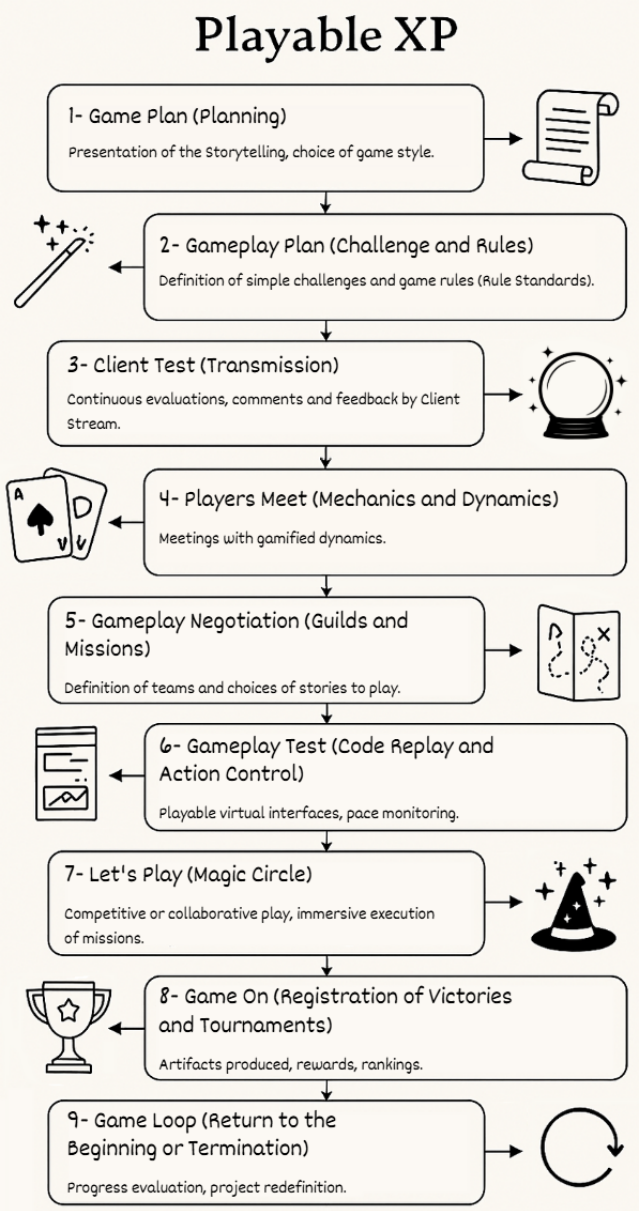

Figure 6. Visual summary of the playable XP process, illustrating the nine sequential proposed stages. Source: Author's Own.

gamified theme can also be held to celebrate its completion and recognize the team's work in a fun and motivating way.

4.10 Visual Summary

Different types of events, artifacts, cycles, participation alternatives, and dynamic elements were presented in this section to illustrate a practical example of how the proposed Playable XP approach can be implemented. To support reader comprehension and reduce the cognitive load associated with processing the detailed textual description, Figure 6 provides a visual summary of the nine steps of the gamified Playable XP process, as exemplified in this section. This diagram illustrates the dynamic flow among activities, artifacts, and participant roles throughout the development cycle, offering a cohesive and accessible overview of how the methodology unfolds in practice.

5 Discussing the Playable XP

The traditional XP methodology offers several advantages [Al-Saqqa *et al*., 2020], [Abrahamsson *et al*., 2017], such as:

- *Incremental development*: Supported through small and frequent system releases.
- *Improved productivity*: Achieved through rapid feedback for multiple versions that can be built daily and are only accepted if they pass the tests.
- *Simplicity*: Maintained through constant code refactoring.
- *Enhanced quality*: Ensured through the development of automated tests before integrating a feature into the system.

