EXPLORING PRODUCT LINE CONCEPTS IN VIDEOGAME BUILDING

Diego Cardoso Borda Castro

Exame de Qualificação de Doutorado apresentado ao Programa de Pós-graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Advisor: Cláudia Maria Lima Werner

Rio de Janeiro
Novembro de 2022

Abstract of Qualifying Exam presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doutor of Science (D.Sc.)

EXPLORING PRODUCT LINE CONCEPTS IN VIDEOGAME BUILDING

Diego Cardoso Borda Castro

November/2022

Advisor: Cláudia Maria Lima Werner

Department: Engenharia de Sistemas e Computação

The gaming industry is one of the world's most influential, attracting fans of all ages, genres, and tastes. However, game development can be a lengthy process, with some titles taking years to reach store shelves. With such a large community of enthusiasts, some consumers cannot wait this long time for the game to be released. As a result, they end up creating their own versions of the game, which is known as mod. This term refers to modifying a game through a player's expression. However, another term that is very similar to this is the modifier term, which seeks to produce games by modifying the original game. A study was conducted, and it was possible to perceive their strong similarity between the term modifiers and the Reuse of Opportunistic Software, as well as some difficulties in this process, with a focus on the lack of tools that support the development and evolution of videogames. A study conducted revealed a clear connection between the term modifiers and the Reuse of Opportunistic Software as well as some difficulties in this process, including the lack of suitable tools and the difficulty in the development and evolution of the game. Based on this, the current work conducted a study on Software Reuse and game / mods development and proposes the EngageSPL platform, having características such as: feature tree, evolution by through modifiers and development pattern verification. With the use of this platform, it is expected that the development of mods will be improved, saving time, money, and effort.

Resumo do Exame de Qualificação apresentado à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doctor em Ciências (D.Sc.)

EXPLORANDO CONCEITOS DE LINHA DE PRODUTO NA CONSTRUÇÃO DE JOGOS

Diego Cardoso Borda Castro

Novembro/2022

Orientador: Cláudia Maria Lima Werner

Programa: Systems Engineering and Computer Science

A indústria de jogos é uma das mais influentes do mundo, atraindo entusiastas de todas as idades, gêneros e gostos. No entanto, o processo de desenvolvimento de um jogo pode ser muito demorado, tendo títulos que demoram anos para chegar nas prateleiras. Com uma comunidade de entusiastas tão grande, alguns consumidores não conseguem esperar tanto tempo até o lançamento do jogo. Devido a isso, acabam criando suas próprias versões do jogo, sendo essa atividade denominada de mod. Esse termo se refere a modifião de um jogador. No entanto, outro termo bem semelhante a este o termo modificador que busca produzir jogos por meio de alter- de um estudo realizado, foi pos perceber sua grande semelhante do termo modificadores e da Reutião de Software oportunista, além de algumas dificuldades nesse processo, com destaque para a falta de ferramentas que apoiem no desenvolvimento e Reutilização de Software e desenvolvimento de videogames e propoe a plataforma EngageSPL que integra os conceitos de Linha de Produto, do framework MDA e de videogames, possuindo funcionalidades, como: o de caracter ísticas, ão por meio de mods e verificação de padrão de desenvolvimento. Espera-se que com o uso dessa plataforma o desenvolvimento de mods seja melhorado, diminuindo o tempo de desenvolvimento, dinheiro e esforço.

Contents

List of Figures

List of Tables

List of Abbreviations

CBD ..... COMPONENT BASED DEVELOPMENT

CIM ..... COMPUTATIONAL INDEPENDENT MODEL

DSPL ..... DYNAMIC SOFTWARE PRODUCT LINE

ENGAGESPL ..... ENGINE FOR GAME GENERATION THROUGH SOFTWARE PRODUCT LINE

MDD ..... MODEL DRIVEN DEVELOPMENT

PIM ..... PLATFORM INDEPENDENT MODEL

PSM ..... PLATFORM SPECIFIC MODEL

OMG ..... OBJECT MANAGEMENT GROUP

SE ..... SOFTWARE ENGINEERING

SR ..... SOFTWARE REUSE

SLR ..... SYSTEMATIC LITERATURE REVIEW

MR ..... MULTIVOCAL REVIEW

SPL ..... SOFTWARE PRODUCT LINE

TAM ..... TECHNOLOGY ACCEPTANCE MODEL

Chapter 1

Introduction

This chapter aims to present the context, motivation, and problems addressed by this work. In addition, the objectives, and the methodology adopted to achieve them are also presented, as well as the organization of the text.

1.1 Motivation and Context

According to experts, there are about 2.2 billion gamers worldwide (PASHKOV, 2021). Games have grown to be one of the most popular forms of entertainment, attracting fans of all genders, ages, and tastes, as well as being a showcase of the industrial environment, amassing billions over the years, being compared even with the movie industry. To have a representation of this growth, in 2017, League of Legends collected 2.1 billion dollars in just one year (JARRETT, 2021); 400 million dollars were collected in 24 hours at the launch of the game Call of Duty: Modern Warfare 3 in 2011 (MARCHAND and HENNIG-THURAU, 2013) and this was over 11 years ago. The game industry is still growing. In 2020, 2021 and 2022, as a result of the quarantine imposed by COVID-19, the video game industry experienced massive growth, raising around $159.3, $175.8 and $196 billion, which is a steady annual growth rate of 9.3, 10.2 and 11.4 percent since 2019 (PASHKOV, 2021; WIDJMAN, 2021). The gaming industry has even been compared to the movie industry due to the amount of investments and income generated throughout the year, being estimated that it may even be higher one day. Experts predict that the gaming industry will reach $260 billion annually by 2025, just three years from now (BEATTIE, 2020).

Regardless of the amount of money generated/invested and the size of the user community, the process of developing a game, from planning to release, may be rather expensive, time-consuming, and involve multiple specialists, with some games taking years to complete. It's worth noting that some of these games take years to develop and, even then, they either never launch or are delivered with several

problems, resulting in significant expenditures for the developer (BILIŃSKA *et al.*, 2020). Along with the issues mentioned above, there are several additional factors that might effect the creation of a game, such as the difficulty of producing something new/innovative, meeting client expectations, and ensuring the game's quality (BILIŃSKA *et al.*, 2020). Cyberpunk 2077 is a possible example of a game that did not follow the ideal flow of success. Despite seven years of development, hundreds of millions of dollars invested, multiple launch delays, and high expectations, numerous errors were discovered in the release, prompting the company to fix the bugs and compensate some users (POLITOWSKI *et al.*, 2021).

However, with such a big community of followers, some of these users cannot wait such long time for a game to be released or become frustrated with the game's numerous bugs, resulting in discontent and anxiety for some of them. As a result, some people modify current games in order to develop their own versions (UNGER, 2012). This process of using already created games to generate new ones is known as mod. When discussing mods, one of the most frequently mentioned topics is the use of modifiers,that can be understood as changes made to games in order to expand the same (NETO and TAYLOR; UNGER, 2012).

Modifiers is very similar to opportunistic Software Reuse (SR). SR is the process of reusing existing systems rather than constructing new ones. It is the process of using existing software artifacts and expertise in order to create something new, saving time and money on development (KRUEGER, 1992). Keeping this in mind, a literature review was carried out to identify ways of building video games and which SR approaches were being used for game development and evolution. As described in chapter 3, four methods were identified: clone-and-own, software components, Model-Driven-Development (MDD), and Software Product Line (SPL). The first method is still the most used today, being very similar to opportunistic reuse and not recommended for large projects due to not having a development systematization. Componentization has also been used, being available in the main known game engines. The last two are still in the testing phase, with few platforms to help with this kind of development. In particular, for the SPL approach, only one development platform was identified, and this was specific to a type of game. It is worth remembering that these platforms are focused on game development and not specifically for modifiers, not having specific features for video game evolution.

One of the main points highlighted by the review was the lack of appropriate tools for video game evolution. Some companies even provide SDKs that allow the modification of some parts of the game or the creation of video games, but often in a very limited way, and there are still few companies that provide this opportunity to allow minor changes in the game. Due to this limited availability of tools to facilitate game extension, a user or programmer who wishes to create a extended game must

comprehend the source code, which requires a significant amount of programming or even starting from scratch. Due to the aforementioned efforts, developing extended video game can be a challenging, time-consuming, and expensive hobby / process.

The study carried out in this work also demonstrated the advantages, disadvantages, contexts of use, and motivation for using each of the identified approaches. On the basis of this information, it was determined that the SPL approach would be the most recommended one for game evolution and video game development due to its numerous benefits, such as the separation of software features, the easy visualization of the product building tree, and the rapid evolution of existing features, which permits the creation of N versions of a single game.

Although the software product line alone can facilitate in the evolution of video games, it does not provide tools for the evolution of existing features, only allowing to choose or remove existing features. As previously mentioned, a extended game is a game that received a modifier that change some of its characteristics (NETO and TAYLOR; UNGER, 2012). Therefore, thinking of new ways to evolve / derive video games, this concept of modifiers can be utilized associated with the product line to add this functionality to change existing characteristics. In this way, the SPL would be responsible for separating the game's features into a tree model (feature tree) and modifiers would have the role of modifying the leaves of these trees that represent the features of the games.

The elemental tetrad (SCHELL, 2008) will be the last new concept added to the video game evolution concept. There are numerous ways to represent a game and its features, and with these frameworks, any game can be characterized according to its mechanics, aesthetics tecnology and story. The proposal is to represent the game using tetrad, and each component of this framework having its own feature tree. Thus, each game would have three trees, and any mod's mechanics, aesthetics tecnology and story could evolve / derive based on the SPL and mutator concepts described above.

According to what has been discussed thus far, the purpose of this research is to combine SPL, modifiers and tetrad in order to propose a video game development platform that reduces players' anxiety and dissatisfaction by assisting in the evolution of video games. In this way, a game would only need to be developed once, and N versions of it could be generated quickly, easily, and at low cost. That way, if a game were built by the platform, any user who wanted to create their version of the game could have this development facilitated.

This work originated from a dissertation (CASTRO, 2020) in which the goal was to teach SR through serious games. The production of games based on existing games was investigated in this dissertation, as well as the adaption of games for diverse settings, which is very similar to opportunistic reuse or what is already

being done in the games community. The dissertation examined the use of games to teach SR. However, in this case, the intention has changed, having SR assisting in the development of games this time.

Continuing with the video game constructed in the dissertation, its design was revised, resulting in the creation of a game product line that demonstrates how the game expanded until the mod. From this manual product line, this idea was improved providing two new games that simulated an SPL, following the idea of building the game development platform. These games were evaluated and showed great potential to aid in game development, thus enabling the creation of a platform to assist in the development of games through product lines. The platform was called EngageSPL and aims to enable the development of a game once and create N versions of it using modifiers and SPL.

1.2 Objective

To aid in video games extension the current research recommends the use of SPL combined with the tetrad framework and the modifier concepts to shorten development time, boost the potential of extending video games, lower investment costs and simplify the process.

With the general objective of improving the video game development process, it is proposed to build a vide game development platform through SPL, modifiers and tetrad techniques with the aim of obtaining the advantages offered by these approaches, such as cost reduction, more quality, increased possibility of expanding the software and reduced development time. This general objective can be decomposed down into specific goals:

- Characterize state-of-the-art on the derivation of games and modifiers.
- Characterize the state of the art of using software reuse approaches for game development.
- Create simple games to validate game development through Software Reuse approaches.
- Develop a platform to support game development through Product Line, modifiers and tetrad. From the platform it will be possible to create and develop a game once and generate N versions from it through the use of modifiers and SPL. **(To be performed).**
- Conduct a platform evaluation study to validate the ease and agility of developing vide games through product lines. **(To be performed).**

1.3 Methodology

The technique used in this research was inspired by the Design Science Research (DSR) model (LACERDA *et al.*, 2013), but was adjusted according to the work need. Each stage contributed to and strengthened the work proposal's construction. The viability study provided a general context, demonstrating that the theme was still important and that the process of extending video games still happened in an ad-hoc way, but that it could be improved and systematized using reuse approaches. The literature review corroborated by demonstrating which reuse techniques were already in use and which would be the best for creating video games and expanding / deriving games, laying the groundwork for the plan to create a game development platform that could be used across product lines, modifiers and the tetrad framework. Finally, prototyping demonstrated that the platform proposal was sound and that further development was possible.

Figure 1.1 presents the methodology used, which is divided into three major parts: Awareness and Search that are initial activities and artifacts which demonstrated the initial ideas of the work; Validation of ideas, information, and artifacts that were generated from the Search activities; Proposal, development of the work proposal. Some artifacts and links have been omitted from the image to make it easier to understand.

The method stated in Figure 1.1 may be interpreted in two ways: each line details how each activity was designed, while each column details all activities performed (first column), their associated artifacts (second column), and their associated objectives (third column). Orange and green colors represent exploratory and conclusive efforts, respectively. Exploratory activities may be defined as those that involve the validation or analysis of early data. Conclusive activities are those that seek to explore deeper into the work overarching issue, which is to investigate reuse strategies for improving the game development process. Each of the activities presented in Figure 1.1 is described in more detail below.

- **Awareness and Search:** Seeks to identify the problem to be solved. Performs two reviews to understand what has already been produced within the area of games and Software Reuse.
  - **Problem Awareness:** Elaborate an initial study to have a first contact with the researched area. Seeking to understand how video games are being extended currently.

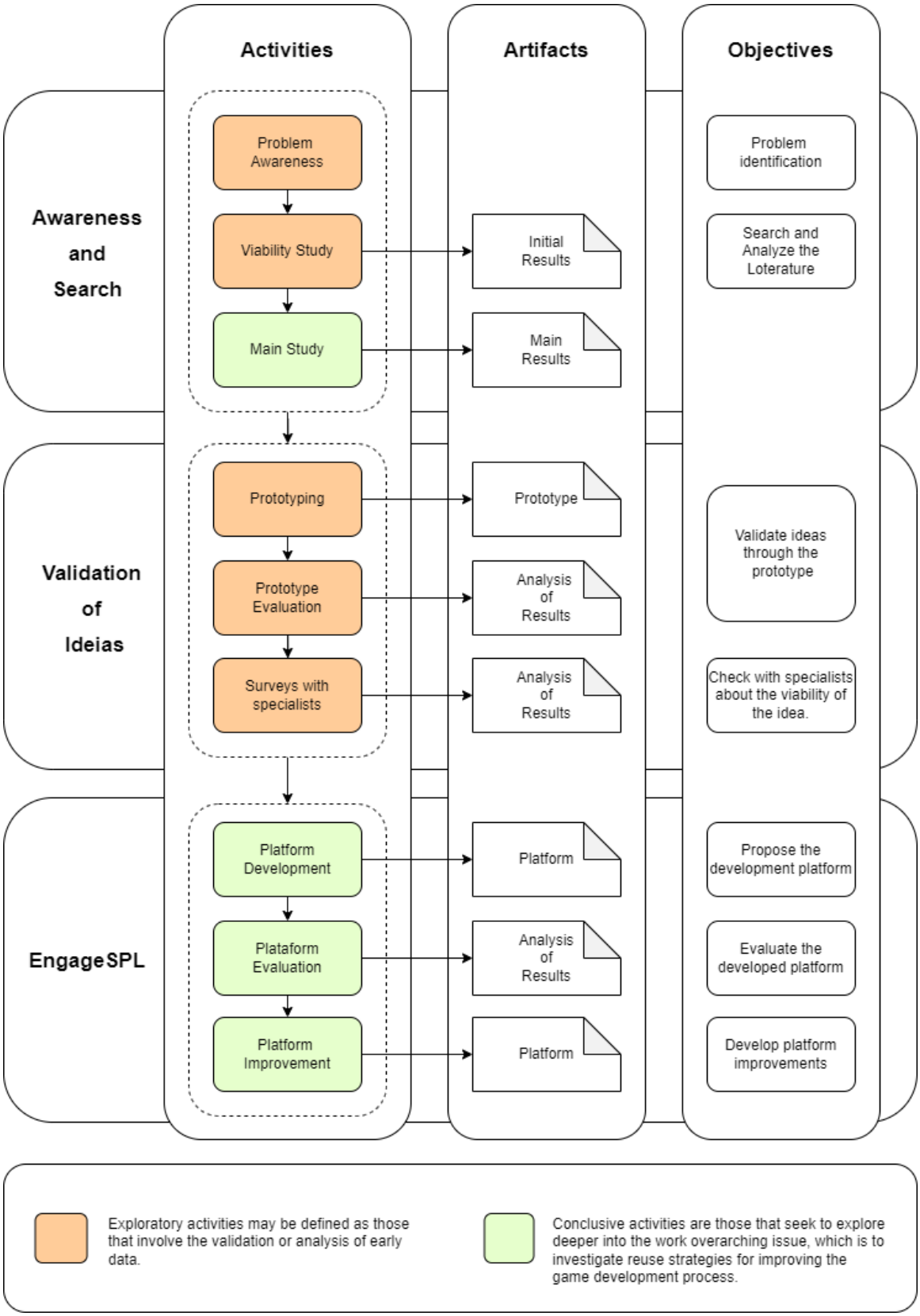

Figure 1.1: Research methodology, adapted from (LACERDA *et al.*, 2013)

    - **Objective:** Identify the problem to be solved with the proposal. Identify what problems exist in video games extension.
  - **Viability Study:** Conduct a preliminary study to find out what has already been produced on game modification development.
    - **Artifact:** Analysis of the results of the preliminary review.
    - **Objectives:** Define state of the art and provide practice on game modification and development. The viability for carrying out the study.
  - **Main Study:** Develop a study to find out what has already been produced on game development and Software Reuse.
    - **Artifact:** Analysis of the results of the main review.
    - **Objectives:** Define state of the art and provide practice on using SR for game development.

- **Validation of ideas:** Activities to generate a practical basis.
  - **Prototyping:** Develop one (or more) game(s) based on previous studies in order to validate SR approaches to game development.
    - **Artifact:** Games that was developed.
    - **Objectives:** Validate some SR approaches to the game development process.
  - **Prototype Evaluation:** Carry out a viability study to evaluate the game created in the previous phase.
    - **Artifact:** Game evaluation.
    - **Objectives:** Evaluate the games from a development perspective. Evaluate whether the developed games indicate ease of construction and whether such a method may aid in game development.
  - **Surveys with specialists:** Check with specialists about the validity / viability of the idea.
    - **Artifact:** Analysis of Survey results.
    - **Objectives:** Determine from the opinions of the specialists whether the platform's development is valid, a good idea, and viable.

- **Proposal:** Based on the information collected in the previous activities, it aims to find a solution to the problem demonstrated.
  - **Platform Development:** Create the EngageSPL platform, which aims to support in the creation of games via Product Line.

    * **Artifact:** Platform that was developed.
    * **Objectives:** Develop the platform that aims to solve the problem found in the research phase. Develop a game development platform by selecting a reuse approach.

  - **Platform Evaluation:** Carry out a viability study to evaluate the game platform in the previous phase.
    * **Artifact:** Platform evaluation.
    * **Objectives:** Evaluate the developed platform to verify if it may help in the development of games more quickly and easily.

  - **Platform Improvement:** Improve EngageSPL Platform.
    * **Artifact:** Platform with identified improvements.
    * **Objectives:** Develop the improvements found in the platform evaluation.

1.4 Text organization

This work is organized into five chapters. In this chapter, the context, as well as the motivation and problem of this research was presented.

Chapter 2 presents the theoretical foundation on some subjects that help in understanding the other chapters. Topics such as games, mods, modifiers, software reuse, product lines, model-based development, and software components will be presented.

Chapter 3 presents the two literature review studies that supported this work. This chapter demonstrates the state of the art about mods/modifiers and game development based on Software Reuse approaches.

Chapter 4 presents a discussion about the problem to be addressed in this work, demonstrating what is intended to be built to solve the problem. Demonstrates the steps taken to validate the idea for solving the problem.

Chapter 5 concludes the qualifying exam by proposing a schedule of future activities to carry out the thesis proposal.

Chapter 2

Theoretical foundation

This chapter aims to provide the theoretical foundation essential to comprehending the research conducted, which may be unfamiliar to the reader. In this sense, some basic concepts about mutators, modifications, video games, and Software Reuse are described in the following.

2.1 Games

Numerous definitions of games are found in the literature. The majority of them, however, center around the following definition: Games may be characterized as activities that make use of an abstract environment in which decisions, actions, and rules are established with the purpose of achieving a leisure activity, such as fun or enjoyment (EHRMANN *et al.*, 1968; HUNICKE *et al.*, 2004).

Numerous characteristics may be found in games. The complexity of a game can increase in proportion to the number of elements. As a result, methods for arranging game concepts are required. There are several ways to arrange ideas for creating a game, including flow models (DORMANS, 2011), gameplay (GUARDIOLA, 2016), and features (XEXÉO *et al.*, 2013). Elemental tetrad (SCHELL, 2008) is a well-known model in the gaming industry. It divides the properties of games into four categories, which are (SCHELL, 2008):

- **Mechanics:** can be interpreted as the rules and activities that may occur during the course of the game.
- **Second level mechanics:** although the second level mechanics are not part of the elemental Tetrad, they will be added in this work to describe mechanics that are generated from the combination of primary mechanics. This addition was made to bring more dynamism to the proposed modification of the games to be built (SCHELL, 2008).

- **Story:** describe the narrative aspect of the game. In this work the story will be modified through the direct modification of the aesthetics.
- **Aesthetics:** are the emotions experienced by the player. Sensation, Fantasy, Narrative, Challenge, Fellowship, Discovery, Expression, and Submission are the most common emotions.
- **Technology:** refers to the tools and the systems used to implement the gameplay. In this work, this category will not be modified due to the technology used for all games being the same, always using Unity for mobile game development.

.

Along with methods for organizing game concepts, there are various cycles of game creation described in the literature; nonetheless, they may be classified into four primary stages (RAMADAN and WIDYANI, 2013). Each of them is highlighted as follows:

- **Pre-production:** aims to define and improve the game's original concepts; this is the stage during which documentation, such as the Game Design Document (GDD), concept art, and game design, is created. Certain procedures begin with a phase called Pitch, which is responsible for the game's idea and basic design.
- **Production:** some processes divide this stage into development and refinement. However, it may be viewed as a continuous cycle process centered on the creation of assets and source code. Finally, the entire technique is validated internally.
- **Testing:** intends to evaluate the usability, playability, and balancing characteristics of the game. This process is typically followed by two releases (alpha and beta), depending on the game's integrity.
- **Release:** final stage of production of the game, it is ready to be released to the public. The release process comprises product launch, project documentation, and game maintenance and expansion planning.

Figure 2.1 is a simplified flow chart illustrating the many stages that might occur during game development. The dotted steps are optional and may be skipped.

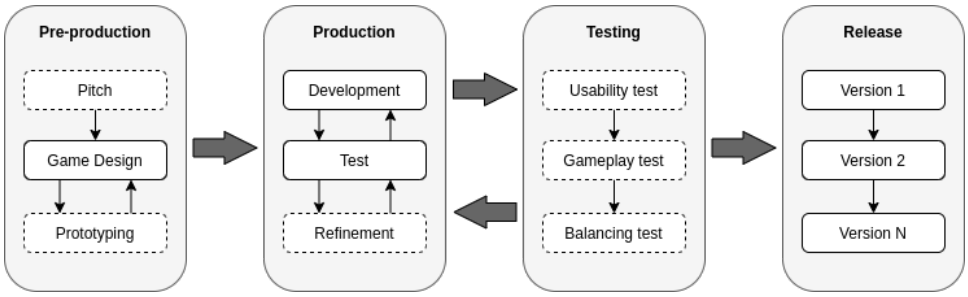

Figure 2.1: Development flow.

As a result of the procedure outlined above, it is reasonable to deduce that developing games may be quite difficult and time-consuming. As a result, some members of the community utilize already created games to build new ones. This is referred to as the Mods method (MCARTHUR and TEATHER, 2015).

2.1.1 Mods and Modifiers

What qualifies a game as "new"? Recognizing what makes a new game is not as simple as it may appear. By defining the games outlined in the preceding section, it is possible to observe various aspects that distinguish them, such as new rules, actions, and player decisions. It is not essential to edit each of these items in order to create a new game. Modification occurs in phases. When these modifications are done by members of the community, they are referred to as mods. Modifications are described as the process of creating tweaks [1], additions [2], or adaptations to a game (SCACCHI, 2011b). In general, mods are modified games that work differently from the original version.

Historically, the term mod derives from hacking, which occurs when gamers alter game code without the permission of the developer. This act was motivated by a number of factors. Among the most notable ones are the following: providing new experiences, arousing curiosity, and financial considerations, to name a few (BILIŃSKA *et al.*, 2020). Mods are an enticing and durable form of player and content creation that have existed for more than a decade. This technique has become so popular that some firms have even developed platforms that allow for the modification of certain games. Another manifestation of this renown is the presence of instances in which the modified game achieved greater popularity than the original one, as it was the case with the Counter-Striker mod, a game derived from Half-life. In a nutshell, they are digital artifacts created by players when fiddling with their preferred games. Video games extesions are made using modifiers, which may be

[1]Tweaks: minor modifications that do not change the main rules of the game (SCACCHI, 2011a,b).

[2]Additions: works as an extension of the game, like adding new maps or objects (SCACCHI, 2011a,b).

thought of as changes to games (NETO and TAYLOR; UNGER, 2012). There are several granularities of mutators/modifiers that may be applied to games, ranging from modest modifications to the game's map to entire rebuilds (SOTAMAA, 2010). In Chapter 3, each mod will be described in more detail.

As previously stated, mods are a form of player expression; however, within the field of game development, another term very similar to it is modifiers, which can be understood as changes made to games, as well as mods, but this time only aimed at extending the original game by adding new features and not as a form of expression, this being the focus of this work.

2.2 Software Reuse

Reuse is effectively used in a variety of industries, including manufacturing, vehicles, and electronics. The term Software Reuse (SR) was coined in 1968 at a NATO conference (RANDELL, 1979) and is one of the disciplines of Software Engineering (SE). It is defined as the process of developing systems from one or more existing ones rather than starting from scratch; that is, it is the process of utilizing existing software artifacts and knowledge to create something new (KRUEGER, 1992). NIU et al. define reuse as a simple vision of "not reinventing the wheel".

SR concerns crosscut several Knowledge Areas (KAs) of the Software Engineering Body of Knowledge (SWEBOK) (BOURQUE and FAIRLEY, 2014), namely the Software Design KAs (e.g., component-based design, software product lines), Software Construction KAs (e.g., construction for reuse and construction with reuse), Software Testing KAs (e.g., test reuse and test patterns), and Software Engineering Process KAs (e.g., reuse processes).

The primary objective of this research is to create games using SR techniques. On the basis of the SWEBOK (BOURQUE and FAIRLEY, 2014) KAs, it is possible to identify certain areas that may be used for this purpose, including Construction for and with Reuse, Software Product Lines, Component-Based Design, and Model-Driven Development. The first two elements stated are crucial to the Reuse process as a whole. For instance, when you construct a component for reuse and then use it, you are developing with reuse.

2.2.1 Components

A software component may be thought of as a self-contained, replaceable bit of code that performs a certain purpose and is reused when creating a new program. It is anticipated that by implementing this approach, several benefits would be realized, including function unification, encapsulation, increased code quality, and more agile

development (SAMETINGER, 1997).

Software development through components can be understood as a series of steps for selecting, composing, modifying, and using artifacts. This type of development can be divided into two stages: development for components and development of components. Development with components can be understood as the process of building software from pre-produced components. Development for components can be understood as building components for future use (BLOIS, 2006). Figure 2.2 demonstrates the development flow with components.

For some time, the gaming community has made use of the concept of software components. Nowadays, libraries containing a variety of pre-produced components are available to assist developers in creating games more efficiently. The game engines themselves already provide libraries for component availability, with the unity asset store serving as one example (KYAW, 2013).

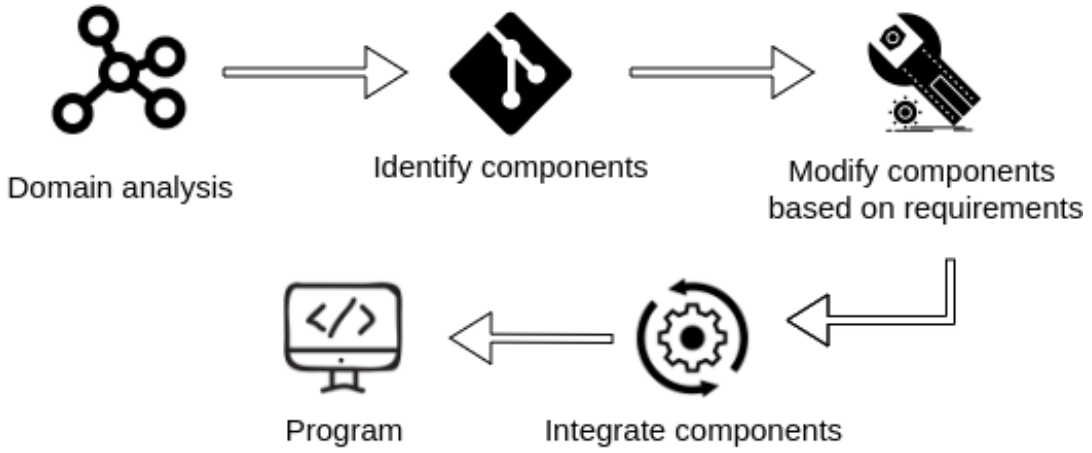

Figure 2.2: Component-based development illustration.

2.2.2 Software Product line

The term Software Product Line (SPL) refers to a collection of strategies, techniques, and tools for the methodical development of comparable systems that have a common core but exhibit distinct characteristics. The utilization of these subjects is predicted to result in a decrease in development time, simpler maintenance and evolution of systems, enhanced programmers satisfaction, and a higher quality of code (KRUEGER, 1992).

SPL divides its approach into two stages: domain engineering, which involves the creation of common assets, and application engineering, which involves the reuse of common elements and the addition of unique elements. SPL is distinct from other forms of reuse in that it contrasts predictive and opportunistic methods. Instead of storing generic software components in a library in the expectation of reusing them, SPL requires the development of software artifacts (i.e., assets) only when their reuse in one or more products is anticipated (KRUEGER, 1992).

Finally, a product line is composed of four sorts of elements: **required** that are present in all applications and constituting the heart of SLP (demonstrated by filled balls); **alternative** that are restrictive characteristics; an application may have one or more of these features (demonstrated by unfilled circle); **optional** that are features that specific applications may or may not have (demonstrated by unfilled balls); and **exclusions** that when only one of these characteristics can be used (demonstrated by filled circle). Based on these elements, it is possible to derive each of the game's features from the product line, thus building a feature tree for each game (SCACCHI, 2011a,b). Figure 2.4 demonstrates these elements based on Feature Oriented Domain Analysis (FODA) (KANG *et al.*, 1990).

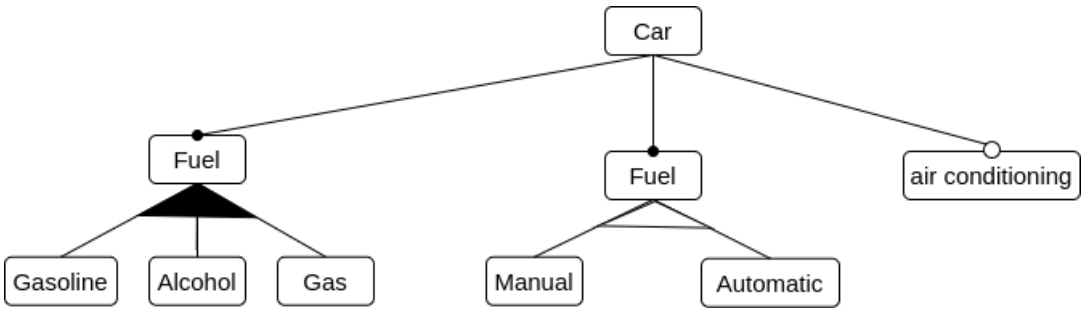

Figure 2.3: Feature Oriented Domain illustration (SCACCHI, 2011a,b).

2.2.3 Model Driven Development

The OMG (Object Management Group) has created standards that enable the establishment of platform-independent development methodologies. This set of standards is referred to as Model Driven Development (MDD) and is intended to decouple feature design from implementation specification. MDD-based techniques enable the creation of software by modeling and applying models of its implementations.

The purpose of MDD is to enable developers to focus exclusively on the application's business requirements, rather than on the platform on which they will be executed (MAIA *et al.*, 2007).

MDD's basic premise is that modeling languages should be used as programming languages, not merely as design languages. As a result, software models cease to be the only documentation artifacts and instead become critical components of the software development process. Prior to MDD, models were created to aid in communication between developers and drafters on a software development project. With MDD, models must now be more precise in order to become a natural part of the software development process. That is, these models serve as input artifacts for the system's construction via transformation execution.

In general, MDD is built on updated and described models (a simplified representation of some concept). It is based on two models: the Platform Independent Model (PIM), which provides an overview of the system independent of the platform, and the Platform Specific Model (PSM) that is a view of the system from a specific perspective for a platform or technology. MDD is a technique that entails developing a PIM using a modeling language and then transforming it to generate a PSM, which generates the software implementation in the selected programming language (MAIA *et al.*, 2007). Figure 2.4 demonstrates the process of building software through MDD.

Comparing the MDD model and the development of Mods, it is possible to propose an integration between them. Mods, in general, are games that receive transformations and generate adaptations of the original games. Games can be decomposed according to their characteristics as mentioned in Section 2.1 and these can be present in the PIM model, thus creating an abstraction of the game. As mentioned, there are several types of modifiers that can be applied to games. These modifiers could be the transformations that are involved in the PIM model. After a series of transformations and the use of a game-specific programming language, a PSM would be generated that could be considered as the final mod.

2.3 Final considerations

This Chapter intended to bring out the main points that the reader needs to know before reading the following chapters. The whole context of games, mods, and Software Reuse techniques has been illustrated so that the rest of the work can be read comprehensively.

It is noticeable from this contextualization chapter that the game production

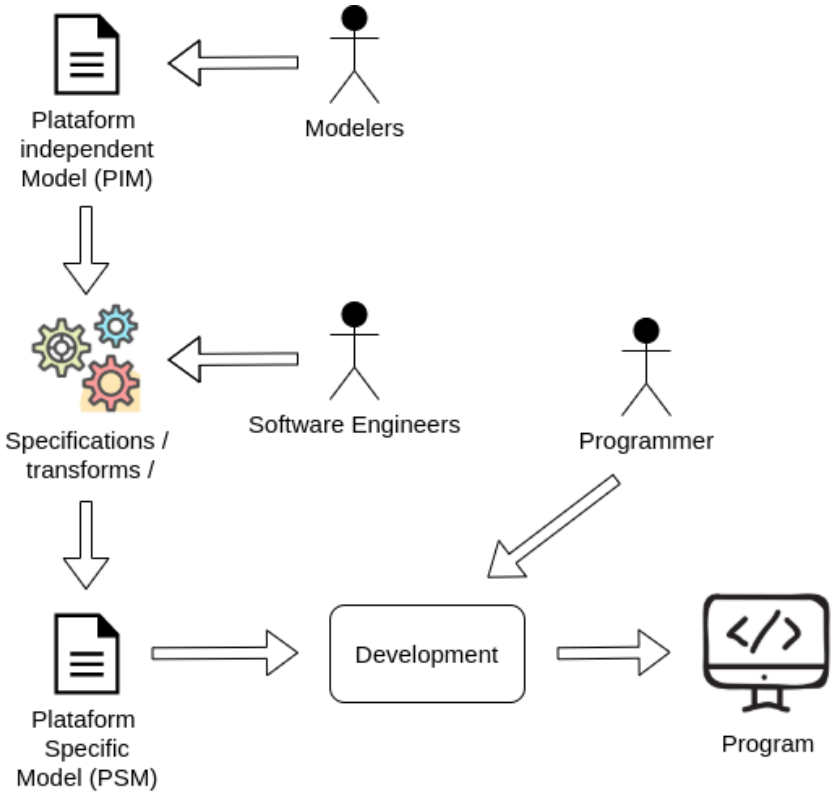

Figure 2.4: Model Driven Development illustration (WADA and SUZUKI, 2005).

process is extremely complex, time-consuming, costly and requires numerous steps and expertise. As a result, many people in the gaming community end up creating game mods out of frustration and dissatisfaction. The motivation, methods of development, and types of mods will be discussed in greater depth in the following chapters.

The definition of the tetrad framework, which is referenced in other parts of the text and even as a foundation for the final proposal of the work, is another piece of information that requires the reader's attention.

Finally, this chapter also demonstrates some SR techniques in a little more detail. These strategies will be explored in the next chapters of the work, according to the scope of game production.

Chapter 3

Literature Review

This chapter seeks to show the study that was done in this research in order to find out what has already been established in terms of video extension / derivation using software reuse techniques. A key point to remember for a thorough understanding of this work is that little information on game extension via modifiers was found in the literature. Although game companies are thought to use game changing techniques as well as Software Reuse, little evidence has been discovered. As a result of the similarity of the terms, mods were researched in the literature. Thus, two searches were made: one on general mod construction and the other on game development using reuse techniques.

It is important to note that the purpose of these two studies was to provide information for the final proposal of this work. It was attempted to conduct a single study on mod derivation; however, few relevant works were discovered and therefore the research was divided in two parts.

3.1 Search protocol

The study described in this chapter took place between October 2021 and March 2022 with the goal of identifying previous work in the areas of games, mutators, and reuse.

Two distinct search strategies were utilized in this study (one for each review), referred to as Systematic Literature Review (SLR) and Multivocal Review (MR). The Viability Study employed the MR technique in order to gather as much information as possible and to take advantage of the gaming community's huge blog and website presence. SLR was used for the main review of the work (review of reuse approaches to game development), and this choice was made because, in the viability study, a search in gray literature was conducted, and the result was not highly relevant in comparison to the effort spent, returning information that had already been discovered in the white literature.

- **SLR:** is a systematic methodological review of research that investigates and categorizes studies in a specific field of study and presents an overview of a certain subject systematically (i.e., using an organized and repeatable process or procedure) (KITCHENHAM *et al.*, 2009).

- **MR:** is a more complete examination of the literature that aims to elicit as much information as possible about a specific subject; hence, it incorporates data from both white (academic papers, books, etc) and gray (blogs, websites, videos, etc) sources. This strategy is typically utilized when there is substantial community support for the study subject and it is necessary to verify practical knowledge on a particular subject (GAROUSI *et al.*, 2019). A MR may be separated into two stages: the first stage involves the search for academic knowledge (in this case, an SLR was used), and the second stage involves the search for gray literature.

All revisions carried out in this work followed the stages of the protocol proposed by (KITCHENHAM *et al.*, 2009). The search string was executed on the main search engines, Scopus[3], ScienceDirect3[4], IEEEXplore[5], and El Compendex[6], as recommended by (CUSKER, 2013) and (KITCHENHAM *et al.*, 2009).

To make this study possible, a core search string was constructed utilizing the PICOC structure (Population, Intervention, Comparison, Outcome, and Context) at four levels (ARAÚJO *et al.*, 2022; PETTICREW and ROBERTS, 2008). The search string was constructed by combining related domain-specific keywords using the logical operator "OR" and fields using the logical operator "AND". This string was utilized throughout the studies. To validate the search string, two control papers were used to create and run the string in the Scopus database, which was the first database where the string was applied. This validation technique aims to ensure the search string's quality by ensuring that it returns relevant articles and author knowledge (MORAES and SOUZA, 2011).

According to (MOTTA *et al.*, 2016) and (MATALONGA *et al.*, 2017), snowballing processes can compensate for the absence of other search engines and supplement the approach by doing research via the references and citations of the papers. Therefore, to minimize the loss of some papers and increase the search range, the forward[7] and backward[8] (one-level) snowballing procedure was used, which checks

[3]https://www.scopus.com

[4]https://www.sciencedirect.com

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

[6]https://www.engineeringvillage.com/home.url

[7]Snowballing Forward: refers to the identification of new papers based on the works that referenced the paper that was analyzed (WOHLIN, 2014)

[8]Snowballing Backward: refers to the identification of new papers based on the works that were

the references and citations of articles seeking relevance (WOHLIN, 2014). The procedures, inclusion and exclusion criteria and quality criteria will be described below. The research questions of each of the reviews will be described in the following sections.

Implementation procedure

1. Execute the search string;
   (a) For searches in gray literature, it was searched for each search string up to page 10 of google. The search strings were formed by combining the keywords of population and intervention;
2. Apply the inclusion / exclusion criteria based on the title;
3. Apply the inclusion / exclusion criteria based on the abstract;
4. Apply the inclusion / exclusion criteria based on the full text;
5. Apply the quality criteria;
6. Apply snowballing backward; and
7. Apply snowballing forward
   (a) For searches in gray literature, the snowballing was performed on site references, on links contained within the site.

Inclusion criteria

1. Viability Study: The article must be in the context of Mods;
2. Main Study: The article must be in the context of Games and Software Reuse;
3. The paper must provide data to answer at least one of the research questions;
4. The paper must be written in English.

Exclusion Criteria

1. Conference call;
2. Studies that can not be fully accessed;
3. Studies that are not in the area of Computer Science or Engineering.

Quality Criteria

referenced in the paper that was analyzed (WOHLIN, 2014)

1. Is the publishing organization reputable?
2. Has the author published another work in the area?
3. Does the author have expertise in the area?
4. Is the article clear?
5. Are the references documented?
6. Does this enrich the research?

3.2 Viability Study

The research method demonstrated in this section was divided into two stages. The first intended to elicit information from white literature (academic papers), whereas the second functioned as a complementary study, eliciting information from gray literature (websites, blogs, etc) (GAROUSI *et al.*, 2019). The protocol used in this research was demonstrated in the previous section. In the following, the research questions will be demonstrated.

Table 3.1: Search string of mutator and games.

P	*Game*
I	Mutator, variant, mods, modification, conversion, add-on, tweak, modding
C	Not applicable
O	Tools, approach*, method*, ideas, framework*, mechanics, interpretation*
C	Creation, production, development, elaboration, generation, practice*
TITLE-ABS-KEY ((*game*) AND (mutator OR variant OR mods OR modification OR conversion OR add-on OR tweak OR modding) AND (tools OR approach* OR method* OR ideas OR framework* OR mechanics OR interpretation*) AND (creation OR production OR development OR elaboration OR generation OR practice)) AND (LIMIT-TO (SUBJAREA, "COMP") OR LIMIT-TO (SUBJAREA , "ENGI"))	
Control papers	1 - Modding as part of game culture UNGER (2012)
	2 - Serious mods: A case for modding in serious games pedagogy MCARTHUR and TEATHER (2015)

Research Questions

- **Q1:** What modifiers are used to create games from others?
- **Q2:** What characteristics are needed to derive a game?
- **Q3:** What are the advantages and difficulties of creating games from others?
- **Q4:** What tools strategy or frameworks support these changes?

The first stage returned a total of 923 papers. When the publications were examined using the inclusion and exclusion criteria, this number was reduced to 14. From these studies, the snowballing process was carried out, and a total of 245 more papers were evaluated. After this approach, 9 more papers were included, totaling 23 papers read and assessed. Tables 3.1 and 3.2 show the papers' analysis and the generation of the search string. Figure 3.1 demonstrates the steps that were taken when performing the search.

With a quick search on the internet, thousands of customized games are available in various databases, demonstrating that the community has already engaged in the practice of game modification (UNGER, 2012).

Based on the first stage, it was determined that the gaming community is quite active when it comes to game creation, enhancements, and changes. As a result, an additional stage was added to the study. A search of the gray literature was conducted in addition to the primary study.

The gray literature search covered up to page 10 of Google for each of the search keywords, yielding 700 links that needed to be validated. After visiting each link, the inclusion and exclusion criteria were applied, resulting in the selection of 21 links for the quality criteria step. The following quality criteria were used to identify and approve these links, where 10 links were selected and approved. The snowballing process was accomplished through the use of backlinks (website reference links). As a result, the entire procedure was restarted for those connections that were authorized, and an additional 335 links were validated. Additionally, 4 other links were included to the search. Finally, 14 more documents were added to the search.

The papers that were chosen for this research are listed in Table 3.2, along with the questions that each document may answer. Column S denotes the kind of study; S denotes the Scopus database, I denotes IEEEXplorer, EI denotes El Compendex, SD is Science Direct, and B denotes Backward and F denotes Forward snowballing.

It is worth noting that just 4 of the second stage's documents are links. The remaining 10 documents are copies of documents discovered during the search. As previously mentioned, one of the primary objectives of multivocal is to illustrate the

practical side of a subject, taking into consideration the fact that gray literature (websites, blogs) demonstrates more practical actions. This can be seen in Table 3.3, where the majority of the documents identified in step two address topic four, with an emphasis on the practical side. The green and red colors were used to show whether a paper answered a question or not.

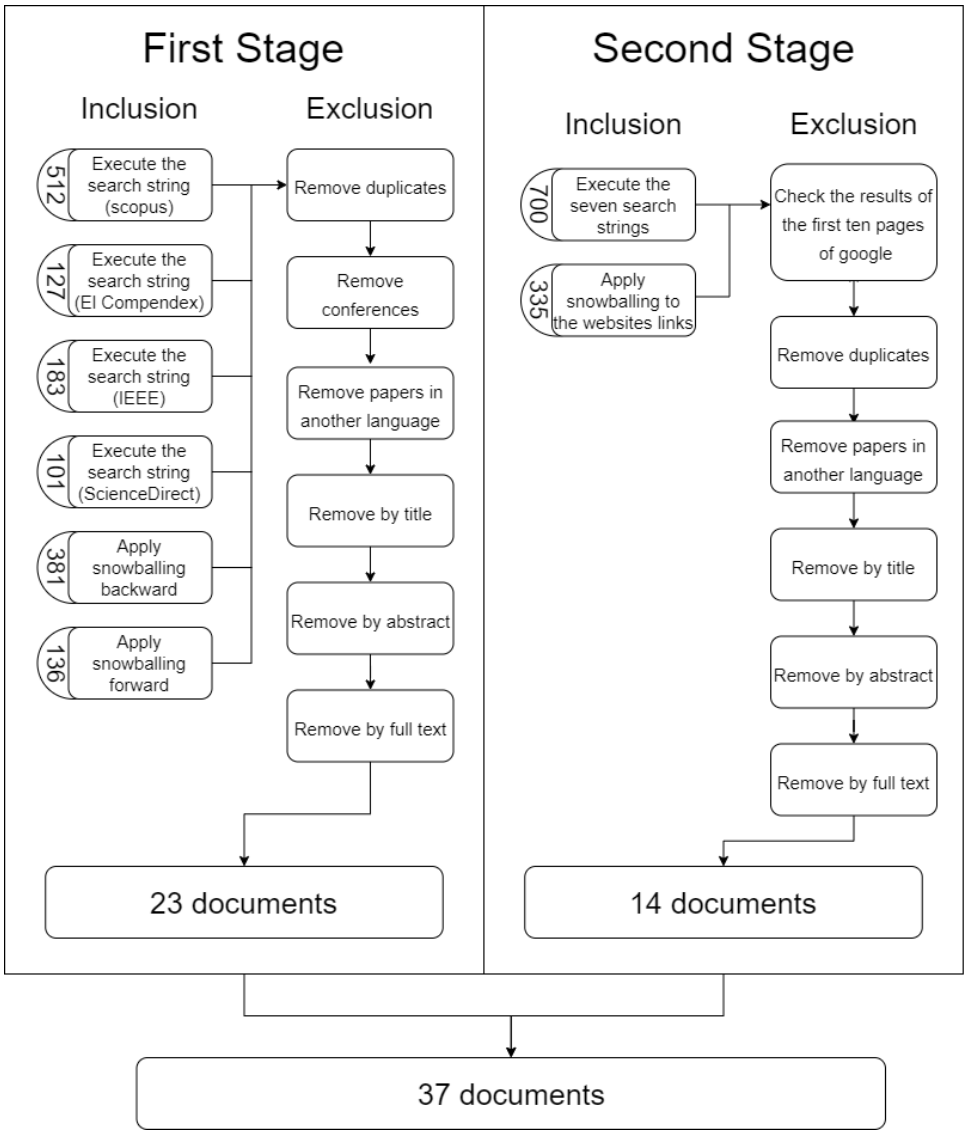

Figure 3.1: Search flow of research on mutators and games.

Table 3.2: Analysis of the papers about mutators and games.

	Scopus		ScienceDirect		IEEEXplore		El Compendex		Snowballing Backward		Snowballing Forward	
Activity	**Result**	**Number of paper**	**Result**	**Number of paper**	**Result**	**Number of paper**	**Result**	**Number of paper**	**Result**	**Number of paper**	**Result**	**Number of paper**
First Execution	512 added	512	101 added	101	183 added	183	127 added	127	381 added	381	136 added	136
Number of papers	**1440 papers**											
Repeated Papers	51 withdraw	461	7 withdraw	94	23 withdraw	160	0 withdraw	127	141 withdraw	240	20 withdraw	116
Remove conference	48 withdraw	413	7 withdraw	87	0 withdraw	160	0 withdraw	127	0 withdraw	240	0 withdraw	116
Papers in another language	10 withdraw	403	0 withdraw	87	0 withdraw	160	0 withdraw	127	4 withdraw	234	3 withdraw	113
Number of papers	**1110 papers**											
Remove by title	324 withdraw	78	77 withdraw	10	139 withdraw	21	96 withdraw	36	172 withdraw	62	86 withdraw	27
Number of papers	**207 papers**											
Remove by abstract	59 withdraw	14	10 withdraw	0	18 withdraw	3	20 withdraw	16	40 withdraw	22	18 withdraw	9
Number of papers	**56 papers**											
Papers not found	0 withdraw	14	0 withdraw	0	0 withdraw	3	0 withdraw	16	0 withdraw	22	0 withdraw	9
Remove by quality criteria	2 withdraw	12	0 withdraw	0	0 withdraw	3	6 withdraw	10	7 withdraw	15	0 withdraw	9
Remove by full paper	0 withdraw	12	0 withdraw	0	1 withdraw	2	10 withdraw	0	11 withdraw	4	4 withdraw	5
Extracted Papers	**23 papers**											

Table 3.3: Traceability matrix of mutators and games.