On the other hand, XP also suffers from the following limitations [Al-Saqqa *et al*., 2020], [Abrahamsson *et al*., 2017]:

- Lack of support for distributed teams, as it focuses on community and co-location.
- Test-driven development requires additional technical training for the involved team members.
- Informal documentation makes it difficult to maintain important project details.
- Actual client involvement is effective, but it is stressful and time-consuming.

Based on the practices and lifecycle indicated for executing a playable XP, it is possible to perceive that the advantages of traditional XP can be maintained, while some identified disadvantages can be addressed. Regarding the advantages, development in playable XP remains incremental, with the adaptation of client *Storytelling* into *Simple Challenges* and *Game Missions*. Gains in productivity, simplicity, and quality are also achievable through *Code Replay* in a *Continuous Victory* that generates results based on *Rule Standards*, with its *Addiction Control* applied.

Regarding the disadvantages of XP, the problem of lack of support for distributed teams can be addressed by implementing *Game Tournaments* featuring *Game Missions* to be solved by *Guilds* formed in their *Multiplayer Programming* work. Tutorial missions can be defined in the *Gameplay Plan* stage to ensure additional training in a fast, engaging way. Maintaining formal documentation can be achieved through the results and feedback from the played matches, which can be defined in the *Gameplay Plan* phase. Finally, client involvement can become dynamic and engaging, conducted through the transmission and continuous monitoring of comments via the *Client Stream* in the project's *Client Test* phase.

However, despite these improvements, the practical application of a playable XP approach still presents significant challenges when deployed in real-world software development scenarios. Key issues include the risk of narrative overload due to the integration of complex storytelling elements, technical feasibility constraints related to implementing game mechanics, resistance from clients unfamiliar with gamification-based approaches, and legal concerns surrounding the use of fictional intellectual property.

When considering the thematic choices used in the gamification strategy, such as references to the Harry Potter universe, while engaging for some users, these themes may not

resonate universally and could lead to issues of cultural imposition or exclusivity. In addition, considering ethical aspects related to inclusivity, developers with neurodivergent profiles might find metaphor-rich environments difficult to navigate, highlighting the importance of offering neutral or customizable narrative alternatives that accommodate diverse modes of interaction and comprehension.

In addition to inclusivity, there are also concerns about clarity and terminology that must be considered. The use of fictional terms (e.g., magic wands or houses) to describe software artifacts may increase cognitive load and hinder communication, especially in heterogeneous or interdisciplinary teams. These concerns are amplified when such terminology intersects with AI-assisted tools, which may misinterpret or misclassify non-standard labels.

Furthermore, the long-term sustainability of narrative-driven development should be critically considered. Maintaining a coherent and engaging storyline over extended project lifecycles may shift focus away from core development objectives and introduce unnecessary complexity. A more practical alternative would be to apply narrative elements in bounded contexts such as hackathons, onboarding processes, or short development sprints, where their motivational impact is maximized without compromising team focus, project clarity, or direction. Another possibility would be to gamify only the roles and personal goals of the developers within the process, working as a sort of Justice League[10] solving real-world client problems through the special powers each developer possesses.

6 Conclusions and Future Work

Human factors involved in the production of diverse systems currently represent a challenge in existing agile methodologies. Process gamification implements playable abstractions in different phases of a system's production, allowing real-time monitoring of a project's current state and its collaborators. With the application of a playable mode in software development methodologies, there is the possibility of extending the use of this gamification to another level, through the inclusion of other existing game elements in order to ensure a playable approach for the development of desired projects.

This work has therefore presented both the concern for and the proposal of a more comprehensive integration of game elements into software processes, aiming to create immersive and balanced contexts capable of spontaneously motivating team members to engage in free play within a playable process, while avoiding the creation of harmful and overly competitive environments for developers. As a result, the application of a playable mode in the XP methodology suggests that stages of a SE process can also be fully realized through games, expanding the possibilities for entertainment and immersion during development. The challenge, in this case, lies in conceiving SE activities that are performed through embedded game mechanics, dynamics, and aesthetics, such as incorporating a Game-of-Games (GoG) approach in the production of digital games.