Title	**Year**	**Q1**	**Q2**	**Q3**	**Q4**	**S**
First Stage						
Building the Perfect Game – An Empirical Study of Game Modifications LEE *et al.*	2020	X	X	X	X	S
To mod or not to mod—an empirical study on game modding as customer value co-creation BILIŃSKA *et al.*	2020		X	X		F
Modding tabletop games for education ABBOTT	2019	X	X	X	X	S
Migrating Java-based apo-games into a composition-based software product line DEBBICHE *et al.*	2019		X	X	X	F
Product line architecture recovery with outlier filtering in software families: the Apo-Games case study LIMA *et al.*	2019	X			X	F
Apo-games-a case study for reverse engineering variability from cloned Java variants KRÜGER *et al.*	2018			X	X	S
Multi-objective optimization for reverse engineering of apo-games feature models MENDONÇA *et al.*	2018		X	X	X	S
Visual and computational modelling of minority games DAMAŠEVIČIUS and AŠERIŠKIS	2017	X	X		X	S
Placing value on community co-creations: A study of a video game 'modding' community PORETSKI and ARAZY	2017		X	X	X	B
Analysis of popularity of game mods: A case study DEY *et al.*	2016		X	X		S
Serious mods: A case for modding in serious games pedagogy MCARTHUR and TEATHER	2016	X		X		I
Design of a math learning game using a Minecraft mod AL-WASHMI *et al.*	2014	X	X	X	X	S

Applying exception handling patterns for user interface customization in software games modification TENGTRIRAT and PROMPOON	2013	X	X	X	X	F
An environment to support collaborative learning by modding GEORGE *et al.*	2014		X	X	X	F
Reporting about the Mod software process ZUPPIROLI *et al.*	2012		X		X	B
A Role-Playing Game for a Software Engineering Lab: Developing a Product Line ZUPPIROLI *et al.*	2012		X		X	I
Remix and play: Lessons from rul ts in texas hold'em and halo 2 CHEUNG and HUANG	2012		X	X	X	S
Modding as part of game culture UNGER	2012	X		X	X	S
Utilizing a 3D game engine to develop a virtual design review system SHIRATUDDIN and THABET	2011	X		X	X	S
Modding as an open source approach to extending computer game systems SCACCHI	2011	X		X	X	S
When the game is not enough: Motivations and practices among computer game modding culture SOTAMAA	2010	X	X	X		B
Modding as a basis for developing game systems SCACCHI	2011	X		X	X	S
Of mods and modders: Chasing down the value of fan-based digital game modifications POSTIGO	2009	X		X	X	B
Am I Mod or Not? - an Analysis of First Person Shooter Modification Culture NIEBORG	2005	X	X	X	X	B
Second Stage						
Unofficial patch KOLBERT	2021	X		X	X	B
Mod (video gaming) PATTERSON	2021	X			X	G
Appropriation & Motivation in Game Modification WEEKE	2020	X	X		X	G
Video game conversion MAGIOLADITIS	2020	X				G

Players as Content Creators the Benefits of Game Modding According to Polish Users. HOFMAN-KOHLMEYER	2019	X	X		X	B
Mod (video games) CLELAND	2018	X	X	X	X	G
Understanding Game Modding through Phases of Mod Development AGARWAL and SEETHARAMAN	2015	X	X	X		G
Does game modding require programming? RAMADAN	2015	X			X	B
Computer game modders' motivations and sense of community: A mixed-methods approach POOR	2014	X		X		B
Game Mods: Design, Theory and Criticism CHAMPION	2013	X	X			G
Computer game mods, modders, modding, and the mod scene WALT	2010	X	X	X	X	G
On modder labour, commodification of play, and mod competitions RAMADAN	2007	X	X	X	X	G
Am I Mod or Not? - An analysis of First Person Shooter modification culture. NIEBORG	2005	X	X	X	X	G

3.2.1 Results

The rise of the mod trend is closely related to increased accessibility to personal computers and the expansion of the internet, which is disseminating an increasing amount of content (SOTAMAA, 2007). The community and academy are increasingly generating game adaptations, assisting game producers in a variety of ways, including recruiting new players, extending the life of a game, providing new views for the game, and resolving bugs. Modifications, in general, are referred to as mods and may be thought of as modifications to an original game (SCACCHI, 2011b).

Modifications to products in the gaming industry done by gamers are now often referred to as modding. Modders employ a variety of strategies in their creations, ranging from basic rearranging of game world parts to complete conversions that can be somewhat independent of the original game (SOTAMAA, 2007). This section will discuss the many sorts of modifiers discovered, their benefits and drawbacks, and lastly, the essential criteria for constructing an adapted game, as well as if there are tools available to aid in this process.

In this research, several studies were identified from different years and countries. Below, these findings will be demonstrated in more detail through graphics. Figure 3.3 shows how many studies answered each of the questions proposed in the study. Figure 3.5 shows the countries of origin of the documents found. Figure 3.4 shows studies grouped by year, while Figure 3.2 shows papers divided by search bases. In this figure, two Veen diagrams are shown, the first that demonstrates the search bases of the study and the second that uses these articles (main study) and compares them with the papers of the snowballing process. It is worth remembering that this diagram only shows information from the white literature.

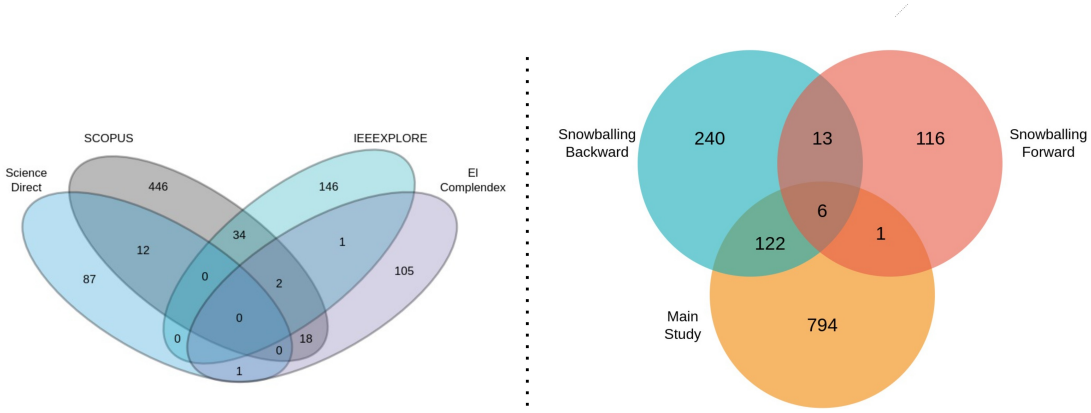

Figure 3.2: Number of papers found in the search on mutators and games grouped by search base.

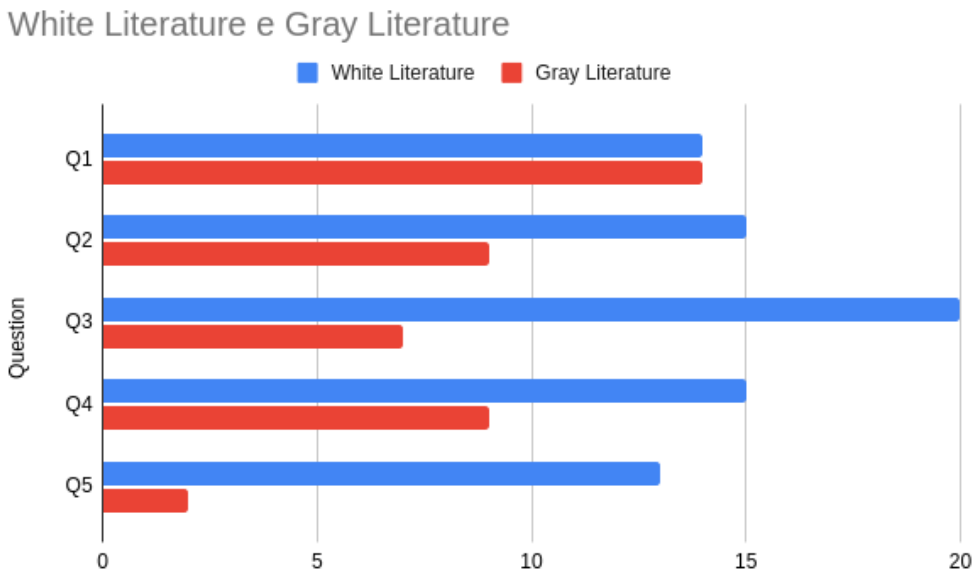

Figure 3.3: Number of questions answered per paper (Research on mutators and games).

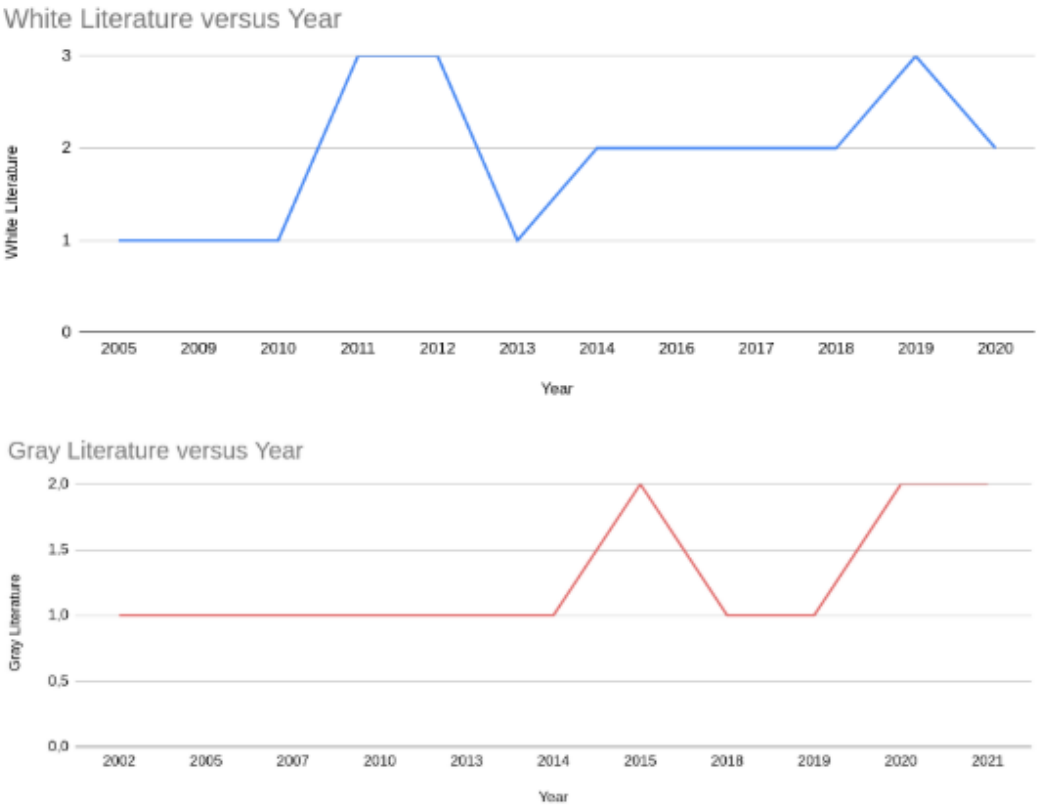

Figure 3.4: Number of papers found in the search on mutators and games grouped by year.

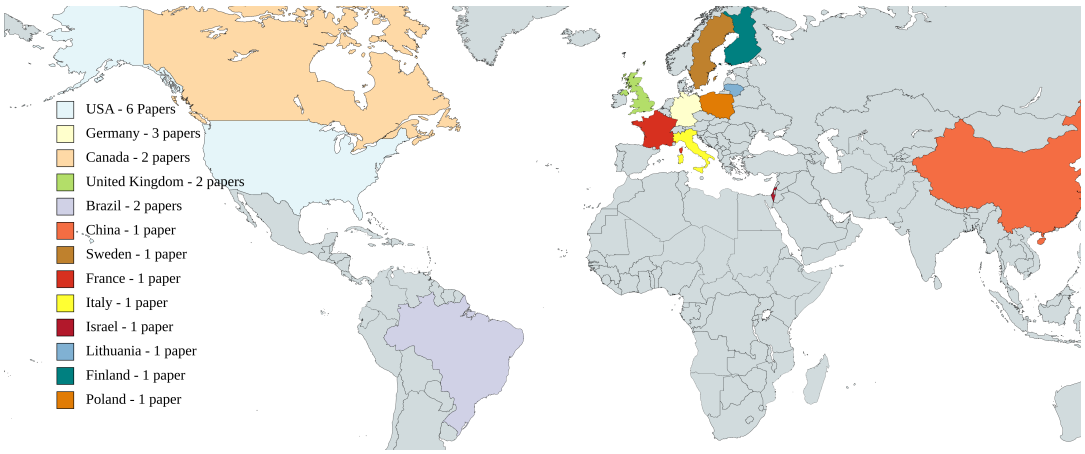

Figure 3.5: Number of papers found in the search on mutators and games grouped by country.

3.2.2 Summary of findings

Q1: What modifiers are used to create games from others?

Mods are as varied as computer games themselves. They vary in size and complexity and can make modest tweaks to the original game or entirely transform its appearance (NIEBORG, 2005a). Modding is the technique and process of modifying or adapting video games. It is often a "Do It Yourself" (DIY) strategy that teaches social and technical skills associated with innovation through the reuse of an existing game's concept. Numerous aspects of the game may be modified, including the UI, game items, problem patches, characters, and rules (LEE *et al.*, 2020). For example: changing the rules of a game produces a variant, allowing players to create a unique gaming experience (CHEUNG and HUANG, 2012).

Mods may be developed by applying mutators to a game. A mutator may be thought of as a change that is made to an existing game; for example, applying a mutator M to a game G results in the creation of a new game named G [M] (CLAUDE CHAUNIER). A game can be classified in a variety of ways depending on the amount of mutators employed. Numerous adaptations and modifications exist, each with a distinct function (LEE *et al.*, 2020; SCACCHI, 2011b,b). Each of them will be described in more detail in the following (BILIŃSKA *et al.*, 2020; CHAMPION, 2013; CLELAND; DACONCEICAO *et al.*, 2013; HOFMAN-KOHLMEYER, 2019; KOLBERT; MCARTHUR and TEATHER, 2015; PATTERSON; POOR, 2014; RAMADAN; SCACCHI, 2011a,b; SOTAMAA, 2010; UNGER, 2012; WALT, 2010; WEEKE, 2020).

- **Interface customization:** The interfaces are designed to emphasize the visual component of the game in order to enhance the experience. This customization entails making changes to the visual element, such as remodeling the accessories, skin, shader, or animation of a character or a game map, altering the game's colors, altering the information displayed on the screen, which is frequently used by players looking to maximize their enjoyment of the game, and displaying some additional information.
- **Conversions:** Conversions are by far the most prevalent type of alteration. They are aimed at changing something already existing or adding new features and can be classified as total or partial.
  - **Partial:** Add a map, a character, and an item; enhance the game's speed; incorporate little mechanisms, bots, and rules. Partial modifiers can still

be classified according to the modifications performed.

    * **Mutators/tweaks:** Modify or add restricted features that do not affect gameplay or mechanics. They can be minor adjustments like changing the game's theme song, boosting the game's speed, or altering some graphic components and minor rules.
    * **Add-ons:** They work as an extension to the game, such as adding additional maps or objects. The game's original principles, scenery, and gameplay have been significantly updated or expanded.
    * **Mods:** They may be thought of as the intersection of the preceding two, as they retain the ability to modify rules and configurations.

  – **Total:** Certain alterations go so far as to result in the creation of totally new games. For instance, a well-known conversion is the CounterStrike mod, which was a Half-Life adaption. In general, the amount of modifiers employed distinguishes a partial conversion from a whole conversion. A complete conversion occurs when a large number of modifiers are applied to the point where something new is created.

- **Machinima:** It may be thought of as the result of modifications that affect the visual replay of game usage sessions. Games are utilized for other goals in this sort of modification, such as conveying a story, creating a movie, or recreating a gaming experience.

- **Hacking:** Concentrates on enhancing the competitive edge in games by broadening the range of experiences in which players may participate via alternative technologies.

- **Patch:** They often focus on resolving unsolved issues and implementing technical improvements. When a community creates this modification, it is referred to as an unofficial or fan patch.

Mods have been classified into several categories depending on the modifiers introduced. On the other hand, they can still be classified according to the standpoint from which they were created. Each of these areas is briefly discussed in further detail below (MAGIOLADITIS).

- **Direct conversions:** the original game's source code is used with just minor alterations;

- **Imitations/clones:** fundamentally, the game remains unchanged. However, the title, images, and audio are frequently altered in order to avoid legal complications;

- **Remakes:** rebuild the game on a new technology;
- **Retro/Emulation:** design an emulator that allows the original game to be played on a different platform.

Mods, according to some students, do not include the cheats, hacks, or fixes (NIEBORG, 2005a). The primary question, though, is what are mods? Mods are defined in this work as a change to a game, which can have several levels, as previously stated. As a result, this work treats the terms above as modifications. It is worth noting that many games, even commercial ones, can be considered modifications. Numerous commercial game developers license an existing engine and customize it to their own requirements (NIEBORG, 2005a). In addition, several games started out as mods. For example, Counter-Strike, League of Legends and PUBG are some of the most famous ones (CLELAND).

Q2: What characteristics are needed to derive a game?

A game is a subcategory of software development in which designers, developers, and software engineers collaborate to create an experience that players may live through the game (PORETSKI and ARAZY, 2017). Once the game is out, the contributors devote their time for making updates and adding material to the main game; Modifications may include new game models, textures, audio, and mechanisms, as well as entire reimaginings (PORETSKI and ARAZY, 2017).

There are two main ways to mod development. The first occurs when a specific game demands expansion through the inclusion of additional components, and the second occurs when looking for games with comparable concepts to the one a gamer is building is discovered and cloned, with specific elements modified to fit the necessary scope (ABBOTT, 2018). Both need the same characteristics.

The development flow of a mod is followed by a sequence of activities. Below, this flow is described (CHEUNG and HUANG, 2012; CIGNONI, 2001).

Development flow of a mod

- **Inception:** Have an idea. Create a rough draft outlining the game's minimal needs. Create an early illustration for the game.
- **Experimentation:** Make sure the game is viable. Look for similar games. Search for games with similar characteristics to the game you want to build.
- **Design:** Define the mod's implementation strategy, including which files should be updated.

- **Development:** Modify the files selected in the previous step to meet one of the required specifications.
- **Test:** Install the mod on your computer and test the game.
- **Persist:** Building a mod occurs in an interactive way, where a new feature is modified to achieve a goal. Therefore, the test and development steps occur in development loops.

Thus far, the most conventional methods for mod construction have been discussed. Other models and frameworks, on the other hand, can aid in its development. Reverse engineering is a prevalent method for software creation, and this approach is also used to create modifications. It is based on the characteristics of a previously generated game and is frequently utilized when access to the source code is unavailable (MENDONÇA *et al.*, 2018). Another method is to use the UAREI model (User, Action, Rule, Entities, and Interface), which was developed with the goal of expressing and showing the game's mechanics graphically. This model includes several graphic components that have the role of illustrating the interactions that might occur while playing (DAMAŠEVIČIUS and AŠERIŠKIS, 2017). Another model that bears a strong resemblance to UAREI is LM-GM, which may be thought of as a visual model for executing and testing a gameplay. LM-GM combines gameplay loops, allowing execution points to be identified. Through its activities, it is possible to observe the game in action. This considerably simplifies and strengthens the game design process, particularly for modders who may not already possess a high degree of game literacy (ABBOTT, 2018).

A game is composed of a collection of components that work together to create the final product. By defining games, the required qualities for their building may be specified. Games may be characterized as activities that take place in an abstract environment where decisions, actions, and rules are created with the goal of achieving a leisure activity in the form of distraction or fun (XEXÉO *et al.*, 2013). On this basis, it is feasible to identify the following features that must be determined prior to the formation of any game: rules, actions, behaviors, objective, game loop, difficulty, and rewards (ABBOTT, 2018; EHRMANN *et al.*, 1968).

Table 3.4 shows each of the characteristics necessary for the interpretation and evolution of a game (ABBOTT, 2018; BILIŃSKA *et al.*, 2020; DACONCEICAO *et al.*, 2013; DAMAŠEVIČIUS and AŠERIŠKIS, 2017; SOTAMAA, 2010; WALT, 2010; WEEKE, 2020). These characteristics were classified into four broad categories that capture the games' properties at a higher level of abstraction. It is worth noting that game mechanics were formerly split into actions and behaviors.

Table 3.4: Derivation of game characteristics

Operation rules	Rules about the player. E.g.: the player can only carry one weapon at a time.	AL-WASHMI *et al.* (2014); BILIŃSKA *et al.* (2020); CHEUNG and HUANG (2012); CIGNONI (2001); LEE *et al.* (2020); LIMA *et al.* (2019); PORETSKI and ARAZY (2017)
Transition rules/states	Understanding the character's state transitions. E.g .: the player can only shoot if he/she has a weapon in his/her hand.	ABBOTT (2018); AL-WASHMI *et al.* (2014); BILIŃSKA *et al.* (2020); LIMA *et al.* (2019); MENDONÇA *et al.* (2018); ZUPPIROLI *et al.* (2012)
Actions	Commands that can be executed by the character. E.g.: shooting and walking.	AL-WASHMI *et al.* (2014); BILIŃSKA *et al.* (2020); DAMAŠEVIČIUS and AŠERIŠKIS (2017); LEE *et al.* (2020); LIMA *et al.* (2019); ZUPPIROLI *et al.* (2012)
Game world		
Levels	The game's stages. Strongly influenced by the gameplay that can change from one stage to the next.	ABBOTT (2018); AL-WASHMI *et al.* (2014); BILIŃSKA *et al.* (2020); LIMA *et al.* (2019); MENDONÇA *et al.* (2018); SHIRATUDDIN and THABET (2011); ZUPPIROLI *et al.* (2012)
Rules of objects	Rules of the objects contained in the world. Eg .: when an object must be locked or unlocked.	ABBOTT (2018); ÅKESSON *et al.* (2019); BILIŃSKA *et al.* (2020); DAMAŠEVIČIUS and AŠERIŠKIS (2017); LEE *et al.* (2020); LIMA *et al.* (2019)

Behavioral rules	Rules of behavior that the world can exhibit. Eg .: if the player collects a specific item it can start to rain.	ABBOTT (2018); CIGNONI (2001); DAMAŠEVIČIUS and AŠERIŠKIS (2017); LIMA *et al.* (2019); PORETSKI and ARAZY (2017); SHIRATUDDIN and THABET (2011)
Temporal states	It works like a state machine, depending on the world's state, it can only go to a specific one. E.g .: if the player is in phase 2 he/she can only go to phase 3.	ABBOTT (2018); CIGNONI (2001); LEE *et al.* (2020); LIMA *et al.* (2019,?); SHIRATUDDIN and THABET (2011)
Mission	What you want to achieve/complete.	ABBOTT (2018); ALWASHMI *et al.* (2014); BILIŃSKA *et al.* (2020); LEE *et al.* (2020); LIMA *et al.* (2019); PORETSKI and ARAZY (2017); SHIRATUDDIN and THABET (2011)
Obstacles	What you want to achieve/complete.	BILIŃSKA *et al.* (2020); CIGNONI (2001); LIMA *et al.* (2019); MCARTHUR and TEATHER (2015); PORETSKI and ARAZY (2017); SHIRATUDDIN and THABET (2011); ZUPPIROLI *et al.* (2012)
Game Play		
Winning and losing conditions	Conditions to win or lose the game.	ABBOTT (2018); BILIŃSKA *et al.* (2020); CIGNONI (2001); LIMA *et al.* (2019); MCARTHUR and TEATHER (2015); NIEBORG (2005b)

Strategic dilemmas	Strategies that can be used in the game. E.g .: combo attacks.	ABBOTT (2018); BILIŃSKA *et al.* (2020); CIGNONI (2001); LIMA *et al.* (2019); MCARTHUR and TEATHER (2015); NIEBORG (2005b)
Chains of actions	Chain of actions that can be combined. Eg .: player action with a map action.	ABBOTT (2018,?); BILIŃSKA *et al.* (2020); CIGNONI (2001); ZUPPIROLI *et al.* (2012)
General features		
Rules	Encapsulates the logic inside the system.	ABBOTT (2018); BILIŃSKA *et al.* (2020); CHEUNG and HUANG (2012); DAMAŠEVIČIUS and AŠERIŠKIS (2017); GEORGE *et al.* (2013); LEE *et al.* (2020); LIMA *et al.* (2019); MENDONÇA *et al.* (2018)
Score	The points obtained by the player throughout the game.	ABBOTT (2018); BILIŃSKA *et al.* (2020); CIGNONI (2001); LIMA *et al.* (2019); MENDONÇA *et al.* (2018); ZUPPIROLI *et al.* (2012)
Behaviors	Commands that are executed by the system.	ABBOTT (2018); BILIŃSKA *et al.* (2020); CIGNONI (2001); LEE *et al.* (2020); LIMA *et al.* (2019); MENDONÇA *et al.* (2018)

Goal	What you want to achieve/complete.	ABBOTT (2018); AL-WASHMI *et al.* (2014); BILIŃSKA *et al.* (2020); LEE *et al.* (2020); LIMA *et al.* (2019); SHIRATUD-DIN and THABET (2011)
Challenge	What must be accomplished to achieve the goal.	ABBOTT (2018); BIL-IŃSKA *et al.* (2020); LEE *et al.* (2020); LIMA *et al.* (2019); MCARTHUR and TEATHER (2015); MEN-DONÇA *et al.* (2018)
Rewards	What to obtain when reaching the goal.	ABBOTT (2018); BIL-IŃSKA *et al.* (2020); DAMAŠEVIČIUS and AŠERIŠKIS (2017); GEORGE *et al.* (2013); LEE *et al.* (2020); LIMA *et al.* (2019)
Game loop	Flow of engagement of the game. It is the execution of the game where the player seeks a goal by executing a challenge and being rewarded with something.	BILIŃSKA *et al.* (2020); CIGNONI (2001); LEE *et al.* (2020); MENDONÇA *et al.* (2018); ZUPPIROLI *et al.* (2012)
Interface	The visual of the game, the game's sprites, and graphics.	AL-WASHMI *et al.* (2014); DAMAŠE-VIČIUS and AŠERIŠKIS (2017); GEORGE *et al.* (2013); LIMA *et al.* (2019); MCARTHUR and TEATHER (2015); MENDONÇA *et al.* (2018); SHIRATUDDIN and THA-BET (2011)

Entities	Objects and elements instantiated within the game.	CIGNONI (2001); DAMAŠEVIČIUS and AŠERIŠKIS (2017); LIMA *et al.* (2019); MENDONÇA *et al.* (2018); NIEBORG (2005b)

Q3: What are the advantages and difficulties of creating games from others?