[10]https://www.dc.com/characters/justice-league

In other words, it is crucial to strike a balance between creative engagement and essential aspects such as clarity, inclusivity, accessibility, and legal responsibility to ensure relevance and applicability across diverse software development teams. In this context, adapting the design principles of playable XP becomes a necessary step toward enabling the sustainable and responsible integration of game-based strategies in professional environments.

Another aspect that must be considered is the empirical validation of the playable XP proposal, which remains an open challenge. Although its exploratory nature is justified in this work, future investigations should prioritize structured evaluation through case studies and experimental designs — such as the MTM prototype — capable of assessing both the effectiveness and the limitations of the approach in varied organizational settings. Embracing open-data policies and transparent methodologies could further enhance the reproducibility and generalizability of results, thereby contributing to the academic and practical consolidation of this gamified model.

Finally, future work should also explore extending the playable mode beyond the XP framework, applying it to other software development processes to assess their feasibility through spontaneous and voluntary play. Particular emphasis should be placed on evaluating productivity gains associated with increased playability during system construction, as well as investigating the integration of empathic, gamified, and direct-manipulation interfaces into different stages of software production. These efforts will improve the understanding of how playful abstractions can enrich engineering workflows without compromising their rigor or effectiveness.

Declarations

Competing interests

The author declares that there are no conflicts of interest regarding the publication of this work.

Availability of data and materials

No additional materials or datasets are available or required for this work.

References

Abrahamsson, P., Salo, O., Ronkainen, J., and Warsta, J. (2017). Agile software development methods: Review and analysis. *arXiv preprint arXiv:1709.08439*. DOI: https://doi.org/10.48550/arXiv.1709.08439.

Al-Saqqa, S., Sawalha, S., and AbdelNabi, H. (2020). Agile software development: Methodologies and trends. *International Journal of Interactive Mobile Technologies*, 14(11). DOI: https://doi.org/10.3991/ijim.v14i11.13269.

Anwer, F. and Aftab, S. (2017). Latest customizations of xp: A systematic literature review. *International Journal of Modern Education and Computer Science*, 9(12):26. DOI: http://dx.doi.org/10.5815/ijmecs.2017.12.04.

Bera, P., Wautelet, Y., and Poels, G. (2023). On the use of chatgpt to support agile software development. In *The Second International Workshop on Agile Methods for Information Systems Engineering (Agil-ISE 2023) co-located with the 35th International Conference on Advanced Information Systems Engineering (CAiSE 2023)*, volume 3414, pages 1–9. CEUR.

Choi, G. and Kim, M. (2018). Battle royale game: In search of a new game genre. *International Journal of Culture Technology (IJCT)*, 2(2):5.

Costa, A. C. S. and Marchiori, P. Z. (2015). Gamificação, elementos de jogos e estratégia: uma matriz de referência. *InCID: Revista de Ciência da Informação e Documentação*, 6(2):44–65. DOI: https://doi.org/10.11606/issn.2178-2075.v6i2p44-65.

Coutinho, L. L., Vieira, I. D. P., and de Souza, J. C. P. (2021). O estado de flow na alta performance de líderes organizacionais the flow state in high performance of organizational leaders. *Brazilian Journal of Development*, 7(8):83333–83348. DOI: https://doi.org/10.34117/bjdv7n8-508.

Csikszentmihalyi, M., Abuhamdeh, S., and Nakamura, J. (2005). Flow. *Handbook of competence and motivation*, pages 598–608.

Embury, S. M., Borizanov, M., and Jay, C. (2019). Red-green-go! a self-organising game for teaching test-driven development. *Agile and Lean Concepts for Teaching and Learning: Bringing Methodologies from Industry to the Classroom*, pages 415–441. DOI: https://dx.doi.org/10.1007/978-981-13-2751-3_19.