While generalizing the motivations of mod makers is difficult, there are various factors that contribute to a user creating a mod. Among the most significant ones are the following: attempting new things, resolving bugs, creating new characters, increasing the difficulty of the game, gaining advantages in the game, extending the game's life cycle, the software was originally designed for a significantly different environment and may require improvement, the official developer is unable to deal with the problems, and so on (AGARWAL and SEETHARAMAN, 2015; KOLBERT; LEE *et al.*, 2020; POOR, 2014).

Modifiers, as well as games, are complex and take time to develop. The time required to produce a mod varies significantly. Being constructed in a few days or requiring a long time to build, with the advantage of reusing some pieces. As previously mentioned, developing a game may be extremely time demanding and might take years. However, the time required to release a mod is far shorter, requiring on average roughly 345 days with the game already completed (LEE *et al.*, 2020), while games can take years to complete. Mods represent a way for the community to contribute to the original game. Depending on the mod's nature, it may only require one or multiple releases. For instance, a mod that enhances a game's texture may require only one version. In comparison, a mod that conducts a thorough evaluation may need one or more releases (HOFMAN-KOHLMEYER, 2019; LEE *et al.*, 2020). It is worth noting that there is a much longer period of time between an official game release and a mod release than there is between consecutive mod releases (LEE *et al.*, 2020). Along with the increase of the game's production period, the cost of the game climbs significantly. While the cost of developing a digital game for commercial distribution has been estimated at over ten million dollars, the cost of modding is far lower (POSTIGO, 2007).

An advantage that can also be attributed to the use of modifiers is the ability to increase the longevity of games (BILIŃSKA *et al.*, 2020; LEE *et al.*, 2020; SOTAMAA, 2010). Every game has a useful life cycle. Modifiers, on the other hand, can

extend the life of the game by adding additional instructions, characters, levels, and other factors, giving players more areas to explore. Using the same logic, modifications may help boost sales, income, and profits for original games, as many people purchase the original game in order to play the mod (AGARWAL and SEETHARAMAN, 2015; LEE *et al.*, 2020; WALT, 2010). The life of a game is not only measured in terms of how long it remains on bestseller lists or can maintain its launch price, but also in terms of how long it is consumed, as well as fans discuss their favorite games on gaming websites and magazines, host servers for team games, distribute game information, and discuss the latest add-on (POSTIGO, 2007; TENGTRIRAT and PROMPOON, 2013).

Another key benefit of modifications is their capacity to attract new players to the game, so extending its longevity. For instance, the Dota 2 game was a mod for the Warcraft game that surpassed 450,000 daily players five years after its debut and 16 years after the original game's release. Thus, the game's player base and longevity are increased (LEE *et al.*, 2020; POSTIGO, 2007). It is worth noting that according to some authors of the Top Ten of First Person Shooters, the success of the Unreal Tournament was mainly due to the mods and mutators available for the game (NIEBORG, 2005a). Additionally, modifiers might provide developers with respite by taking off the pressure to provide fresh material for a game. Remember that in some cases, even modifiers can generate new games, which is the case with Warcraft 3 and Dota 2 (LEE *et al.*, 2020).

Along with the benefits described above, several additional are still directly tied to the community member who worked on the modification. Among the primary ones that stand out are the user's expression, communication between the firm and the end-user, and diversity of the game via end-user ideas (BILIŃSKA *et al.*, 2020). Mods have become such a common practice in the community that they may now be considered a form of culture, allowing the user to incorporate their experiences into the game (SCACCHI, 2011a; UNGER, 2012).

When a player buys a game, he or she obtains a license to use the product. This license is structured in the manner of a copyright-based agreement (WALT, 2010). Mods made by the community are susceptible to contract violations, placing their authors at considerable danger of being held liable for their actions, which frequently involve scamming games, exploiting product faults, or committing copyright infringement (SCACCHI, 2011b).

Last but not least, this mod technique has gotten so widespread that many large game production firms have opted to enlist members of the gaming community in order to cut development costs and risks. This cooperation enables the players to pool their unique perspectives and extensive experience and skills, therefore boosting the quality of the innovation without incurring additional resources, as well as certifying

that it was what the community expected in terms of the game (PORETSKI and ARAZY, 2017; TENGTRIRAT and PROMPOON, 2013).

Despite all the advantages described so far, some difficulties and challenges must be observed when creating mods. The first and main problem is the initial investment to produce a mod, which is necessary to understand the source code, reverse engineering and extract its features (KRÜGER *et al.*, 2018; MENDONÇA *et al.*, 2018). Following this line of reasoning, some studies have already been carried out using the product line. However, this approach also requires an initial investment to conceptualize the initial features of the project (KRÜGER *et al.*, 2018; MENDONÇA *et al.*, 2018).

Table 3.5 demonstrates the advantages and disadvantages of using mods. The green color shows the advantages and the red the disadvantages.

Table 3.5: Advantages and disadvantages of using mods.

Advantages / Disadvantages	**Papers**
Communication between the company and the end-user	AGARWAL and SEETHARAMAN (2015); KOLBERT; LEE *et al.* (2020) POOR (2014)
Diversification of the game through end-user ideas	HOFMAN-KOHLMEYER (2019); LEE *et al.* (2020); POSTIGO (2007)
Decreases the risk of game bugs	AGARWAL and SEETHARAMAN (2015); BILIŃSKA *et al.* (2020); KOLBERT SOTAMAA (2010); TENGTRIRAT and PROMPOON (2013); WALT (2010) LEE *et al.* (2020); POOR (2014)
Create new instructions, characters, levels, and other elements, providing players with new aspects to explore	BILIŃSKA *et al.* (2020); SCACCHI (2011a); UNGER (2012)
Increase the game life cycle	BILIŃSKA *et al.* (2020); LEE *et al.* (2020); SOTAMAA (2010) AGARWAL and SEETHARAMAN (2015); TENGTRIRAT and PROMPOON (2013); WALT (2010)
Shorter development time	BILIŃSKA *et al.* (2020); PORETSKI and ARAZY (2017); TENGTRIRAT and PROMPOON (2013)

Lower development cost	AGARWAL and SEETHARAMAN (2015); TENGTRIRAT and PROMPOON (2013); WALT (2010) KOLBERT; POOR (2014)
Increase the number of players of the original game	AGARWAL and SEETHARAMAN (2015); LEE *et al.* (2020); POOR (2014)
High initial investment	KRÜGER *et al.* (2018); MENDONÇA *et al.* (2018)

Q4: What tools or frameworks support these changes?

Numerous frameworks and tools facilitate the building of modifications. However, the most prevalent technique of mod development so far has been cloning and do-it-yourself. The modder selects the basic game to be updated, verifies the characteristics he/she wants to modify, and then produces the new game (KRÜGER *et al.*, 2018). This less complex strategy is referred to opportunistic reuse or ad-hoc reuse, and it comprises cloning, copying, and straining. Opportunistic reuse provides immediate advantages and produces the desired outcome. However, the quality of the project is not a priority, significant reworking leads to unexpected behavior and an unstable software structure (LIMA *et al.*, 2019).

Typically, games are changed using tools that enable access to an unencrypted internal representation of the game program. While it may appear as though game developers would aim to discourage consumers from customizing their games, this is not the case. Developers of video games are increasingly providing software tools for customizing their products in order to boost sales and market share (AGARWAL and SEETHARAMAN, 2015). Software development kits (SDKs) for games/domains supplied to users by game development studios represent a modern business approach for engaging users and assisting in product innovation outside the studio (HOFMAN-KOHLMEYER, 2019; PORETSKI and ARAZY, 2017; SCACCHI, 2011b,b). In addition to SDKs, which are the most common way of accessing the game's source code, several other platforms provide access to the game's source code and allow modifications. Among the main ones are the Creation Kit, GECK, Construction Set, MODKit, REDKit, Modbuddy, and D'jinni (LEE *et al.*, 2020).

Another possibility for the development of modifiers is through free software games, in which the end user has complete access to the game's source code and may modify it as desired (SCACCHI, 2011b). However, this strategy is used by small businesses or anonymous developers.

There are firms that assist and encourage the production of modifications with the goal of reducing problems, improving the game's quality and consistency, and

generating new ideas. This technique leverages the users' ideas and wants to generate improvements for the game sold. The Unreal engine was created to provide access to all of its technology's components. This enabled it to host multiple events dubbed Unreal Tournaments, in which the developer may express his/her creativity while developing his/her mods (LEE *et al.*, 2020; NIEBORG, 2005a). Other companies permit the construction of modifications as well, although without providing direct access to the components. For instance, Blizzard Entertainment's World of Warcraft has a UI modification tool that enables add-ons to modify the user interface panel, resulting in an enhanced gameplay experience. But these add-ons do not modify or convert the game into something entirely different since Blizzard seeks to ensure that players have access to the same configuration and mechanics as the original game (WALT, 2010).

In adiction, there are developers that produce mods by reverse engineering the source code of the original game. It is worth noting that this method for developing modifications is unlawful and violates the copyright of the original games (KOLBERT).

Finally, single mod distribution platform may include several modifications for a single game. They all, however, adapt the same basic game. These platforms must be demonstrated in some way, such as demonstrating which files were changed in each mod, or if one mod is compatible with another, as both can change the same original game file (LEE *et al.*, 2020).

Due to the large number of game variations generated based on an original game, maintenance can become difficult, and businesses may consider transitioning to a line of software products, referred to as an extractive method, to assist with mass game production (DEBBICHE *et al.*, 2019; KRÜGER *et al.*, 2018; LIMA *et al.*, 2019; MENDONÇA *et al.*, 2018; ZUPPIROLI *et al.*, 2012).

Finally, there are several sites that support mods, integrating the community and providing thousands of games. Among the main games are gamemodding [1], moddb [2], modsonline [3], among others (CIGNONI, 2001; LEE *et al.*, 2020).

3.2.3 Conclusion

Game companies are growing in size, generating billions of dollars each year, releasing many titles each year, and attracting fans of all ages and genres. However, as has been seen so far, developing a game may be a long process that might take years to complete. However, the gaming community is rising daily. With such a vast user community, some members may experience anxiety or dissatisfaction at

[1] http://www.gamemodding.net
[2] https://www.moddb.com
[3] http://modsonline.com/

the prospect of having to wait so long for a game to be published.

With a little online search, it is possible to locate multiple games for sale and several websites that provide modifications for them. As previously mentioned, a mod may be defined as a modification made to a game and depending on the level of this modification this mod can receive different names, such as: patches, tweaks, add-ons, among others. This method of modifying games can result in a number of benefits for the company that generated the original games. The benefits are numerous, and some businesses even encourage this practice. Among the primary benefits are an increase in users, an increase in sales, and an increase in the game's longevity, among others. However, the study revealed that the process of developing a mod might be expensive and ad-hoc.

It is noted the presence of tools and frameworks that support modifications, ranging from clones to SDKs and tools made accessible by the game's developer, among other techniques. However, these tools are frequently associated with a number of difficulties, including a large initial investment, a steep learning curve, the requirement to comprehend the source code, and the fact that the majority of these techniques are limited to the creation of basic games.

Throughout the investigation, multiple papers were found that contrasted ad-hoc mod development with opportunistic reuse, in which software is constructed by copy and paste (FENSKE *et al.*, 2017; SIERRA *et al.*, 2019). Demonstrating once again the need of systematizing the process of mod development. Additionally, it was seen in these same papers an early application of product lines for game building. However, it was employed superficially and exploratorily.

As is already known, RS can bring several types of advantages in the construction of software in general, from systematization of development to an increase in delivery speed and cost reduction. It was observed that these advantages were being highlighted in some articles through the research, but, however, the approaches demonstrated were of an exploratory nature, with the exception of software componentization, an approach that has already been used by programmers and is even found in some more current advanced engines (ALBASSAM and GOMAA, 2013; AOUADI *et al.*, 2016).

As a result, it was decided to conduct another search that would complement the one mentioned in this section. The proposed study sought to determine which RS techniques are being utilized to create games or mods. The study can be read in more detail in the next section.

3.3 Main Study

Based on the information from the viability study, it was possible to find several works that made references to SR techniques for building mods. As a result, this section main purpose is to identify how SR has been integrated into the game production process.

The protocol used in this search was demonstrated in Section 3.1. In the following, the research questions will be demonstrated. Figure 3.6 demonstrates the steps that were taken when performing the search.

Research Questions

- **Q1:** What was the motivation for using SR?
- **Q2:** What were the reuse methods used?
- **Q3:** What are the advantages of the methods used?
- **Q4:** What are the difficulties of the methods used?
- **Q5:** How were the methods applied?
- **Q6:** What are the requirements for creating games using reuse?
- **Q7:** What were the tools used?

The search string returned a total of 1200 papers. When analyzed according to the inclusion and exclusion filters, this number dropped to 22 papers. The snowballing process was performed from these papers, and 732 more papers were analyzed. After this procedure, more 13 papers were included, totaling 35 papers read and analyzed. Tables 3.7 and 3.6 show the generation of the search string and the papers' analysis. The colors green and red were used to show whether a paper answered a question or not. It is worth noting that the papers used to validate the search string were papers found in the study of the previous chapter that highlighted the use of SR.

Table 3.8 shows the documents that were selected for this search, showing which questions each document can answer. Column S indicates the type of study, S was used for the Scopus database, I for IEEEXplorer, EI for El Compendex, SD for Science Direct, B for Backward and F for Forward snowballing.

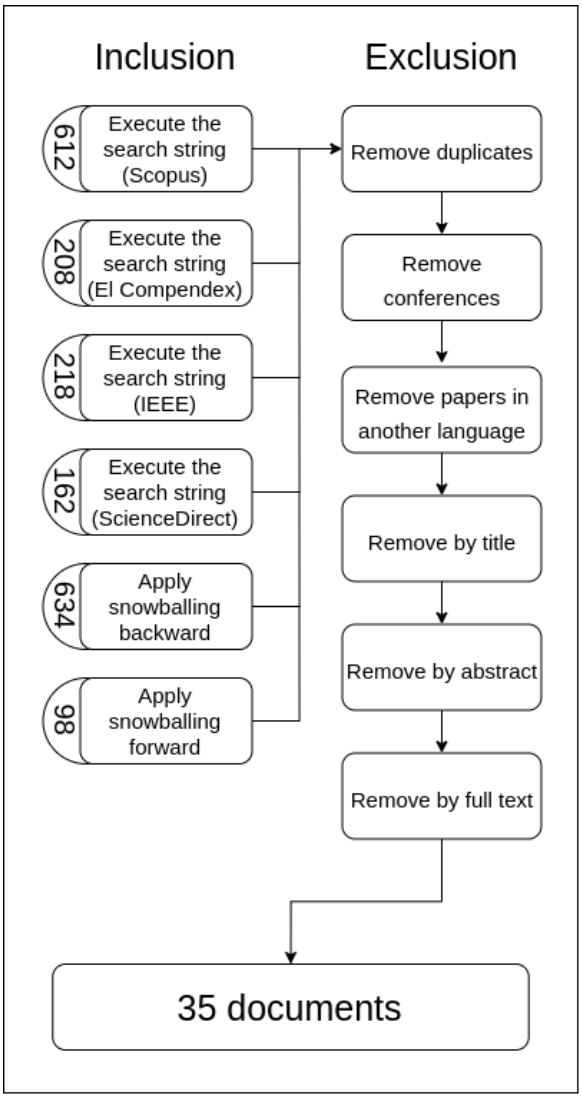

Figure 3.6: Search flow of research on games and reuse approaches.

Table 3.6: Search String of games and reuse approaches.

P	*Game*
I	Product line* OR product famil* OR variability OR model-driven* OR feature modeling OR model transformation OR component development OR asset* OR reuse OR software reuse
C	Not applicable
O	Tools OR approach* OR method* OR ideas OR framework*
C	Create OR creation OR production OR development OR elaboration OR generation OR generate

TITLE-ABS-KEY ((*game*) AND ("product line*" OR "product famil*" OR variability OR model-driven* OR "feature modeling" OR "model transformation" OR "component development" OR asset* OR "reuse" OR "software reuse") AND (tools OR approach* OR method* OR ideas OR framework*) AND (create OR creation OR production OR development OR elaboration OR generation OR generate)) AND (LIMIT-TO (SUBJAREA , "COMP") OR LIMIT-TO (SUBJAREA , "ENGI"))	
Control papers	1 - Modding as part of game culture
	2 - Building the perfect game – an empirical study of game modifications pedagogy

Table 3.7: Analysis of the papers about games and reuse approaches.

	Scopus		ScienceDirect		IEEEXplore		El Compendex		Snowballing Backward		Snowballing Forward	
Activity	**Result**	**Number of paper**	**Result**	**Number of paper**	**Result**	**Number of paper**	**Result**	**Number of paper**	**Result**	**Number of paper**	**Result**	**Number of paper**
First Execution	612 added	612	162 added	162	218 added	218	208 added	208	634 added	634	98 added	98
Number of papers	**1932 papers**											
Repeated Papers	118 withdraw	494	64 withdraw	98	41 withdraw	177	54 withdraw	154	251 withdraw	383	34 withdraw	64
Remove conference	48 withdraw	413	7 withdraw	87	0 withdraw	160	0 withdraw	127	0 withdraw	240	0 withdraw	116
Papers in another language	10 withdraw	403	0 withdraw	87	0 withdraw	160	0 withdraw	127	4 withdraw	234	3 withdraw	113
Number of papers	**1110 papers**											
Remove by title	324 withdraw	78	77 withdraw	10	139 withdraw	21	96 withdraw	36	172 withdraw	62	86 withdraw	27
Number of papers	**207 papers**											
Remove by abstract	59 withdraw	14	10 withdraw	0	18 withdraw	3	20 withdraw	16	40 withdraw	22	18 withdraw	9
Number of papers	**56 papers**											
Papers not found	0 withdraw	14	0 withdraw	0	0 withdraw	3	0 withdraw	16	0 withdraw	22	0 withdraw	9
Remove by quality criteria	2 withdraw	12	0 withdraw	0	0 withdraw	3	6 withdraw	10	7 withdraw	15	0 withdraw	9
Remove by full paper	0 withdraw	12	0 withdraw	0	1 withdraw	2	10 withdraw	0	11 withdraw	4	4 withdraw	5
Extracted Papers	**23 papers**											

Table 3.8: Traceability matrix of games and reuse approaches.

Title	**Year**	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Q7**	**S**
SeGa4Biz: Model-Driven framework for developing serious games for business processes GUO *et al.* (2015b)	2021		X	X	X	X	X	X	EI
A model-driven framework for developing android-based classic multiplayer 2D board games DERAKHSHANDI *et al.* (2021)	2021	X	X	X	X	X	X	X	S
ProDSPL: Proactive self-adaptation based on Dynamic Software Product Lines AYALA *et al.* (2021)	2021	X	X	X	X	X	X		S
PhyDSLK: a model-driven framework for generating exergames BALDASSARRE *et al.* (2021)	2020	X	X	X	X	X	X		EI
Lessons from practicing an adapted model driven approach in game development GUO *et al.* (2015b)	2020	X	X	X	X	X	X		EI
Product line architecture recovery with outlier filtering in software families: the Apo-Games case study LIMA *et al.* (2019)	2019	X	X			X			S
Model-driven game development: A literature review LIMA *et al.* (2019)	2019	X	X			X	X	X	S
Migrating the android apo-games into an annotation-based software product line LIMA *et al.* (2019)	2019	X	X	X		X	X	X	S
A Comparative Analysis of Game Engines to Develop Core Assets for a Software Product Line of Mini-Games SIERRA *et al.* (2019)	2019		X	X	X	X	X		F
Creating a software product line of mini-games to support language therapy GUO *et al.* (2015b)	2018	X	X	X	X	X		X	EI

AsKME: A Feature-Based Approach to Develop Multiplatform Quiz Games SARINHO *et al.* (2018)	2018	X				X	X	X	S
Making Serious Games with Reusable Software Components VEGT *et al.* (2018)	2018			X			X	X	F
A model-driven approach to generate and deploy videogames on multiple platforms NÚÑEZ-VALDEZ *et al.* (2017)	2017		X	X	X	X	X	X	S
MEnDiGa: A Minimal Engine for Digital Games BOAVENTURA and SARINHO (2017)	2017	X	X		X	X	X	X	S
Variant-preserving refactorings for migrating cloned products to a product ling FENSKE *et al.* (2017)	2017	X	X	X	X	X	X		B
Engine-cooperative game modeling (ECGM): Bridge model-driven game development and game engine tool-chains ZHU *et al.* (2016)	2017	X			X	X	X	X	S
RAGE reusable game software components and their integration into serious game engines VEGT *et al.* (2016)	2017		X	X	X	X	X	X	S
Models and mechanisms for implementing playful scenarios AOUADI *et al.* (2016)	2016		X		X	X	X	X	I
The RAGE advanced game technologies repository for supporting applied game development GEORGIEV *et al.* (2016)	2016	X	X	X	X	X	X		EI
A Workflow for Model Driven Game Development GUO *et al.* (2015a)	2015	X	X	X	X	X	X		I
A DSL for rapid prototyping of cross-platform tower defense SÁNCHEZ *et al.* (2015)	2015		X	X		X	X		I

Building a Game Engine: A Tale of Modern Model-Driven Engineering GUANA *et al.* (2015)	2015	X	X	X	X		X	X	B
Applying software product lines to multiplatform video games ALBASSAM and GOMAA (2013)	2013	X	X	X	X	X	X	X	EI
A platform independent game technology model for model driven serious games development TANG *et al.* (2013)	2013		X		X	X	X	X	S
A model driven development framework for serious games THILLAINATHAN (2013)	2013		X	X		X	X	X	S
Gade4all: developing multiplatform videogames based on domain specific languages and model driven engineering VALDEZ *et al.* (2013)	2013	X		X		X	X	X	B
A model driven serious games development approach for game-based learning TANG and HANNEGHAN (2013)	2013	X	X	X		X	X		B
A feature-based environment for digital games SARINHO *et al.* (2012)	2012	X			X	X	X	X	B
The RPG DSL: A case study of language engineering using MDD for generating RPG games for mobile phones MARQUES *et al.* (2012)	2012	X	X		X	X	X	X	B
SharpLudus revisited: From ad hoc and monolithic digital game DSLs to effectively customized DSM approaches FURTADO *et al.* (2011)	2011					X	X	X	B
Model-driven game development-case study. A mtc for maze-game s prototyping MORALES *et al.* (2011)	2011	X	X		X	X	X	X	B

3.3.1 Results

The emergence of the mod trend is closely related to increased accessibility to personal computers and the expansion of the Internet, which is disseminating an increasing amount of content (SOTAMAA, 2007). The community and academy are progressively creating game adaptations, aiding game producers in a number of ways, including attracting new players, prolonging the life of the game, proposing new game concepts, and fixing issues. Modifications are referred to as mods in general and may be regarded as modifications to an original game (SCACCHI, 2011b).

Modifications made by community players to industrial games are now widely known as modding. Modders use a variety of methods to make their games different, from simple changes to the layout of the game world to completely new games that are independent of the original one (SOTAMAA, 2007). This section will discuss the many types of modifiers discovered, their benefits and drawbacks, and lastly, the essential criteria for constructing an adapted game, as well as if there are tools available to aid in this process.

Numerous studies from various years and countries were uncovered throughout this investigation. These findings will be illustrated in more detail below. Figure 3.9 shows how many studies answered each of the questions proposed in the study. Figure 3.10 shows the countries of origin of the documents found. Figure 3.8 shows studies grouped by year, while Figure 3.2 shows papers divided by search bases. In this figure, two Veen diagrams are shown, the first that demonstrates the research bases of the study and the second that uses these articles (main study) and compares them with the papers of the snowballing process.