Farooq, M. S., Omer, U., Ramzan, A., Rasheed, M. A., and Atal, Z. (2023). Behavior driven development: A systematic literature review. *IEEE Access*. DOI: http://dx.doi.org/10.1109/ACCESS.2023.3302356.

Frost, B. (2016). *Atomic design*. Brad Frost Pittsburgh.

García, F., Pedreira, O., Piattini, M., Cerdeira-Pena, A., and Penabad, M. (2017). A framework for gamification in software engineering. *Journal of Systems and Software*, 132:21–40. DOI: https://doi.org/10.1016/j.jss.2017.06.021.

Grenning, J. (2002). Planning poker or how to avoid analysis paralysis while release planning. *Hawthorn Woods: Renaissance Software Consulting*, 3:22–23. `https://sewiki.iai.uni-bonn.de/_media/teaching/labs/xp/2005a/doc.planningpoker-v1.pdf`, Accessed: 20 September 2025.

Hedlund, M., Jonsson, A., Bogdan, C., Meixner, G., Ekblom Bak, E., and Matviienko, A. (2023). Blocklyvr: Exploring block-based programming in virtual reality. In *Proceedings of the 22nd International Conference on Mobile and Ubiquitous Multimedia*, pages 257–269. DOI: https://dx.doi.org/10.1145/3626705.3627779.

Hung, W., Jonassen, D. H., and Liu, R. (2008). Problem-based learning. In *Handbook of research on educational communications and technology*, pages 485–506. Routledge. DOI: http://dx.doi.org/10.1007/978-1-4419-1428-$6_2$10.

Hunicke, R., LeBlanc, M., Zubek, R., *et al*. (2004). Mda: A formal approach to game design and game research. In *Proceedings of the AAAI Workshop on Challenges in Game AI*, volume 4, page 1722. San Jose, CA.

Juul, J. (2008). The magic circle and the puzzle piece. `https://jesperjuul.net/text/magiccirclepuzzlepiece.pdf`, Accessed: 17 September 2025.

Kamei, H. (2014). Flow e psicologia positiva: estado de fluxo, motivação e alto desempenho. *Goiânia: IBC*.

Knutsen, K. Í. (2021). Visual scripting in game development. `https://www.theseus.fi/bitstream/handle/10024/500439/Knutsen_Krist%F3fer.pdf;jsessionid=F26845C87AAAE320973CCF7AF31024BD?sequence=2`, Accessed: 17 September 2025.

Koivisto, J. and Hamari, J. (2019). The rise of motivational information systems: A review of gamification research. *International journal of information management*, 45:191–210. DOI: https://doi.org/10.1016/j.ijinfomgt.2018.10.013.

Krancher, O. (2020). Agile software development practices and success in outsourced projects: The moderating role of requirements risk. In *Agile Processes in Software Engineering and Extreme Programming: 21st International Conference on Agile Software Development, XP 2020, Copenhagen, Denmark, June 8–12, 2020, Proceedings 21*, pages 56–72. Springer. DOI: http://dx.doi.org/10.1007/978-3-030-49392-9_4.

Kumar, P. (2024). Large language models (llms): survey, technical frameworks, and future challenges. *Artificial Intelligence Review*, 57(10):260. DOI: http://dx.doi.org/10.1007/s10462-024-10888-y.

Lee, W.-T. and Chen, C.-H. (2023). Agile software development and reuse approach with scrum and software product line engineering. *Electronics*, 12(15):3291. DOI: https://doi.org/10.3390/electronics12153291.

Luz, R. B. and Neto, A. (2012). Usando dojos de programação para o ensino de desenvolvimento dirigido por testes. *Anais do Simpósio Brasileiro de Informática na Educação*, 23(1). DOI: https://doi.org/10.5753/cbie.sbie.2012.

Naik, N. and Jenkins, P. (2019). Relax, it's a game: Utilising gamification in learning agile scrum software development. In *2019 IEEE Conference on Games (CoG)*, pages 1–4. IEEE. DOI: https://dx.doi.org/10.1109/CIG.2019.8848104.

Odushegun, L. (2023). Aesthetic semantics: Affect rating of atomic visual web aesthetics for use in affective user experience design. *International Journal of Human-Computer Studies*, 171:102978. DOI: https://doi.org/10.1016/j.ijhcs.2022.102978.

Parsons, D. (2014). Creating game-like activities in agile software engineering education. In *Proceedings of the Australasian Software Engineering Conference, Education Track, Sydney, Australia*.

Pedreira, O., García, F., Brisaboa, N., and Piattini, M. (2015). Gamification in software engineering–a systematic mapping. *Information and software technology*, 57:157–168. DOI: https://doi.org/10.1016/j.infsof.2014.08.007.

Rokis, K. and Kirikova, M. (2022). Challenges of low-code/no-code software development: A literature review. In *International conference on business informatics research*, pages 3–17. Springer. DOI: https://dx.doi.org/10.1007/978-3-031-16947-2_1.

Rossetti, R. D. and Ramos, R. A. O. (2022). A influência

do medo no flow. `https://tede.pucsp.br/handle/handle/29602`, Accessed: 17 September 2025.

Sarinho, V. (2024). Práticas e processos para uma proposta de programação extrema em um modo jogável. In *Anais do XXIII Simpósio Brasileiro de Jogos e Entretenimento Digital*, pages 106–117, Porto Alegre, RS, Brasil. SBC. DOI: https://doi.org/10.5753/sbgames.2024.240818.

Sarinho, V. T. (2019). "bdd assemble!": A paper-based game proposal for behavior driven development design learning. In *Entertainment Computing and Serious Games: First IFIP TC 14 Joint International Conference, ICEC-JCSG 2019, Arequipa, Peru, November 11–15, 2019, Proceedings 1*, pages 431–435. Springer. DOI: https://doi.org/10.1007/978-3-030-34644-$7_4$1.

Sarinho, V. T. (2020). Applying user stories as game elements and interactions in a game of games design proposal. In *Proceedings of the XIX SBGames*, pages 40–46. SBC. `https://www.sbgames.org/proceedings2020/ArtesDesignFull/209314.pdf`, Accessed: 17 September 2025.

Sarinho, V. T. and Apolinário, A. L. (2009). A generative programming approach for game development. In *2009 VIII Brazilian Symposium on Games and Digital Entertainment*, pages 83–92. IEEE. DOI: https://dx.doi.org/10.1109/SBGAMES.2009.18.

Souza Teixeira, E. A. and Fonseca Ramos, F. (2014). Interações e literacias: notas sobre o design de interfaces e a experiência de uso. *Ciência da Informação*, 43(3).

Spil, T. A. and Bruinsma, G. (2016). Designing serious games with the game of games. In *Proceedings of the European Conference on Games-based Learning*, pages 634–643.

Sutherland, J. and Sutherland, J. (2014). *Scrum: the art of doing twice the work in half the time*. Crown Currency.

Vincur, J., Konopka, M., Tvarozek, J., Hoang, M., and Navrat, P. (2017). Cubely: virtual reality block-based programming environment. In *Proceedings of the 23rd ACM symposium on virtual reality software and technology*, pages 1–2. DOI: http://dx.doi.org/10.1145/3139131.3141785.

Williams, L. (2010). Agile software development methodologies and practices. In *Advances in computers*, volume 80, pages 1–44. Elsevier. DOI: https://doi.org/10.1016/S0065-2458(10)80001-4.

Wood, D. F. (2003). Problem based learning. *Bmj*, 326(7384):328–330. DOI: https://dx.doi.org/10.1136/bmj.39546.716053.80.