It is possible to deduce many conclusions from these figures, such as the fact that multiple nations are interested in the subject of game development through reuse. The average number of papers published per year remained consistent, demonstrating that it is an interesting subject and that while many works apply SR techniques to game development, the majority of them, highlights the difficulties encountered, demonstrating that it is a hard task that can generate numerous benefits.

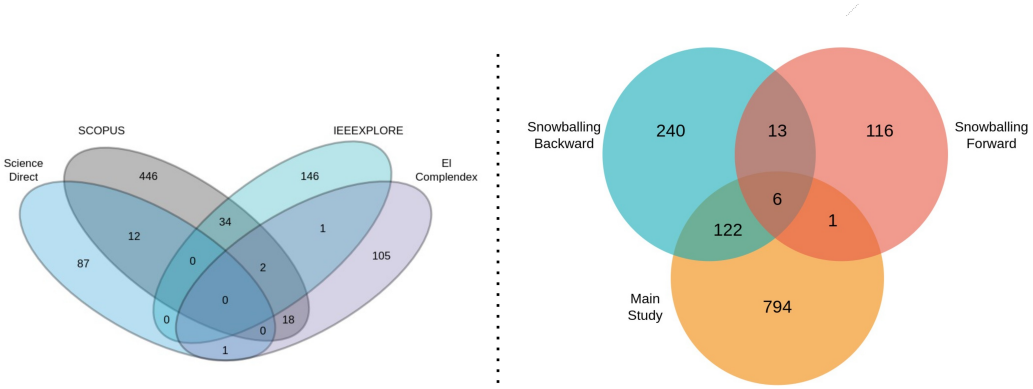

Figure 3.7: Number of papers found in the search on games and reuse approaches grouped by search base.

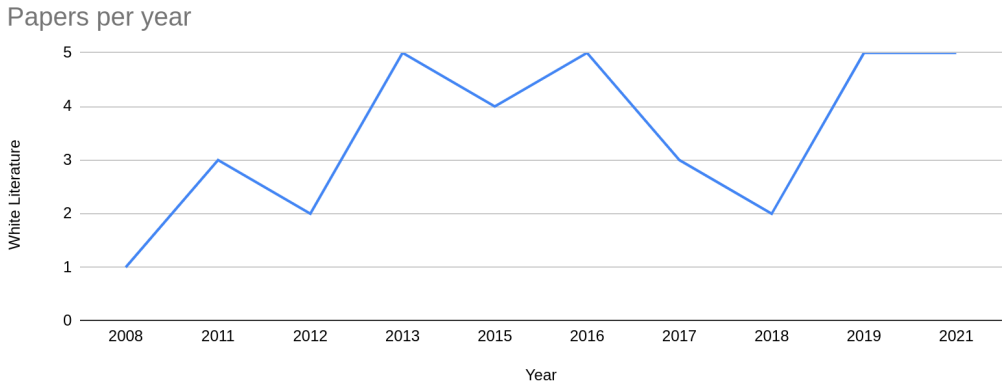

Figure 3.8: Number of papers found in the search on games and reuse approaches grouped by year.

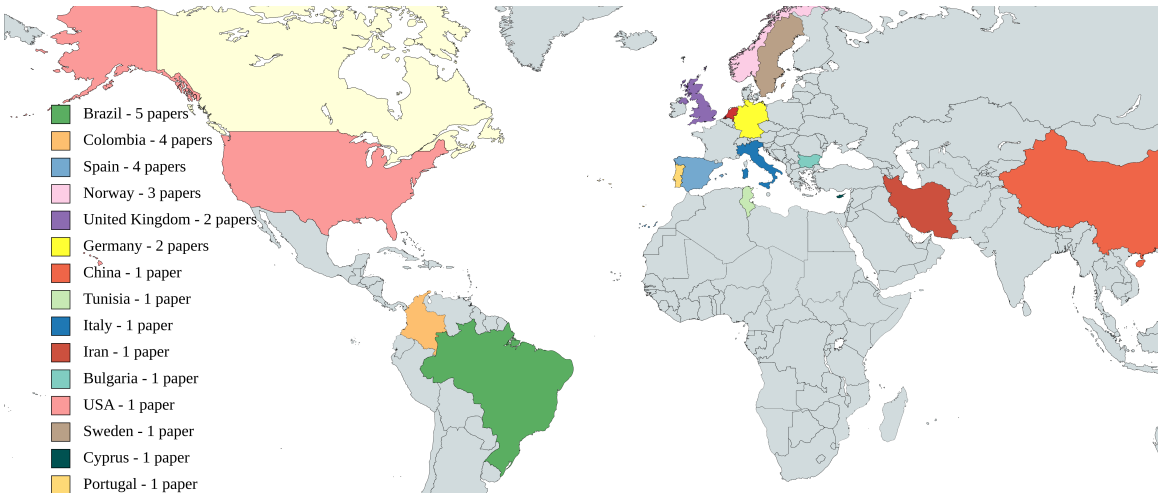

Figure 3.10: Number of papers found in the search on games and reuse approaches grouped by country.

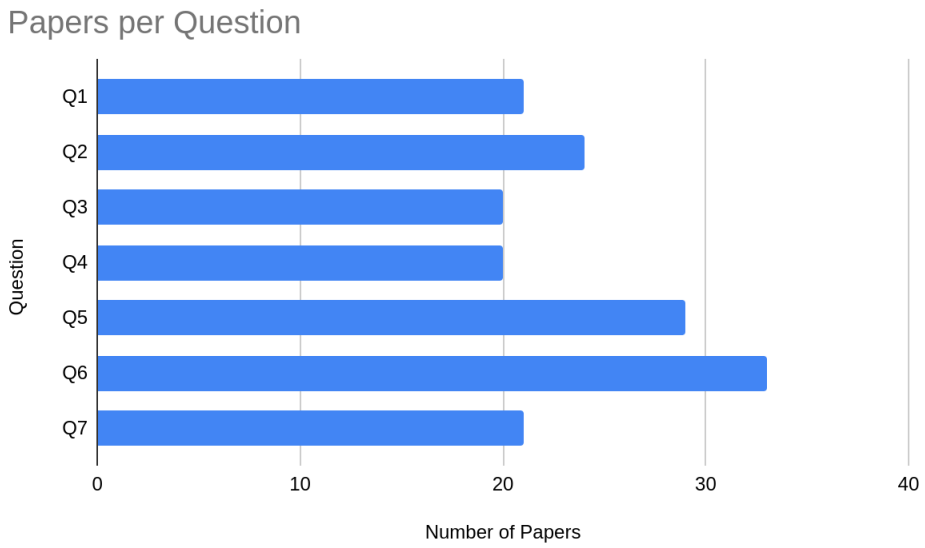

Figure 3.9: Number of questions answered per paper (Search on games and reuse approaches).

3.3.2 Summary of findings

Q1: What was the motivation for using SR?

It is common in the game area to have development problems due to its high complexity. These difficulties include low code reuse to develop a game and consequently low productivity in the development (evolution and maintenance) of game solutions (MARQUES *et al.*, 2012). There are various advantages of utilizing SR in application development. Among the most significant ones are reduced development time, increased software quality, cost savings, and scale development. Numerous businesses and development industries have already used these strategies into their apps to enhance their performance. Creating reusable systems is a difficult challenge that has sparked considerable attention in recent years, particularly in fields such as mobile computing and robotics. Nonetheless, it is not widely employed in the realm of computer gaming (AYALA *et al.*, 2021; GUO *et al.*, 2015a).

One of the main incentives for using SR principles is to reduce development time and costs. Consider the following scenario: a rehabilitation center uses games to aid treatment. Physiotherapy is classified into several stages, each having different particularities for each problem, and there are different types of physiotherapy. Consider the time and money it takes to build such a large number of games, some of which can be highly customized for a given physiotherapy. For this, numerous reusable forms have already been used in recent years in the production of games.

They can simplify and accelerate the development of game systems; these games can only be built once, with subsequent iterations serving as expansions of the original one (BOAVENTURA and SARINHO, 2017; GEORGIEV *et al.*, 2016; SIERRA *et al.*, 2019; ZHU *et al.*, 2016).

The gaming industry has risen significantly in recent years, owing to several new investments and innovations. These emerging technologies are frequently still under development, adding to the difficulty of development and introducing more flaws (ALBASSAM and GOMAA, 2013). This rise is occurring across all platforms, including PC, mobile, and console. This resulted in reuse, as the production of a game may be so time-consuming that businesses prefer to build a single game and then attempt to expand it (DERAKHSHANDI *et al.*, 2021).

Another reason for using reuse techniques in game development is the scarcity of game engines that support such approaches, and when they do, they sometimes lack the complexity required by advanced games or have a high development complexity, resulting in a steep learning curve (GUO *et al.*, 2015b).

Finally, code and component generation techniques can further accelerate the development of games, making them expand or even create from others already created more quickly (TANG and HANNEGHAN, 2013).

Q2: What were the reuse methods used?

Developing games may be a time-consuming and error-prone process. Much has been stated in recent years about the need of dealing with complexity, and several ways have been devised and tried to do this. Such settings encourage reuse, reducing the need to create software assets from scratch (SARINHO *et al.*, 2012). As with any other field of software development, the games industry is a unique domain that may benefit from a variety of SR approaches, including product line, domain modeling, components, and code-generating domain languages (FURTADO *et al.*, 2011).

The most prevalent method of reuse remains ad-hoc, comparable to opportunistic reuse, in which goods are copied and changed to generate varieties (also known as the clone-and-own approach). Clone-and-own enables low-cost variation development by reusing existing code. However, this strategy results in a large amount of duplicate code, making it difficult to maintain and grow the original code one (FENSKE *et al.*, 2017; FURTADO *et al.*, 2011).

Component-based development is a field of study that seeks to identify software components that may be reused in multiple applications and to construct these components in a reusable manner, developing once and reusing N times. This is the second most often used approach of SR for game development. Almost every

engine available today supports and distributes materials for game creation (ZHU and WANG, 2019).

SPL is a collection of strategies, techniques, and tools for developing comparable systems that provides a common core but has distinct characteristics. This method is advantageous for game expansion since it allows the separation of related aspects in order to adjust individual characteristics. Consider a game that can be played on both desktop and mobile devices. All components, including rules, objects, mechanics, and other aspects, are repeated, with the exception of visual elements, such as sprints and screen size (FENSKE *et al.*, 2017; LIMA *et al.*, 2019). DSPL is an extension of SPL that enables runtime modification of certain product line elements for self-adaptation (AYALA *et al.*, 2021). This method is well-suited for developing apps that are context-dependent and must adapt to their environment. It may be used to adjust game mechanics and rules in real time and to build new games on a regular basis.

MDD is a SE approach comprised of techniques for automating the development of software code from high-level abstract models. These models are intended to codify the structure, behavior, and needs of software applications that will be improved later (AOUADI *et al.*, 2016; BALDASSARRE *et al.*, 2021; GUO *et al.*, 2015a; SÁNCHEZ *et al.*, 2015; TANG *et al.*, 2013; ZHU *et al.*, 2016). By using current game engines for new game creation, this strategy bridges the gap between game concept and implementation. However, none of the MDD techniques described in the literature proved persuasively that their tools were well integrated (ZHU *et al.*, 2016).

Table 3.9 demonstrates the main reuse methods used for game development.

Table 3.9: SR methods used for game development

Reuse methods	**Papers**
Clone-and-own	ALBASSAM and GOMAA (2013); AOUADI *et al.* (2016); BALDASSARRE *et al.* (2021); BOAVENTURA and SARINHO (2017); DERAKHSHANDI *et al.* (2021); FENSKE *et al.* (2017); GEORGIEV *et al.* (2016); GUANA *et al.* (2015); GUO *et al.* (2015a,b); MARQUES *et al.* (2012); MORALES *et al.* (2011); SÁNCHEZ *et al.* (2015); SARINHO *et al.* (2012); SIERRA *et al.* (2019); TANG and HANNEGHAN (2013); VEGT *et al.* (2016, 2018); ZHU and WANG (2019); **?**

Component-based development	ALBASSAM and GOMAA (2013); AOUADI *et al.* (2016); BALDASSARRE *et al.* (2021); BOAVENTURA and SARINHO (2017); DERAKHSHANDI *et al.* (2021); FENSKE *et al.* (2017); GEORGIEV *et al.* (2016); GUANA *et al.* (2015); GUO *et al.* (2015a,b); MARQUES *et al.* (2012); MORALES *et al.* (2011); SÁNCHEZ *et al.* (2015); SARINHO *et al.* (2012); SIERRA *et al.* (2019); TANG and HANNEGHAN (2013); VEGT *et al.* (2016, 2018); ZHU and WANG (2019)
Software Product Line	ALBASSAM and GOMAA (2013); AYALA *et al.* (2021); LIMA *et al.* (2019); SARINHO *et al.* (2018); SIERRA *et al.* (2019,?); ZHU and WANG (2019)
Model Driven Development	AOUADI *et al.* (2016); BALDASSARRE *et al.* (2021); DERAKHSHANDI *et al.* (2021); FURTADO *et al.* (2011); GUANA *et al.* (2015); GUO *et al.* (2015a,b); KHORRAM *et al.* (2021); MARQUES *et al.* (2012); NÚÑEZ-VALDEZ *et al.* (2017); SÁNCHEZ *et al.* (2015); TANG and HANNEGHAN (2013); TANG *et al.* (2013); THILLAINATHAN (2013); VALDEZ *et al.* (2013); ZHU and WANG (2019); ZHU *et al.* (2016)

Q3: What are the advantages of the methods used?

The majority of organizations begin by copying whole systems and customizing them to new client requirements. While this technique yields a high rate of return in the short term, it may result in the propagation of flaws and challenges with scalability and maintainability (FENSKE *et al.*, 2017; SÁNCHEZ *et al.*, 2015). Therefore, despite being one of the possible uses of SR, this approach is not recommended due to the disadvantages caused by it.

In recent years, several reusable methodologies have been used to game production. They have the potential to simplify and speed manufacturing by concentrating on modeling, components, product lines, and reusable components (BOAVENTURA and SARINHO, 2017). Despite this, the gaming business had platform variability problems as a result of the enormous volume of games and the requirement to build them for a variety of platforms, including mobile, console, and computer (DEBBICHE *et al.*, 2019; SÁNCHEZ *et al.*, 2015). On the basis of this problem, it is able to demonstrate one of the primary benefits of the SR idea, which is increased scalability. As previously stated, the same game must support many platforms. For instance, the product line approach may be utilized to produce these games for a

variety of platforms. The visual components and screen sizes would be customized for each platform, but the remainder would be reused, decreasing development time and expense (GUO *et al.*, 2015a; LIMA *et al.*, 2019). Another feature of the product line is the ability to create modifications. If the original game was developed with expansion in mind, it is feasible to establish a product line for it in which all of its characteristics are malleable, and to create a new game, all that is required is to tweak one of the product line tree's characteristics while reusing everything else. As a result, the product line may be a way to achieve a compromise between the requirement for game versions, component reuse, and manufacturing costs (AYALA *et al.*, 2021; RINCÓN *et al.*, 2018; SIERRA *et al.*, 2019).

Apart from the product line, other techniques provide a number of advantages when utilized. For example, MDD enables quick prototyping of system variations, which may be quite beneficial during the early stages of game development or while creating exergames [1] (BALDASSARRE *et al.*, 2021). However, this approach has also been used for serious game development, as it has complexity management, providing different levels of abstraction and a good level of automation and code generation (KHORRAM *et al.*, 2021). In addition, MDD also reduces the requirement of knowledge and experience due to its abstraction, making complex rules explicit directly in the abstraction without the need for domain on the part of the developer (ZHU and WANG, 2019). MDD is not a silver bullet. Many factors can influence the gain in productivity brought about by approaches involving MDD. The decision must be careful on whether or not to adopt MDD according to various aspects of the domains/projects (GUO *et al.*, 2015b). MDD, in general, offers outstanding benefits to its practitioners, including higher productivity, interoperability, portability between multiple platforms, support for document generation, simplicity of maintenance, and increased software quality, which increases product reliability. Additionally, the ability to encapsulate technical components of creation decreases the obstacles that prevent non-technical individuals from producing games (GUANA *et al.*, 2015; TANG and HANNEGHAN, 2013; TANG *et al.*, 2013; THILLAINATHAN, 2013; VALDEZ *et al.*, 2013).

Software components can also bring advantages to those who use them. It is possible to partition the program into components that may be utilized in several applications, hence decreasing the application's development time and cost. This is a strategy that developers and contemporary engines frequently employ (GEORGIEV *et al.*, 2016). Additionally, once numerous components are completed, the original expense in developing a game will be lowered incrementally as new components are created and utilized (VEGT *et al.*, 2016).

Table 3.5 demonstrates a summary of the advantages of each reuse method found.

[1] Exergames: games for physical training (DJAOUTI *et al.*, 2011).

In this table, three colors are used with different meanings: red for when there is no advantage, yellow for having the advantage in a small or medium degree, and green for when there is a degree of advantage.

Table 3.10: Advantages associated with the use of reuse techniques.

	Clone-and-own	**Component-based development**	**Software Product Line**	**Model Driven Development**
Lower initial investment	X			
Lower investment for expansion		X	X	X
Maintainability		X	X	X
Scalability		X	X	X
Shorter development time	X	X	X	X
Non-technical development				X
less complexity		X	X	X
Prototyping	X			X

Q4: What are the difficulties of the methods used?

According to some experts, game makers continue to adopt clone-and-own strategies to adapt their video games to multiple platforms and users. According to them, the industry need tools/frameworks that enable the development of a multiplatform video game software product line (ALBASSAM and GOMAA, 2013). Implementation difficulties may be the factors that made these tools/frameworks not yet widely created and used.

Despite the numerous benefits indicated in the preceding question, there are a few drawbacks for employing SR in game creation. For instance, establishing reuse environments such as SPL takes a substantial upfront expenditure. As a result, corporations frequently develop product versions using opportunistic reuse techniques such as clone-and-own. However, this method leads in little code reuse and limited code scalability, making complex games both expensive and time-consuming to design (FENSKE *et al.*, 2017; ZHU and WANG, 2019).

Building games may be difficult, as they need the specification of several architectures, rules, behaviors, and gameplay features, among other things. This can be a problem when reuse is required, as this piece must be flexible enough to be reused in a variety of settings while also simplifying implementation through its abstractions (GUANA *et al.*, 2015).

Finally, another difficulty presented by some scholars is the lack of tools supporting reuse techniques. Currently, several engines support game development. However, few of these tools provide extensive support for reuse techniques, except assets/components and the few that provides access has a high learning curve, which makes it challenging to develop and learn.

Q5: How were the methods applied?

It was possible to determine through a review of the literature that there are some solutions to the productivity and portability challenges connected with games. Numerous ways have been developed to control the heterogeneity of the game domain between instances and strategies. SPL was the first approach identified, with the goal of identifying common application components and specifying only those that vary among applications. This procedure is carried out in three stages: domain analysis, variability analysis, and application. Domain analysis is used to collect and assess information about the application domain in order to get the desired outcomes for the product line. Typically, a domain analysis results in four deliverables: a domain definition, domain terminology, descriptions of domain concepts, and explanations of domain ideas' commonality and variety, as well as their interrelationship (GUO *et al.*, 2015a). Variability analysis is used to determine which components of the product line require modification depending on the results of a domain study. This is the stage at which generic components of the product line are created. The application stage tries to categorize all related qualities for the purpose of building the SPL and specifying and expanding it in the future. However, the majority of existing techniques focus on separating comparable games and then use all of their attributes to create the SPL, which results in a lot of work (ALBASSAM and GOMAA, 2013; RINCÓN *et al.*, 2018).

MDD segregates conceptual and technological issues, allowing designers and developers to focus only on the product's underlying logic. This separation seeks to improve the system's quality and minimize the time required to design it, while also assuring interoperability, reliability, and performance (SÁNCHEZ *et al.*, 2015). This technique is composed of three components: definition, application, and analysis of variability. The definition establishes the domain and creates or selects the appropriate tools, which may include editors, domain-specific libraries, and code

generators. The application defines domain-specific language (DSL) models and creates code using those models. The development process takes a high-level abstraction model as input and applies N transformations until it produces a low-level abstraction model from which code is created (BALDASSARRE *et al.*, 2021; GUO *et al.*, 2015b; SÁNCHEZ *et al.*, 2015; TANG *et al.*, 2013). As previously noted, these models require N modifications before they may be used. This model may have a variety of names depending on its abstraction level. The first model developed is referred to as the Computational Independent Model (CIM), and it provides the generic information about the model. This model will be enhanced to become a Platform-Independent Model (PIM), which will incorporate more intricate rules. Finally, the PIM model is converted to a Platform-Specific Model (PSM), which provides information on the platform on which the game will run. The game code may be generated according to the PSM level (AOUADI *et al.*, 2016; DERAKHSHANDI *et al.*, 2021). Finally, the variability analysis indicates the potential changes between all models developed (GUO *et al.*, 2015a,b).

The following steps are involved in component development: separation of similar codes, generic writing of codes in the form of components, creation of the asset repository, and use of components. The first step aims to search within all applications for the codes that are rewritten N times, thus the possible codes to be componentized. Then each of these codes is analyzed to create reusable components. Finally, an asset repository is created to store the created components for future use (GEORGIEV *et al.*, 2016; VEGT *et al.*, 2016).

Finally, approaches like clone-and-own operate at the granularity of entire systems. This approach does not have many large steps. It only has two steps: the first one would be to clone the original application and the second one would be to modify the code (DEBBICHE *et al.*, 2019).

Q6: What are the requirements for creating games using reuse?

Developing a reuse approach requires defining a specific domain or market segment without trying to cover all possible domains. The first task performed is to define the problem, objective and purpose that are sought with the game to be built. Then, a domain analysis of the existing games and their internal characteristics is carried out so that it is possible to group their similarities (NÚÑEZ-VALDEZ *et al.*, 2017; THILLAINATHAN, 2013).

There are several characteristics that need to be defined in order to be able to use a specific reuse approach. Among them, the following stand out: technical characteristics, forms of use, flexibility, and simplicity of the code to be generated (GEORGIEV *et al.*, 2016).

The characteristics defined above can be understood as structural characteristics. However, there are other elements that need to be defined before the games are built, among which the main ones are: game objects, stage scenery, behavioral properties, and gameplay, among other characteristics. Table 3.4 demonstrates each of these elements, being the same for building mods (BOAVENTURA and SARINHO, 2017; DERAKHSHANDI *et al.*, 2021; GUANA *et al.*, 2015; RINCÓN *et al.*, 2018; SARINHO *et al.*, 2012, 2018; TANG and HANNEGHAN, 2013; ZHU and WANG, 2019).

Q7: What were the tools used?

Various tools have been used to facilitate game creation by providing easy user interfaces and code generation. But these tools are often domain-specific, not comprehensive, and often only support simple games. Furthermore, these game engines are often huge and complex. This makes the learning curve steep and the cost of usage quite high, even though most game engine features are not used (GUO *et al.*, 2015a).

Game engines are currently the biggest achievement in terms of software reuse in the gaming domain. However, current engines like Unity, XNA and Unreal currently do not use approaches like MDD and SPL directly, supporting only components and assets. Also, as mentioned, these have a high learning curve and low support for people with no programming experience (ALBASSAM and GOMAA, 2013; AOUADI *et al.*, 2016; NÚÑEZ-VALDEZ *et al.*, 2017; ZHU and WANG, 2019). Other engines, like Construct, GameSalad and GameMaker, have a low learning curve and support users with no programming experience. However, games built with these engines are simpler than those that can be built with the previous ones (NÚÑEZ-VALDEZ *et al.*, 2017; RINCÓN *et al.*, 2018).

There are also developers who do not use any game generation tools, using only IDEs such as IntelliJ and FeatureIDE to facilitate their interaction in building the game (ÅKESSON *et al.*, 2019).

All the tools mentioned above only support software componentization directly. However, there are other tools that support other types of reuse approaches such as MDD and SPL. However, these tools are not yet used on a large scale. The following table shows the tools that were found in the review and how they can be used for different types of reuse.

Table 3.11: Tools used for game development.

	CBD	SPL	MDD	**Papers**
Unity3D	X			ALBASSAM and GOMAA (2013); AOUADI *et al.* (2016); DERAKHSHANDI *et al.* (2021); NÚÑEZ-VALDEZ *et al.* (2017); SIERRA *et al.* (2019); TANG *et al.* (2013); THILLAINATHAN (2013); VALDEZ *et al.* (2013); VEGT *et al.* (2016, 2018); ZHU and WANG (2019)
Unreal	X			DERAKHSHANDI *et al.* (2021); TANG *et al.* (2013)
XNA	X			DERAKHSHANDI *et al.* (2021); TANG *et al.* (2013); VEGT *et al.* (2016)
Construct	X			RINCÓN *et al.* (2018)
GameMaker	X			DERAKHSHANDI *et al.* (2021); NÚÑEZ-VALDEZ *et al.* (2017); VALDEZ *et al.* (2013)
GameSalad	X			DERAKHSHANDI *et al.* (2021); NÚÑEZ-VALDEZ *et al.* (2017); VALDEZ *et al.* (2013)
Stencyl	X			VALDEZ *et al.* (2013)
RAGE	X			VEGT *et al.* (2018)
PhyDSL			X	BALDASSARRE *et al.* (2021); GUANA *et al.* (2015)
SeGa4Biz			X	KHORRAM *et al.* (2021)
SeGMEnt			X	TANG *et al.* (2013)
FEnDiGa			X	SARINHO *et al.* (2012)
MEnDiGa			X	BOAVENTURA and SARINHO (2017)
AsKME		X		SARINHO *et al.* (2018)

3.3.3 Conclusion

This study was an extension of the study in Section 3.2 that sought to identify how reuse approaches are being used to support game development. This study revealed that there are four main reuse approaches in game development, including componentization, MDD, SPL, and clone-and-own. Each of these forms has its own unique features and applications, but they can also be used together to get the most out of them.

The analysis indicated that the great majority of publications emphasized the benefits of code reuse in the construction of games, including reduced cost and development time, increased quality, the ability to expand the game, among others. However, a significant percentage of them highlighted challenges in establishing reuse settings, suggesting that it is a topic worth investigating but requiring a lot of work.

Numerous technologies, including frameworks, methodologies, and engines, were discovered that facilitate the building of games through reuse. However, these tools are largely in the experimental stage or are usually associated with certain problems, such as a steep learning curve, complicated implementation, and the construction of only simple games. Among the tools identified, the most often used strategy to reuse was componentization, which is still utilized in modern engines. Additionally, five tools support MDD, but just one tool supports SPL. However, when it comes to these latest approaches, the tools are still in the experimentation stage.

Another problem that is worth mentioning is that there are few tools that support development guided by reuse and that, in addition, these tools need a scope for creating the game; that is, it is only possible to reuse the artifacts if the games have well-defined characteristics.

Additionally, it was possible to observe that the process of creating a game may be rather difficult, including numerous phases and requiring knowledge of several aspects, including rules, characters, and the console on which the game would be conducted, among others.

Based on what has been described so far, it is possible to notice that the use of Software Reuse to build games is something that can generate many benefits, but that currently there are few tools that support this approach.

Chapter 4

Exploratory Studies

This chapter aims to introduce the problem that this work seeks to solve, as well as the initial prototypes that were made to validate the idea.

4.1 Problem discussion

Several reasons lead a game user to create a Mod. Among the main ones are: trying new things, solving bugs, creating new characters, increasing the difficulty of the game, gaining advantages about the game, increasing the game life cycle, among others (LEE *et al.*, 2020). As previously stated, little information on modifiers has been found in the literature, which is why mods were sought. However, because the two have similar terms, it is assumed that their motivations, disadvantages, and advantages are also similar.However, the term mod will be used to facilitate understanding.

Mods, as well as games, are complex and take time to develop. The time it takes to build a mod can vary a lot. Being built in a few days, or taking a long development time. Depending on the nature of a mod, it may require only one or more releases. For example, a mod that improves a texture in a game may need only one version. On the other hand, a mod that does a full review may require one or more releases (LEE *et al.*, 2020) (HOFMAN-KOHLMEYER, 2019).

There are several ways to create a mod, among the most common ones are: having access to the game's original code, be it open source or provided by the game's producer, using reverse engineering, or even modifying the game through a platform provided by the producer as, for example, World of Warcraft provides a UI customization tool that allows add-ons to reconfigure the user interface panel (LEE *et al.*, 2020; WALT, 2010). However, this UI customization is limited, allowing only to edit some characters, create new maps and minor modifications.

Through the studies carried out, it was possible to observe that the community has already been using forms of reuse to build games. Game development through

existing ones is known as mods and is applied on a large scale by the community and even by companies that aim to improve their games or increase their game list. However, although mods have become a popular way to express oneself in the community, they are still made in an ad-hoc manner and are often compared to opportunistic reuse, being performed most of the time through clone-and-own. As seen in the search, there are additional methods for developing games with SR, such as software componentization, which makes them accessible in the most popular game engines. However, there are different approaches, such as SPL and MDD are still in testing phases, with few tools available for support. In this study, only one tool was found for supporting SPL approach.

The use of reuse in software development may supporting various benefits, including a reduction in development time, an improvement in software quality, and the opportunity to expand the product more quickly and easily. Looking at these outstanding advantages, it is possible to connect these benefits to the challenges of game production that are: long development time, risk of numerous defects, and complexity of generating new versions of the original game.

There are several approaches for using software reuse, from clone-and-own to product lines. Each of these approaches has different advantages and purposes of use (KRUEGER, 1992; MORISIO *et al.*, 2002). Next, each of these will be discussed, focusing on game development. It is worth remembering that all the following observations were made keeping in mind the development of mods and game expansion.

- **Clone-and-own:**
  - **Context of use:** this approach might be adopted for small projects in which an extension of the game is not initially planned.
  - **Advantage:** this strategy delivers an excellent initial return by reusing a large part of the existing game.
  - **Disadvantage:** future maintenance and updates could be costly if the same changes and improvements are made to many projects at the same time.
  - **Example:** take an open source game and modify specific features.
- **Componentization:**
  - **Context of use:** this approach is applicable for medium to large projects [1]. The expansion of the original game is not planned in advance, but

[1] A project is considered 'small' if line of code (LOC) of the project is between 10.000 and

by reusing the components, new games may be produced. Its primary purpose is to construct new games by reusing its components.

  - **Advantage:** this method is already prevalent, as it is employed by several engines. If a modification is made to a component, all games using that component will receive the update.
  - **Disadvantage:** does not allow expansion of games directly.
  - **Example:** in a (First-Person Shooter) FPS game, for instance, all movement and a significant portion of the actions may be componentized because they will be the same in all games.

- **MDD:**
  - **Context of use:** this approach is applicable for medium to large projects. The original game's expansion is not preplanned. This approach is well used because its implementation only starts after a well-elaborated model is created after many iterations.
  - **Advantage:** platforms that are based on this development process allow full games to be created by anybody with limited programming skills.
  - **Disadvantage:** unlike the others, it is not a development pattern, but a development process.
  - **Example:** use the Model-Based Development process in projects with poorly defined requirements and with a great possibility of modification due to the possibility of modifying the models before actually starting the project.

- **SPL:**
  - **Context of use:** this approach is applicable for medium to large projects. The expansion of the original game is intended earlier.
  - **Advantage:** several games can be derived from a single game by selecting game features.
  - **Disadvantage:** few tools and platforms available.
  - **Example:** a simple use of the product line in game development enables the creation of games for many sorts of devices. Consider a game that requires a mobile and desktop version. Using product lines, the rules,

40.000, 'medium' if LOC is between 40.000 and 100.000 and 'large' if LOC are typically over 100.000 (QURESHI, 2012). But this is considered just an estimate, remembering that several factors can still be considered, such as people involved, difficulty, criticality development time, among other factors.

mechanics, and other game pieces will be utilized, and just the sprites will be modified to accommodate multiple screen sizes.

Based on the benefits, goals, and settings of the highlighted reuse approaches, it is possible to deduce that there are several ways to use reuse to create games, depending on the context and the amount of work spent. Some of these approaches are less methodical than others, such as clone-and-own, others are already being used as is the case of components, and others are still in the experimental phase, such as MDD and SPL.

Based on what has been described up to this point, it is possible to note that the clone-and-own strategy is not advised owing to its lack of opportunities for game evolution. Componentization is an effective method for developing high-quality software and has already been implemented in the largest gaming engines. However, when it comes to game evolution, this method alone is not enough, because of the requirement to comprehend the component and its evolution. This technique is best suited for code reuse in multiple types of games rather than evolution game or development of mods. The MDD approach aims to manage any software engineering process in which models are built and refined. At a high level, the MDD concept is similar to the low-code concept in that there are pre-made models that can be used and modified by the programmer. However, when considering only evolution and creation of mods, this approach proves to be more limited, allowing to manage only one mod at a time. As a result, SPL is the most recommended reuse strategy for the creation and evolution of games and mods, allowing vertical selection and evolution of features, where these can be chosen to generate the final product, allowing addition or removal of features, and allowing for the creation of N versions of the game at once. However, as previously stated, all of the platforms found in this review that supported the SPL approach were in the testing phase and required a significant amount of work to build, even though they were still specific to a certain type of game, which reduced the possibility of generalization of the approach.

Another common problem in the gaming field is the representation of game features. In the literature, there are numerous game representation approaches that present elements and behaviors, such as the elemental tetrad NETO and TAYLOR; SCHELL (2008); UNGER (2012) and the ORC framework GUARDIOLA (2016). However, these frameworks are not yet integrated into any tool for automatic representation, being still a challenge for the gaming community (SARINHO *et al.*, 2012). In light of this, it is possible to imagine a visual tool for developing games using SPL idea, where the game to be made would be represented by a tree and the developer could view the game's characteristics and generate different versions of it.

As previously mentioned, there are N methods to describe the characteristics of a game, and one of the most well-known is the elemental tetrad model, which

illustrates the mechanics, second level mechanics, and aesthetics of a game's characteristics. Considering that it is possible to describe a game using these elements, it is plausible to assume that a new game will be formed if a mutator is applied to one of these elements (mechanics, second level mechanics and aesthetics). It is worth remembering that a mutator can be understood as a small modification that can be applied to a characteristic of a game to modify it NETO and TAYLOR; UNGER (2012). For example, to modify a game's character movement mechanics, just apply a mechanic mutator that modifies this movement, for example, increasing it whenever it is close to an enemy. Based on this, the elemental tetrad framework was combined with the SPL and mutators approach to develop a new game engine. With this, each edge would represent a mutator applied to a game, and each new node would represent a potential feature. As elemental tetrad is partitioned into four parts, it is possible to imagine that 3 feature trees will be needed (discarding technology), one for each part of elemental tetrad. Figure 4.1 demonstrates an example of a mechanics tree, it is possible to observe the original game with all its mechanics and two mods that were made from it, adding mutators and excluding some characteristics. For example, mod one was created by removing the possibility of a machine gun, increasing walking speed and increasing the number of shots from the default weapon.

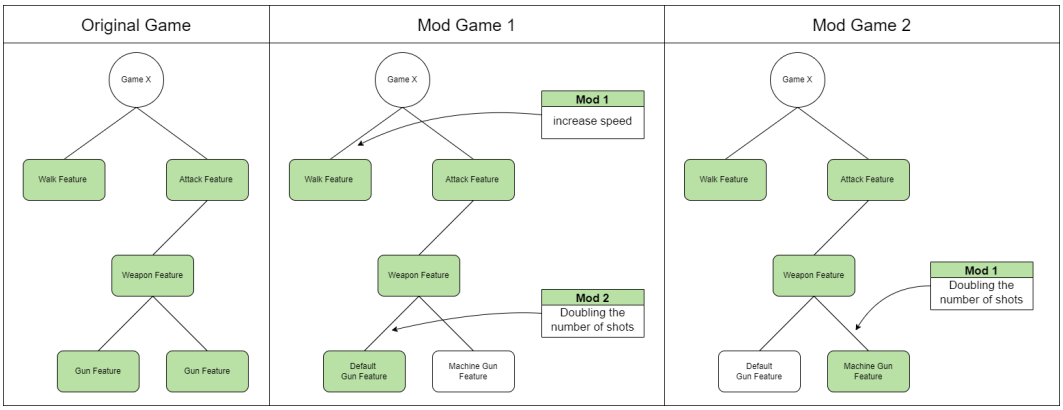

Figure 4.1: Illustration of a feature derivation tree.

On the basis of this concept, it is possible to imagine an SPL-based game development platform that will use this concept. Through it, it would be able to code according to a specific pattern, and all of the game's features would be represented as a tree of features. Thus, it is possible to see the game's features and then alter or choose the features of a potential mod, letting the developer create the game once and then change it directly through the features tree. The platform aims to provide facilitators that seek to accelerate the development of mods, such as: the feature tree

that separates visually the game into all its features, projects patterns that must be followed and mutators that can be used to modify the created games. It is expected that the usage of this platform would reduce redo work, increase the number of mods developed, and accelerate the process of generating mods and games.

All previous chapters concentrated only on theoretical studies demonstrating difficulties and potential solutions. Using just the author's ideas and existing literature as a foundation, no instrument was created to verify whether or not such a concept was viable. Therefore, some proofs of concept were carried out to verify the viability of building the platform described above. Next, they will be described in more detail.

4.2 Proof of concept

The main point of this section is to show the games and ideas that were developed to see if the initial idea from the previous section could be built on.

As discussed in the last session, the product line was the main reuse strategy chosen for study in this work. The purpose of this was to establish a development platform that facilitates the creation of games using this method. However, prior to the creation of the platform, research was conducted to determine how this strategy would be implemented. In this section, four proofs of concept will be demonstrated that aim to validate the possibility of building the platform.

As stated in Chapter 2, games have several types of definitions, and one of them is that they can be understood as a simulation of a real case (XEXÉO *et al.*, 2013). This work aims to build a platform to assist in game development. However, before the final platform is constructed, prototypes were develop as a proof of concept regarding the viability of building the platform. Based on these premises, first, the construction of games will be used to simulate the possibilities of creating the platform due to the ease of development by the author.

4.2.1 Initial exploration

The first prototype built aimed to validate the difficulty of working with game mutators, that is, to validate how game characteristics could be parameterized to be modified, resulting in configurable games. As it is the simplest prototype to be developed, it was created using the GameMaker Studio 2 and with the help of one of the students of a Software Reuse course. Being the only prototype that was built with assistance. All other prototypes were built from the Unity engine, due to being more elaborate prototypes.

The prototype developed aimed to create games based on the configurations

of the application's three JSON (an JSON representing a model for each part of elemental tetrad). From the application it is possible to develop several platform games, running at 60 frames per second, depending on the settings that are chosen within the JSON files. Within the JSON files there are several attributes to be configured, below each file will be demonstrated in more detail.

- Mechanics
  - Player's weapon (bool)
  - Enemy's weapon (bool)
  - Player's score (bool)
- Aesthetics
  - Weapon sprite (int)
  - Player sprite (int)
  - Enemy sprite (int)
  - Scenario sprite (int)
- Second level mechanics
  - Player speed (int)
  - Player's HP (int)
  - Player jumping force (int)
  - Enemy speed (int)
  - Enemy's HP (int)
  - Enemy jumping force (int)
  - Weapon force (int)
  - Weapon speed (int)
  - Weapon type (int)
  - Gravity

Figure 4.2: Configuration JSON example.

As previously stated, the games are built using three configuration files: the mechanics file, which controls which additional mechanics the game will have (Boolean variables), the second level mechanics file, which controls the strength and speed of the game objectives (integer variables), and the aesthetic file, which controls the visual aspect of the game (integer variables that point to a file in a folder with the sprites). It is worth mentioning that the prototype could be evolved to have much more mechanics and second level mechanics, however, with the few options that were implemented, it is already possible to develop many versions of platform

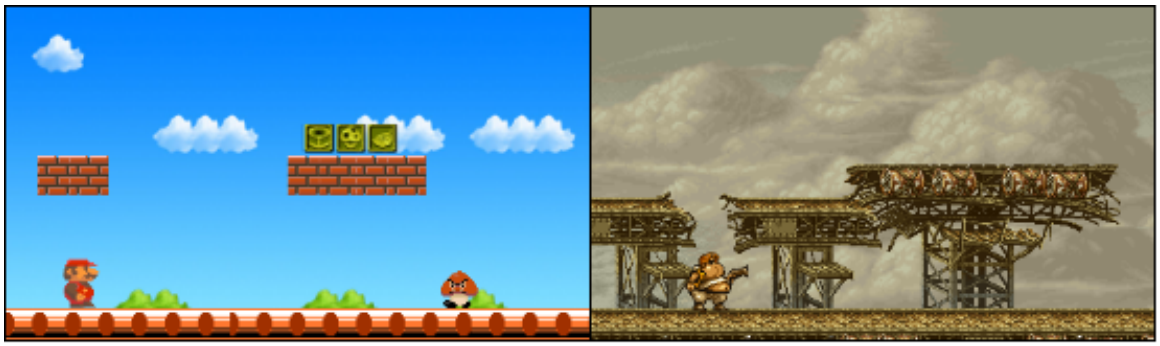

Figure 4.3: Games created from the prototype.

games. Figures 4.2 and 4.3 demonstrate the game configuration files, one for each part of elemental tetrad, and games that were created from the prototype. From the first figure you can see early games that were created using the sprites of mario and metal slug.

Although it was only a prototype for creating very simple platform games without any animation, with static sprites and without elaborate mechanics, from the prototype it was possible to observe that it would be possible to parameterize the features of the games and to develop different games by modifying only a few variables. Thus, it was possible to move on to the second prototype that sought to verify the difficulty of developing a product line for a specific game.

4.2.2 Developing a product line

For the creation of the initial product line, the Codeboy game was used, which was built within a master's dissertation (CASTRO, 2020). Codeboy was a mod development experience in which a Lightbot adaptation was created with the goal of teaching programming fundamentals through games.

Lightbot's mechanics, second level mechanics, and aesthetics, as well as the game's background, were changed for the construction of Codeboy in order to teach reuse ideas. Next, each of the modifications based on the elemental tetrad will be described.

- **Mechanics:** character movement, drag-and-drop actions, and sequence auto-execution have all been intact. New mechanisms were added for catching the star and opening the treasure's chest.

- **second level mechanics:** the second level mechanics of develop functions were updated in accordance with the elemental tetrad to enable the usage of

FODA[7] (KANG *et al.*, 1990)decision tree, which is a common model in the SR domain.

- **Aesthetics:** the game's aesthetics were kept, but new sprites were used to alter the visual aesthetics.

Figure 4.4 shows the Lightbot and Codeboy creation trees. It can be seen that the Lightbot game received two mutators over the Codeboy game was generated, and that each new node in the tree represents a new game.

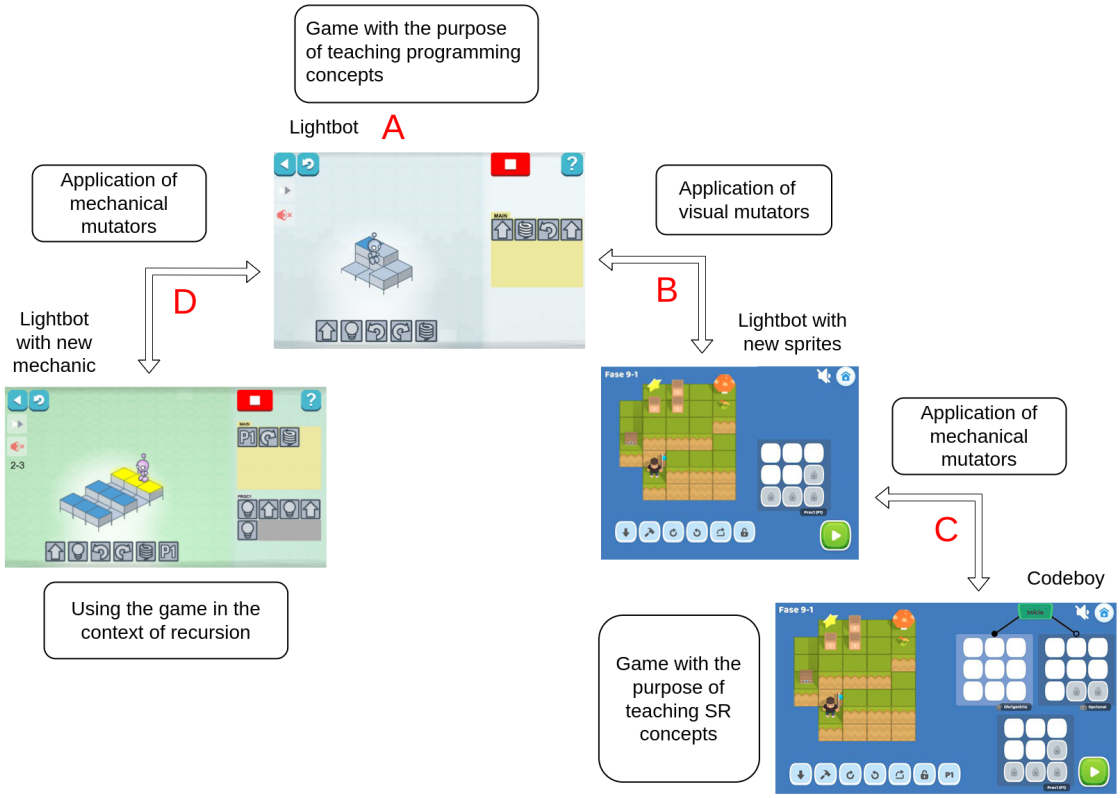

Figure 4.4: Tetrad Generation Game (CASTRO and WERNER, 2021; GOUWS *et al.*, 2013).

From figure 4.4, it is possible to perceive the application of mutators to create games. Each of the nodes of the tree that can be interpreted as new games is represented by a letter (A, B, C, or D) in the figure. The tree begins with letter A, illustrating the original lightbot game. This node allows the application of visual mutators from the right and mechanical mutators from the left. Continuing along the left side, visual mutators were applied, which provided the game a new appearance with new sprites, thus arriving at node B. From this node, mutators of

[7]Feature oriented domain analysis (FODA): It can be understood as a model of a domain analysis that demonstrates which system characteristics are mandatory, exclusive, or optional(KANG *et al.*, 1990).

mechanics were applied, so developing Codeboy, a new game with new mechanics. Returning to node A, on the left, mechanical mutators were applied, establishing new characteristics for the game and transforming it into a new game designed to teach recursion. N mutators can be applied in each phase, however, some steps were omitted from the figure for the sake of simplification.

Table 4.1 compares the possible four games demonstrated in the product line according to the elemental tetrad framework. The same letters A, B, C and D will be used to represent the games. Aesthetics according to the elemental tetrad framework are the sensations that players can feel while playing, however, to change a game's aesthetics, N mechanics and second level mechanics mutators are needed to change such aesthetics. Therefore, the aesthetics of feelings for the three displayed games are identical: Challenge (take the character from point A to B), Submission (game is seen as a hobby for the user) and Expression (the player creates his/her own sequence of moves). From this table it is possible to observe that small changes in the mechanics, second level mechanics and aesthetics of a game can generate new games.

Table 4.1: Lightbot and Codeboy characterized according to the MDA framework.

	Mechanic	**Dynamic**	**Aesthetic**
A	Walk and turn. Turn on the light.	Create movement sequences.	Original Sprites.
B	Walk and turn. Turn on the light.	Create movement sequences.	New Sprites.
C	Walk, turn, push and break. Collect items (Star and chest).	Create movement sequences. Reuse functions. Choose to open the treasure's chest (optional).	New Sprites.
D	Walk and turn. Turn on the light.	Create movement sequences. Reuse functions.	Original Sprites.

The second prototype used the game developed for the master's degree as a use case to demonstrate how a game product line would be built. Despite the fact that it was a previously built game, it was possible to reuse its code to build a product

line in which features were rebuilt, modified, and added. This study revealed that it was possible to separate the characteristics of the games into smaller parts, allowing us to move forward with the study in one more stage.

4.2.3 Dynamic Tetrad Game

After demonstrating that it is possible to parameterize game settings via the first prototype and separate game features into smaller parts via the SPL of prototype 2, it is time to create prototypes for community validation.

As previously mentioned, any game can be explained through its mechanics, second level mechanics, and aesthetics. In light of this, it can be asserted that if each of these elements is modified, a new game can be create. Thus, a game prototype based on the elemental tetrad was constructed, in which modifications are applied to the game at runtime, producing new mods periodically.

The built prototype applies the concept of dynamic product lines to mods. As a prototype, this technique was utilized to construct the game due to its higher efficiency in arranging future modifications, since as the game does not need to be reconstructed each time a change is made. The characteristics of the classic product line are developed during the design phase, however, for the development of the prototype Dynamic Software Product Line (DSPL) (AYALA *et al.*, 2021) was chosen, due to the ease in coordinating the changes within the game.

The game was designed to simulate a DSPL, with each tetrad component changing during the course of the game. Creating a mod involves adding one or more mutators to a game. Therefore, it is plausible to believe that there are mutators, one for each part of tetrad architecture, and that these mutators are used to build new modifications for games. Further abstracting this concept, it is conceivable to assume that tetrad modifiers applied sequentially may yield each of the categories of mods outlined in Section 3.2. Using an aesthetics modifier, for instance, might suffice to construct an interface customisation hack. To make mutators or tweaks, it would be essential to employ a second level mechanics mutator; for a complete conversion, N mutators of all three dimensions would be required.

Elemental tetrad utilizes the term aesthetics to describe to the feelings experienced by the game's player, such as creativity, challenge, and community, among others. It is said that the game provokes specific feelings. However, the player receives these experiences in relation to cause and consequence. It is believed that the game's mechanics and second level mechanics create a sensation for the player. Due of this cause-and-effect relationship, it is difficult to create novel feelings while developing modifications. When rules and second level mechanics are modified, feelings may alter, but there is no one modifier that provides a new aesthetic. Therefore, in

this work, the aesthetic component of the game will be considered solely in terms of its visual aspect, i.e., sprites, maps, on-screen objects, etc.

The game functions like an infinite runner (CASTRO and WERNER, 2021). Each time the player reaches a checkpoint (marked by a flag), a new random combination of game mechanics, second level mechanics, and aesthetics is generated, thus producing a new version of the game at runtime. Figure 4.6 shows four phases generated at random by the game. Each time the player meets a flag, a new type of random mod is produced. To visualize the specified attributes, the game displays a panel displaying the selected configuration. Figure 4.6 shows the panel. From this figure it is possible to observe all the characteristics that are being modified throughout the game. Table 4.2 describes each of the features in more detail.

Figure 4.5: Elemental tetrad Generation Game (CASTRO and WERNER, 2021).

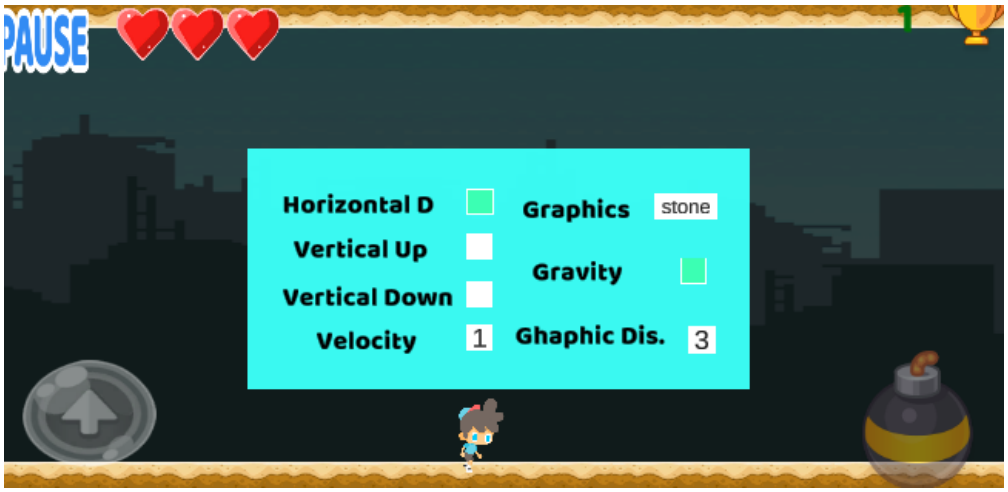

Figure 4.6: Configuration panel (CASTRO and WERNER, 2021).

Table 4.2: Changeable game features;

Characteristic	**Explanation**
Movement type	Controlled through 3 booleans. **Horizontal:** The character will run horizontally. **Vertical up:** The character will run from top to bottom in a vertical position. **Vertical down:** The character will run from bottom to top in a vertical position.
Velocity	It is controlled through a number with 3 values (1, 2 and 3) that creates a vector with fast, medium and low speeds.
Graphics	The game has 3 stages: Desert, Ice, stone and swamp.
Gravity	It is controlled through a boolean that defines whether or not the character will float.
Graphics Display	It is controlled through a number with 3 values (1, 2 and 3) that creates a vector of distance between the objects on the screen. Control through which the game will check if there are more or less spaced objects.

Table 4.3: Game mechanics, dynamics and aesthetics.

Mechanics	Run and jump or fly, destroy objects with bombs, destroy objects by clicking on the object, lives, score, pick up coins to buy new levels, lose lives by hitting objects
Dynamics	Increase or decrease speed, increase or decrease gravity, change character directions
Aesthetics or Interface	Fire environment Ice environment Earth environment Air environment

The game's properties are categorized in Table 4.3 according to the elemental tetrad. The required gameplay elements include lives, scores, money, and the loss of life when colliding with level objects. Note that any random mod will possess these characteristics. There is a boolean variable for each random game feature that determines whether it will be implemented or not. For each mechanism, for instance, the game will pick between walking, jumping, or flying, destroying items with bombs, and damaging objects by clicking, serving the same rule for the second level mechanics. Each step will pick a terrain type for aesthetic purposes.

The feature tree for full game functionality is demonstrated in Figure 4.7, where

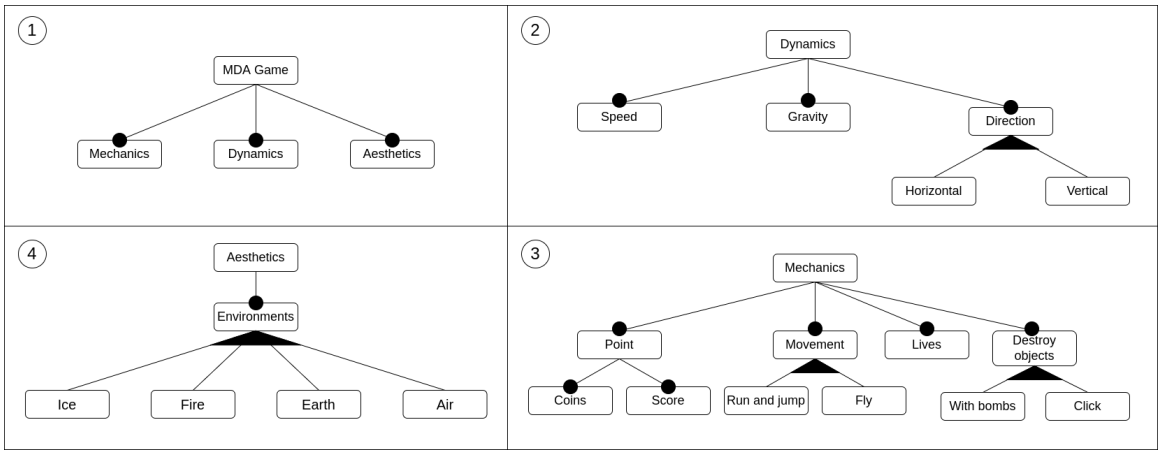

Figure 4.7: Elemental tetrad Feature tree (CASTRO and WERNER, 2021).

filled balls are required and empty balls are optional. Triangles with fill indicate that just one feature may be selected. This graphic depicts a model based on the FODA characteristics model (KANG *et al.*, 1990). To facilitate viewing, the graphic has been separated into the three components of elemental tetrad.

The prototype created aims to demonstrate the possibility of automating the process, having a product line where the original game is the core of the game's functionalities. The gamer itself could generate modifications to the game from the use of mutators according to the dimensions of elemental tetrad. From the initial prototype, it was possible to observe two distinct views of product line: a more traditional one where each node is a characteristic and to create the game it is necessary to navigate the tree and add the elements, and another where each edge of the tree can be a mutator that generates a new node when applied to an existing node, thereby creating a new game with different characteristics (this work aims to combine the two views).

4.2.4 Classic Tetrad Game

This work's first prototype aimed to develop an manual product line, which was derived from an original game built from scratch through reverse engineering with modifications made directly in the source code. The previous section described a prototype of a game that aimed to simulate a dynamic product line; this one was a little more challenging, but the changes made during runtime were easier to manage because the game did not need to be recompiled for each change. The game that will be described presently seeks to replicate a classic SPL, with the game being constructed throughout development, hence increasing a litle bit the implementation difficulties.

The game may be interpreted as a platform in which the protagonist must com-

plete the level's three tasks. At the beginning of the game, the player must select the characteristics he/she wants to incorporate to the game. There are three feature trees for this purpose, one for each level of elemental tetrad. Figure 4.8 demonstrates some levels generated by the game, from which it is also possible to see the objectives that were selected to be conquered. Figure 4.10 demonstrates the game's characteristics selection menu, having a characteristics tree for each part of elemental tetrad. This figure also displays the game's rules menu, which describes how to play. Figure 4.9 demonstrates the positioning of each of the elements of elemental tetrad in the game.

Figure 4.8: Levels generated by the game.

Figure 4.9: Game feature selection trees, grouped according to the elemental tetrad.

The game works as a product line where the player can choose each part of the game as its mechanics, its objectives, its enemies and other elements. In each game, the player will choose the characteristics he/she prefers and the game builder will

Figure 4.10: Game characteristics selection trees.

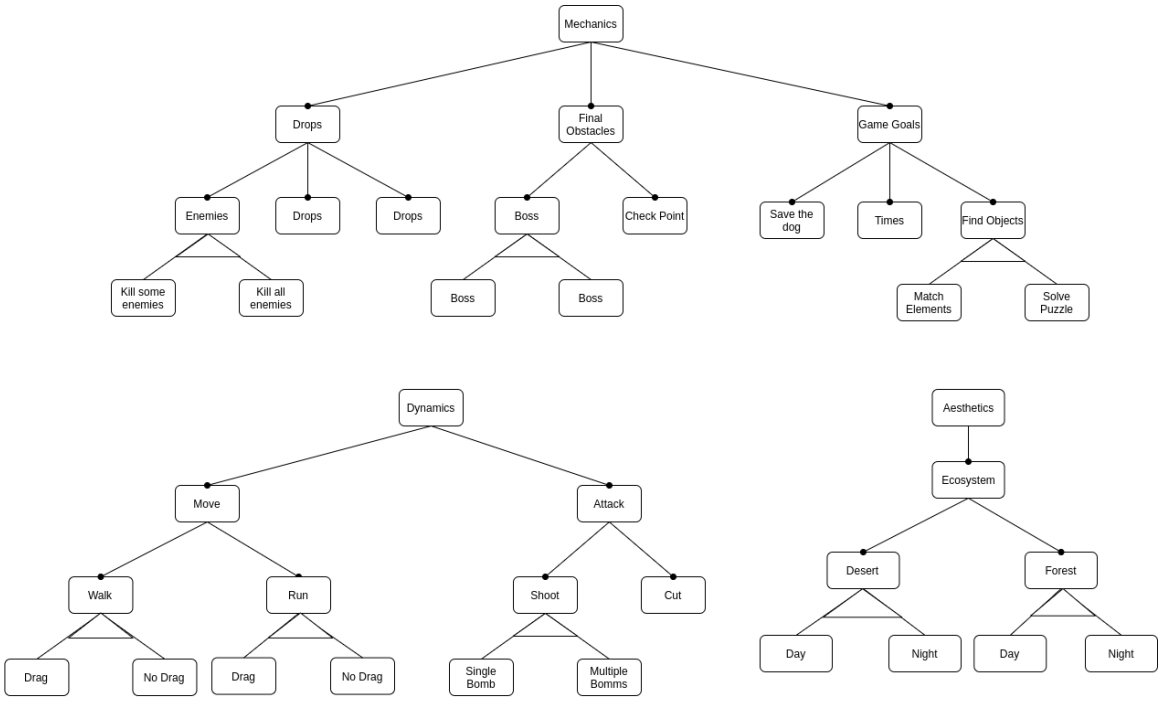

Figure 4.11: Tetrad SPL Classic game feature tree.

create a game with the selected characteristics. Next, each of the elements that can be selected or modified by the game will be described. Figure 4.11 demonstrates all three FODA (KANG *et al.*, 1990) trees with the characteristics that can be chosen.

Finally, to make the game more dynamic and difficult, a number of features were added, including characters with varying amounts of health, distinct attacks, and an A* algorithm to follow the character.

Elements that can be selected or modified:

- **Mechanics:**
  - **Drops:** drops in the language of games can be understood as the act of an enemy dropping an item in a match, however, in this game it was used with a sense of items or elements that are dispersed across the game.
    - **Enemies:** Choose if the player should kill all enemies or not.
    - **Coins:** Items that must be collected in the game.
    - **Lifes:** Items that must be collected in the game.
  - **Final obstacles:** obstacles to be overcome by the player.
    - **Boss:** choose if the game will have one or multiple bosses.
    - **Check Point:** choose whether the game will have checkpoints.
  - **Game goal:** main objective of the game.
    - **Save dog:** save the dog from the cage.
    - **Timer:** choose if the game should be won before time runs out.
    - **Find objects:** Select the items that the player must find in the game. There are two options: match elements, which must find the secret object, and puzzle solver, which must find similar objects.
- **Second level mechanics:**
  - **Move:** choose the second level mechanics of movement.
    - **Walk:** the character can only walk.
    - **Run:** the character can only run.
  - **attack:** choose the second level mechanics of attack.
    - **Shoot:** choose between the two possibilities of weapons, with one shot at a time or many shots.
    - **Cut:** choose weapon with sword option.
- **Aesthetics:**
  - **Ecosystem:** choose the game stage.
    - **Desert:** choose between day or night.
    - **Forest:** choose between day or night.

From this game, it was possible to expand the idea of building a game platform even further. This game attempted to demonstrate how the platform's product line

would behave by separating and dividing the characteristics according to the characteristics tree and elemental tetrad, allowing the player to select the game's elements, rules, and objectives, thereby creating a unique game for each tree configuration. This game was hard-coded being necessary to create several conditionals for its creation, however for the development of the platform, it is intended that this tree be created automatically based on the source code of the game to be modified.

4.3 Evaluation

This session will present the planning and preliminary qualitative results of the prototypes built to validate the idea of this work. Descriptions about the planning, participants, procedure and results are presented.

4.3.1 Planning

In the search for evidence about the importance, need and approval of the gaming community in the face of the proposal to create the platform, a study was carried out with the objective of validating this proposal.

The main item to be evaluated in this initial evaluation would be the community's approval of the platform's development, so it would be a technology validation, and then some questions from an adapted Technology Acceptance Model (TAM) questionnaire were used (DAVIS, 1993). TAM collects data primarily on the usefulness and usability of the presented idea, allowing users to determine whether the idea to be built will be useful to the community. This model is well-known in the academic field for measuring technology acceptance and has strengths such as focusing on technology-specific information; being extensible, allowing it to be applied in different contexts; and being able to be used during, and after the adoption of a specific technology. However, in this work, some questions of this questionnaire were modified to validate the possible usefulness of the idea before the development of the platform. It is worth remembering that the TAM will be used for final validation of the tool after it has been developed (DOS SANTOS, 2016). However, because the idea is validated through a game, the MEEGA (PETRI *et al.*, 2016) questionnaire was also used. This questionnaire is mainly used to validate the usability and experience provided by a game, however, in this study, it will be used to determine whether or not the usability and experience of the game influenced or not the acceptance of the technology. It is mainly being used to ensure that if the game has a problem, and it does not inhibit the idea from being accepted. As a result, the MEEGA questionnaire will be used to validate the game's usability and experience, while the TAM questions will be used to evaluate the usefulness of the tool to be

built.

The evaluation procedure ran from 10/10/2022 to 10/21/2022, with a pilot evaluation on 10/07/2022 with a participant to make sure that the game and questionnaire had no issues that would affect the evaluation, as well as to confirm the execution time that a candidate would take to carry out the experiment. Following the pilot execution, it was possible to determine that the procedure took an average of 30 minutes and did not present any problem that would affect the experiment.

4.3.2 Participants sample

For the pilot study, only one undergraduate student was used to validate the games and the questionnaire in order to identify possible problems. Three groups of participants were chosen for the main study. This division of groups was created to be able to divide the different levels of experience of the participants, each of these groups will be described in more detail below.

- **Experts:** The first sample was selected from graduate students from Federal University of Rio de Janeiro (UFRJ), UNIRIO and Rio de Janeiro State University (UERJ) who had prior experience with games and Software Reuse. Because this group had more experience in the area, this group attempted to validate the idea more prudently.

- **SR students:** The second population was drawn from a sample of students enrolled in the SR course at UFRJ. This population had less experience with reuse than the first, but they were younger participants who were familiar with games and programming. From this population, it was intended to obtain a less rigorous perspective than the first group and with a vision a little bit focused on people who already used games and programming.

- **Gaming community:** The third group of participants was chosen from the gaming community, leaving the invitation open to anyone who wanted to take part. This group was formed in order to gain a less academic and rigorous perspective and understand what the gaming community is looking for, since they are the biggest mod creators.

4.3.3 Procedure

The study was conducted remotely through the availability of the materials required for the study's execution. However, for the pilot study and the first two populations, the entire experiment was carried out via a Google Meeting call, with

the think-aloud protocol (JÄÄSKELÄINEN, 2010). This protocol was used to collect additional information, such as whether the player was having difficulty with the game and whether he/she liked the proposed idea, among other things. Using this protocol, it was also possible to capture the sound of the environment, which aided in understanding some sensations felt by the player through their reactions, such as claims, sighs, and expressions of fatigue or stress. The main stages of this study are as follows:

- **Study and game description:** This step involved the distribution of a form containing the primary information required to conduct the evaluation, such as explanatory texts about the study, basic game commands, and urls for the questionnaire and installer. It is worth mentioning that the questionnaire was made available in two versions through google forms, English and Portuguese, seeking to have a greater number of participants. The questionnaires used can be found in Appendix A and B.

- **Participants' characterization:** A characterization questionnaire was made available to each of the participants.

- **Game execution:** Participants must install and test the two games sent to them.

- **Completing the qualitative questionnaire:** The TAM (DAVIS, 1993) and MEEGA questionnaires (PETRI *et al.*, 2016) adapted to the context were made available to each of the participants.

4.3.4 Results

This evaluation's primary objective is to determine the viability of the platform development concept. Due to this, the two games described in Sections 4.2.3 and 4.2.4 were evaluated using a single questionnaire, as the purpose of the evaluation was not to evaluate each individual game, but rather to introduce the concept of the platform and determine its viability. This decision was taken in order to reduce or optimize the evaluation time and thus obtain a better result.

Despite the questionnaire used to collect information about usability and game experience, this information was collected only in the event that any game feature is misunderstood or is not to the user's liking; however, this does not affect the final result of the evaluation.

Participants sample

The study involved 46 evaluators, who were divided into three groups: specialists, who were defined as experts users with experience in the field of games and SR; students who took the SR course; and the gaming enthusiast community.

It is worth remembering that, many different answers were given in relation to the participants' time of experience, and in order to present this result in a concise manner, these answers were divided into six categories: More than 10 years, Up to 10 years, Up to 5 years, Up to 3 years, Less than a year and Without experience.

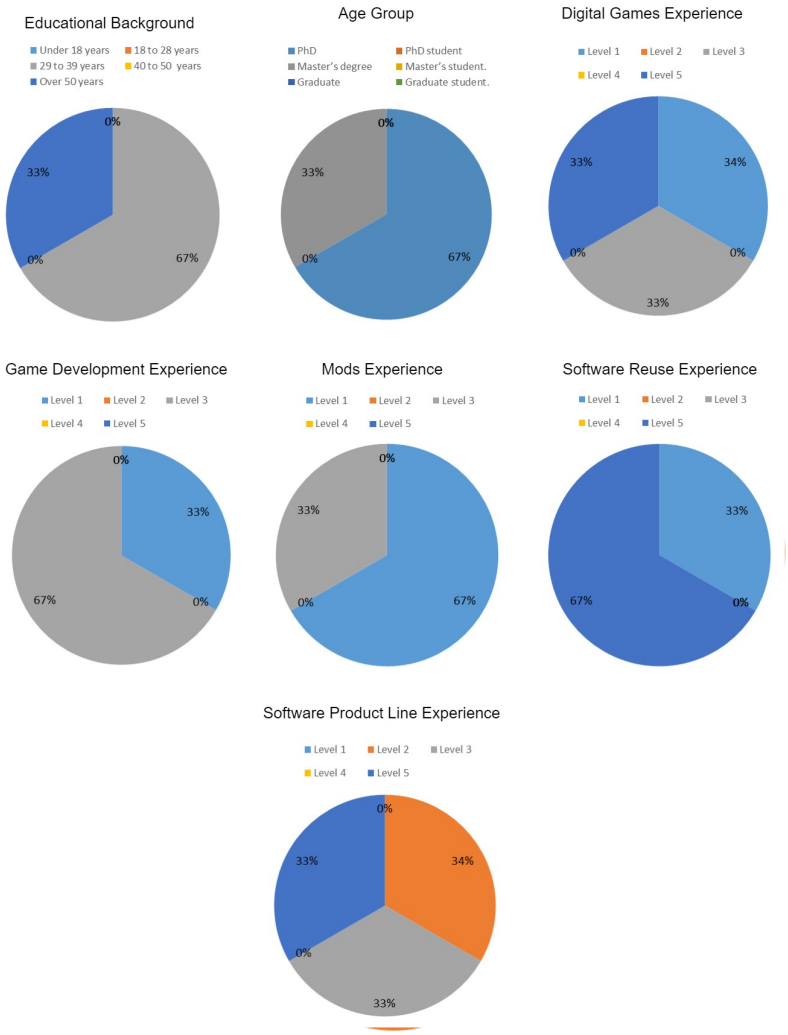

Figure 4.12: Characterization of specialists. Source: from the author.

Analysis of results

Figures 4.12 and 4.13 show how the study specialists were classified, showing that there were three specialists that evaluated the study, each with a distinct level of expertise in each subject. From theses figures, it is possible to infer that two users have been involved with games for more than ten years. However, it is important

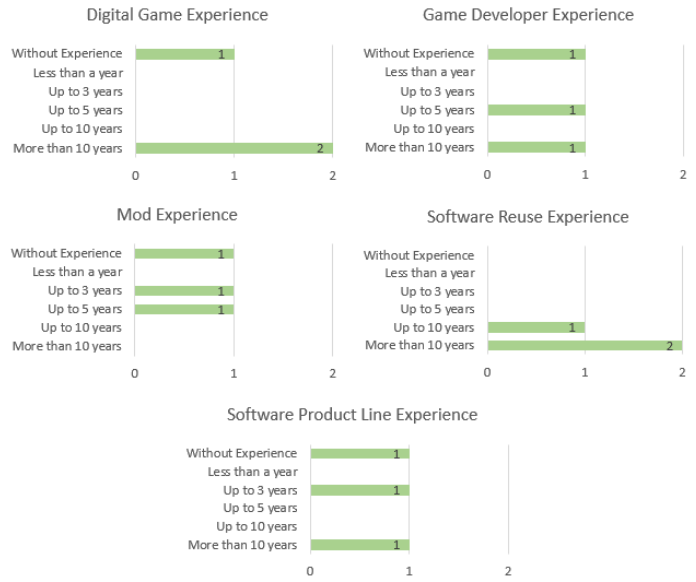

Figure 4.13: Experience of specialists. Source: from the author.

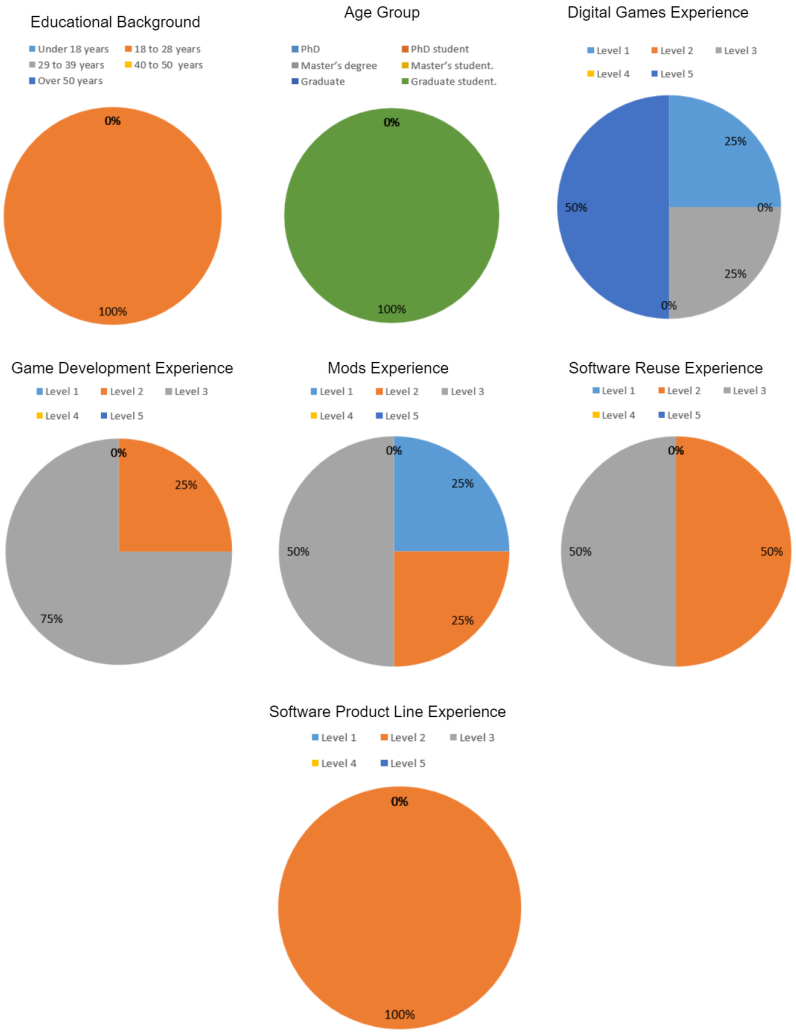

Figure 4.14: Characterization of students. Source: from the author.

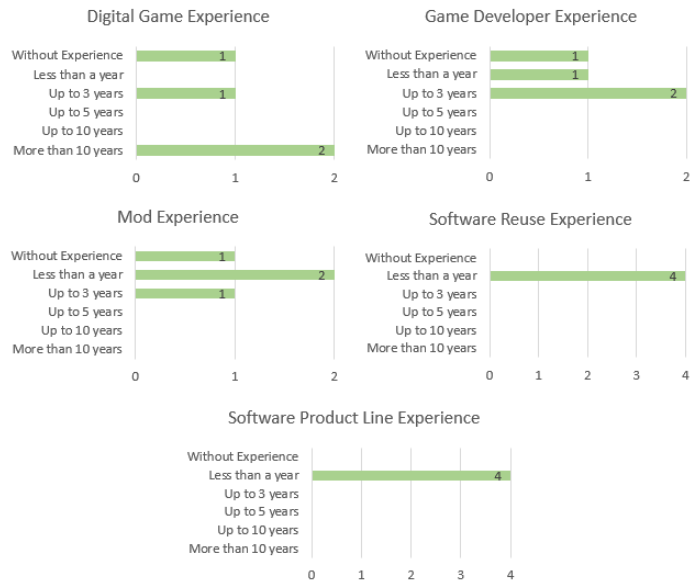

Figure 4.15: Experience of students. Source: from the author.

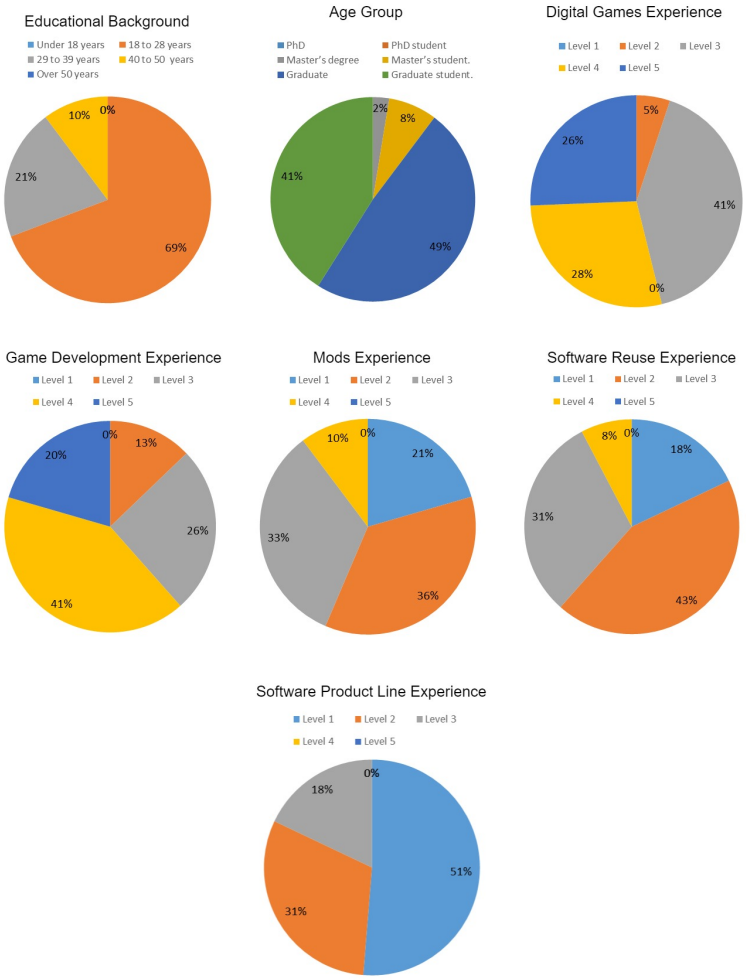

Figure 4.16: Characterization of community. Source: from the author.

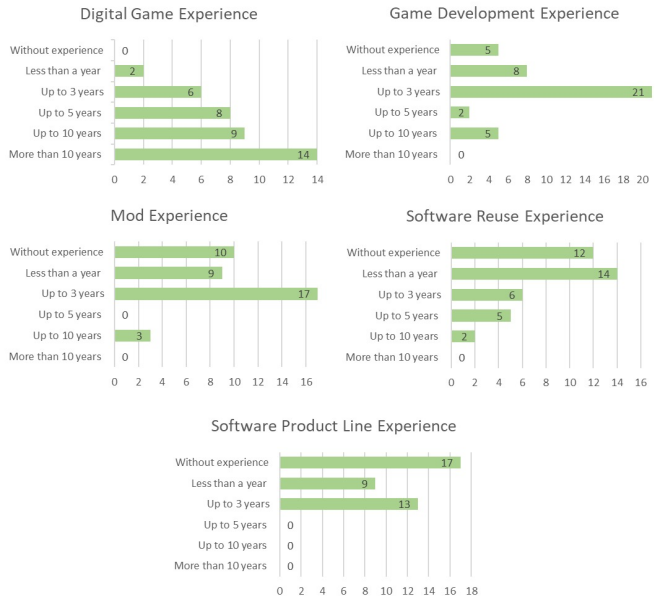

Figure 4.17: Experience of community. Source: from the author.

to note that this time is only in relation to the time they had contact with games, whether they were playing a video game or even doing a little more research on the topic. However, when it comes to game development, it should be noted that only one member has been an expert in the field for more than 10 years, while the other has only 5 years of expertise. It is important to note that this experience time decreases even more when the topic of mods is brought up, with 5 and 3 years of experience for each participant. About SR note that all participants have at least 10 years of experience in the subject, but when dealing with SPL only one participant has this time of experience. When analyzing this information, it is possible to notice that the group consists of a specialist who is somewhat more game-focused, one who is more expert about reuse, and lastly a more generalist user.

When looking at the student research group's data, it is possible to see that the students are all around the same age and are still enrolled in graduation. Information on the students' characteristics is included in Figures 4.12 and 4.13.

Another noteworthy statistic is that while half of the students has more than 10 years of expertise with digital games, when it comes to game development or mods this experience drops to 3 or even zero. Finally, in terms of experience with SR, all students had less than a year of experience, which was to be expected given that they are undergraduate students and are only now becoming familiarized with the field. It is important to note that although everyone was familiar with software development and Software Engineering, they knew little about the SR idea.

When looking at the community assessment group, it is clear that the majority of the participants has between the ages of 18 and 28 and are either graduates or in

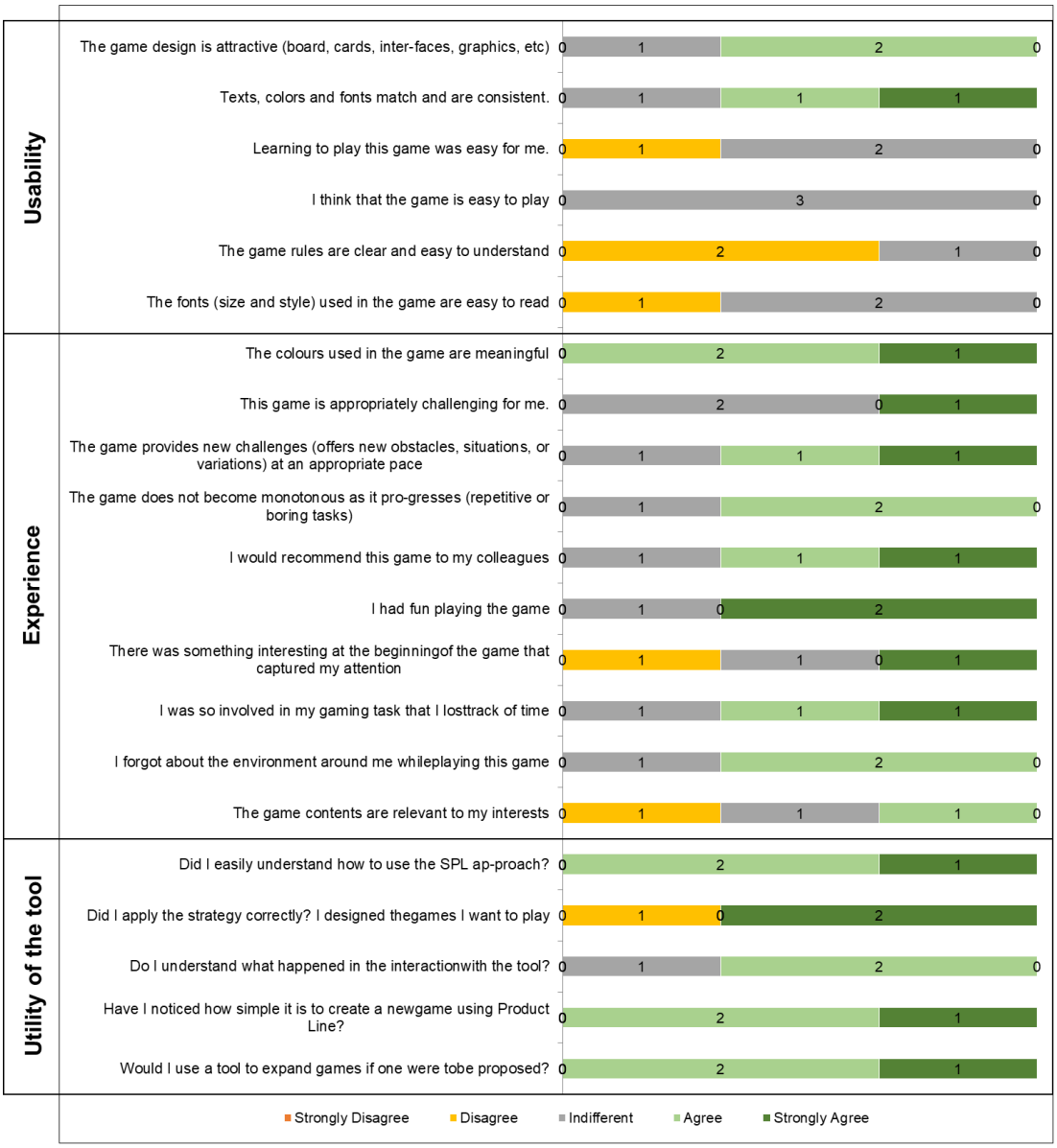

Figure 4.18: Meega / TAM Questionnaire with Specialists Response. Source: from the author.

the process of graduating. When the experiences of these participants are examined, it is clear that the majority of them have been in contact with games for a long time, having been players for more than ten years and having knowledge about mods and game development for about three years. However, when it comes to SR and SPL, it is possible to notice that the majority of evaluators lack ors has limited experience in the field. This was to be expected given that the studies was conducted with the gaming community in mind. Figures 4.16 and 4.17 demonstrate information about the characterization of the community study group.

Despite having three groups of participants with varying levels of knowledge about games and SR, the answers generally followed a pattern. However, it was possible to notice some slightly lower scores in the answers of the specialists, which

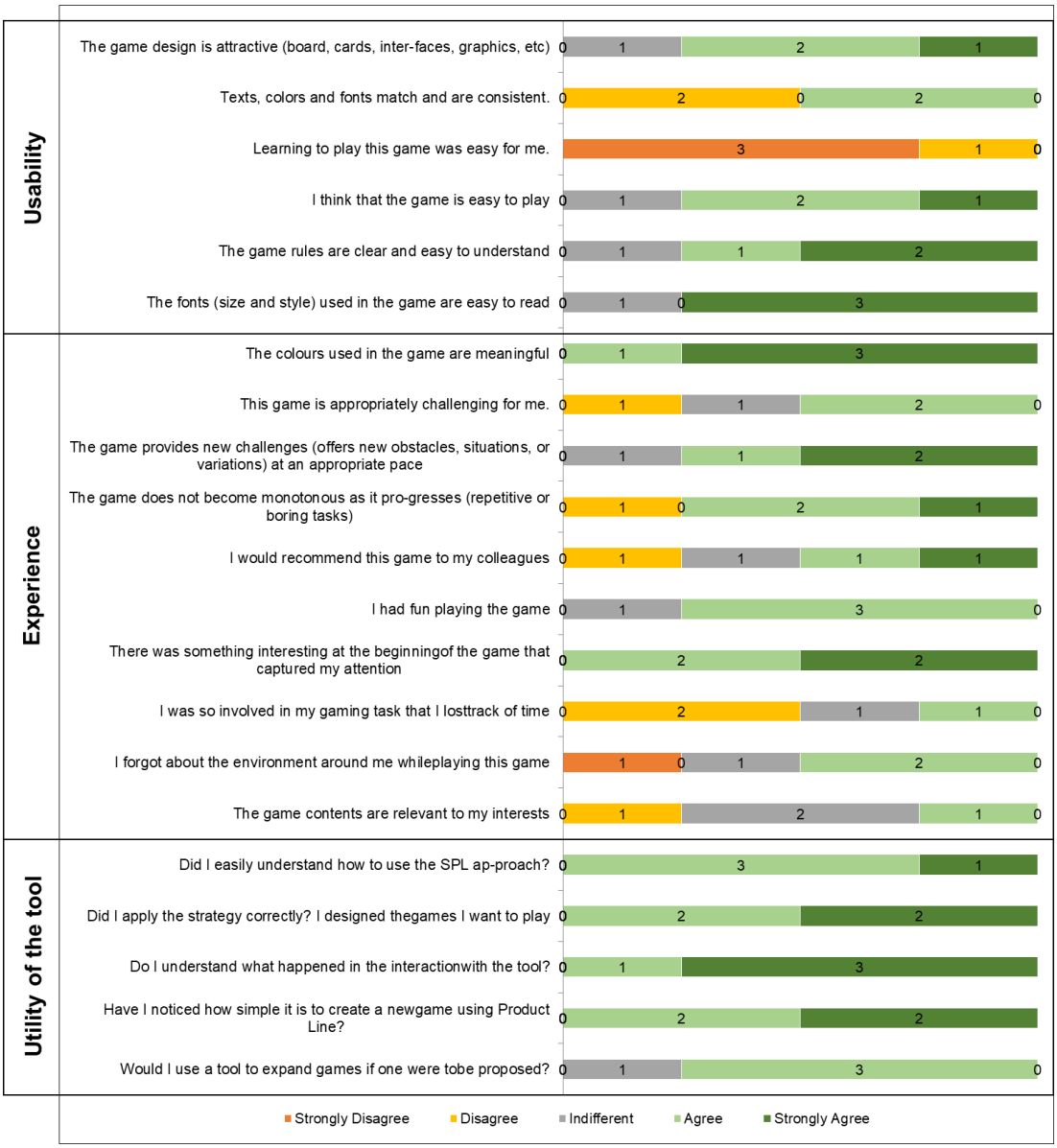

Figure 4.19: Meega / TAM Questionnaire with Students Response. Source: from the author.

was to be expected given that they are specialists and expect a more accurate result. Figures 4.18, 4.19 and 4.20 demonstrate the answers regarding usability, experience and usefulness of the proposed idea.

It should be noted that only one questionnaire was used for both games, taking into account that the main purpose of the evaluation was to check the viability of building the platform. However, a few Meega questionnaire questions were also used to see if the game's usability and experience could influence the evaluation's final result.

Regarding the game's usability and experience, it was possible to notice that it did not directly affect the study's results; however, some improvements that could be made were identified, such as: improving the character's movement, having more

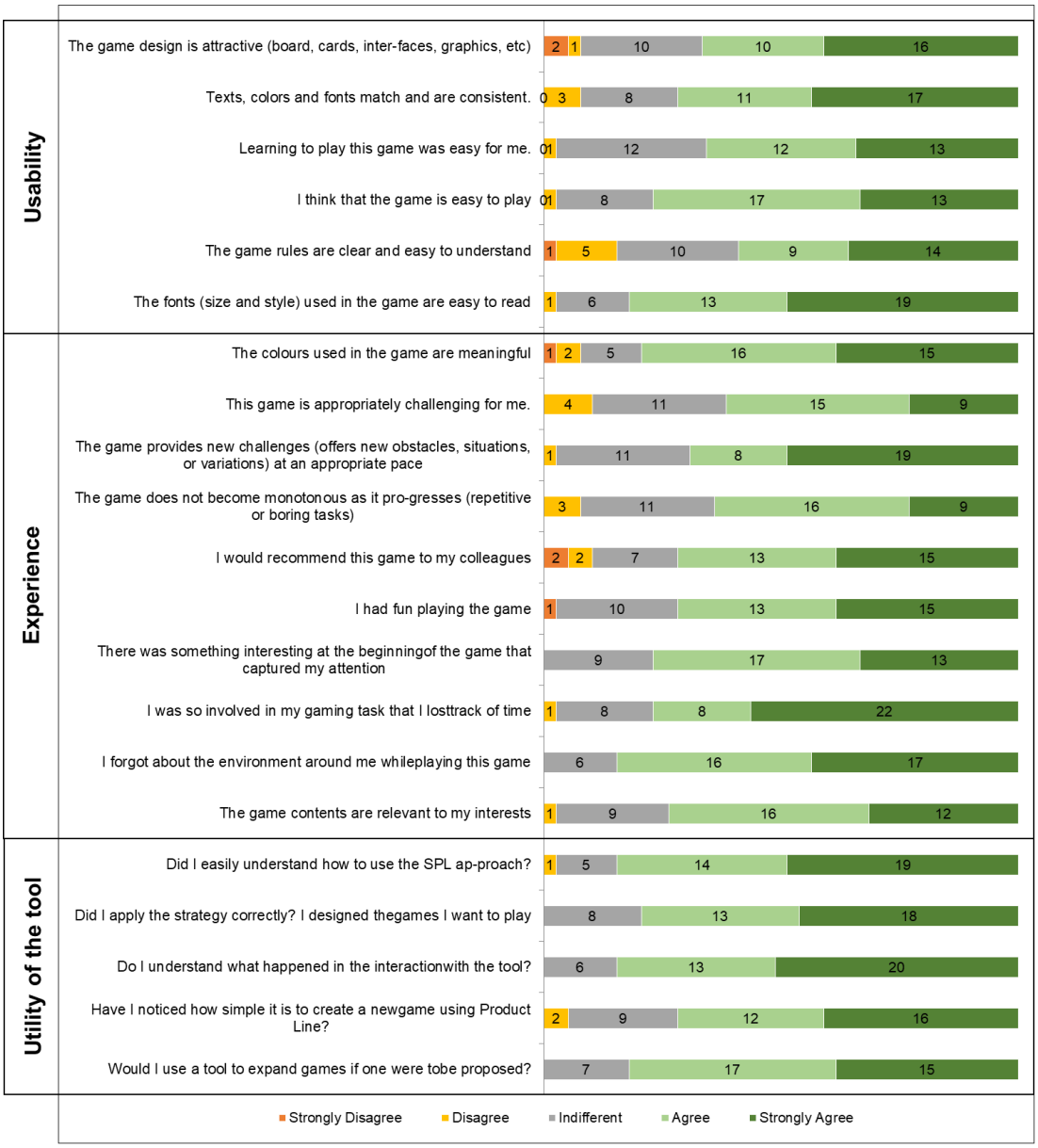

Figure 4.20: Meega / TAM Questionnaire with Community Response. Source: from the author.

instructions on how to play, and using some context variables to improve the player experience by randomly generating the game. Despite the fact that only one questionnaire was used for both games, it is believed that the majority of usability issues are related to the first game, based on comments such as: utilizing context factors to generate the game at random to enhance the user experience; Interesting, beautiful, and intuitive game. However, understanding how games are generated can be difficult for non-gamers.

Regarding the verification of the utility of the game, a considerable number of favorable answers were observed, indicating that users were able to construct games from the notion of SPL and comprehend how the concept of a feature tree would

function. The questionnaire generated responses such as: Regarding the possibility of deriving games from the selection of elements of a line, it appears to be a solution option with real application and contribution to the community; Very interesting concept, I was able to create three completely different games; Interesting concept, which would greatly benefit the game development community.

Looking at the answers in general, it is possible to notice that more than 70% of the answers obtained positive results (Agree and Strongly Agree) and that approximately 95% of the respondents gave answers greater than or equal to Indifferent, with only 5% of the answers obtaining a negative note (Disagree and Strongly Disagree), based on a total of 46 participants. This percentage is even higher when considering only the answers that discuss the tool's usefulness, with more than 81% of positive responses, more than 98% of responses above indifferent, and fewer than 2% of negative responses. From this, it is clear that the suggested tool has a great deal of promise and utility, and that it has the potential to aid the community in the development of mods.

4.4 Conclusion

This chapter presented three games produced with various product line perspectives and creative efforts. The first product line concept was a manual process in which the Lightbot game was modified N times, generating two mods. The second concept was to create mods through a dynamic product line, with the game randomly selecting each of the new game's qualities. Finally, a game was create in which the user selects each of the game's characteristics. Figure 4.21 demonstrates the progression in terms of difficulty and evolution of building the SPL-based game development platform and what each prototype sought to validate at work.

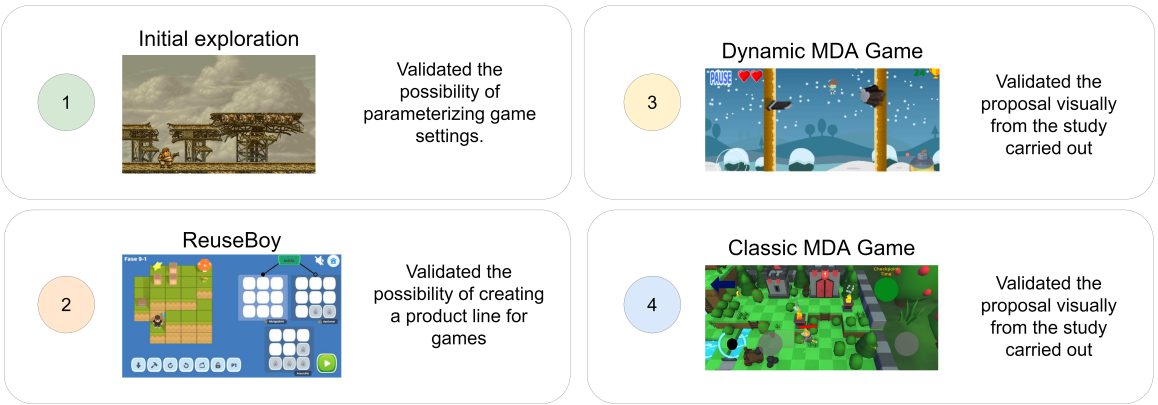

Figure 4.21: Progression in terms of difficulty in building the SPL platform.

As mentioned, four prototypes were created, each of which attempted to validate a portion of the difficulty and possibility of creating the platform. The first sought to validate the difficulty in parameterizing game settings, the second sought to validate the difficulty in manually creating a product line, and the third and fourth sought to validate the platform's concept with the community in an interactive manner.

The last two games created attempted to demonstrate the concept of building the platform to be proposed, and an evaluation was conducted with them to verify the feasibility of building the platform. In general, regardless of the group of evaluators, it was possible to notice a large amount of positive responses for both usability and experience, thus demonstrating that these variables did not influence the final result of the research. Regarding usefulness, it was also possible to perceive positive responses, thus demonstrating that the idea of the platform is valid and would be useful for the community. With this, the development of the platform was followed.

Chapter 5

EngageSPL Platform

As previously stated, the term mods was chosen because it is similar to the term modifiers. It is assumed that the terms advantages, disadvantages, and motivation are comparable.

This chapter aims to describe in more detail the EngageSPL platform that aims to help the development of video games through SPL.

5.1 Overview

Based on the main difficulties of developing mods/modifiers described in Chapter 3, such as lack of appropriate tools, high development cost, difficulty in understanding the source code and delay in development time, the present work proposes the development of a platform that incorporates SPL concepts, the elemental tetrade framework, and modifiers to assist in these problems. Although the platform tries to contribute with all problems, the understanding of source code would not be solved directly, but could be improved by: viewing the feature tree where the code will be separated by features, pre-fabricated modifiers where features can be modified by them and by coding language standards which can make the code easier to understand. It is important to remember that this is not the main purpose of the platform. The main purpose of the platform is to make it possible to build a game once and generate N different versions of it quickly, easily, and with little effort. Table 5.1 demonstrates how each of the problems is intended to be solved through the platform.

Table 5.1: EngageSPL features VS mod development problems.

Problems	Features that aim to help solve the problem

Lack of appropriate tools	Development of a specific platform for the evolution of mods that will make use of the concepts of Product Lines, MDA and mutators
High development cost	As the platform facilitates the evolution of mods, the creation of new mods will become cheaper.
Delay in development time	The process will require less work and less time to develop
Difficulty in understanding the source code	This topic will not be resolved directly. However, certain features will facilitate in its difficulty. (1) The platform will have a development pattern that enables the generation of the feature tree; this pattern will reduce the difficulty of understanding the code, taking into account that all games will have the same structure. (2) Even if the developer does not understand the code, he/she will still be able to view the game's features through the feature tree and modify them through the use of mutators by viewing the feature tree and applying modifiers.

Based on the initial experience with the games described in Chapter 4 and the problems listed above, the thesis proposal is the creation of a platform for the development of video games using the concepts of SPL, elemental tetrade, and modifiers. The platform's core concept is the integration of these three concepts, which makes it possible to divide the game's features into a tree structure and then modify, add, or remove them. The name given to the platform was **EN**gine for **GA**me **GE**neration through **S**oftware **P**roduct **L**ine (EngageSPL).

A platform aims to provide a number of features that will aid in the development of video games, particularly in their evolution. It is intended that the platform will allow to create a game once and then evolve several video games with different characteristics by using modifiers on the edges of the features tree. The platform's concept is to have three feature trees, one for each part of elemental tetrade framework, and then apply the modifiers from these trees. As a result, through the platform, the mechanics, tecnology, story, and aesthetics of the video games can be edited easily and even without programming. Figure 5.1 shows an example of how modifiers can be applied to feature trees, showing three trees, one for each part of the elemental tetrade. From this figure, it is possible to visualize examples of the feature trees, as well as the addition of a new mechanic to the mechanics tree. Through the second level mechanics tree, it is possible to visualize a modifier's edit to the game's second level mechanics, and through aesthetics, it is possible to visualize an edit to

the game's aesthetics.

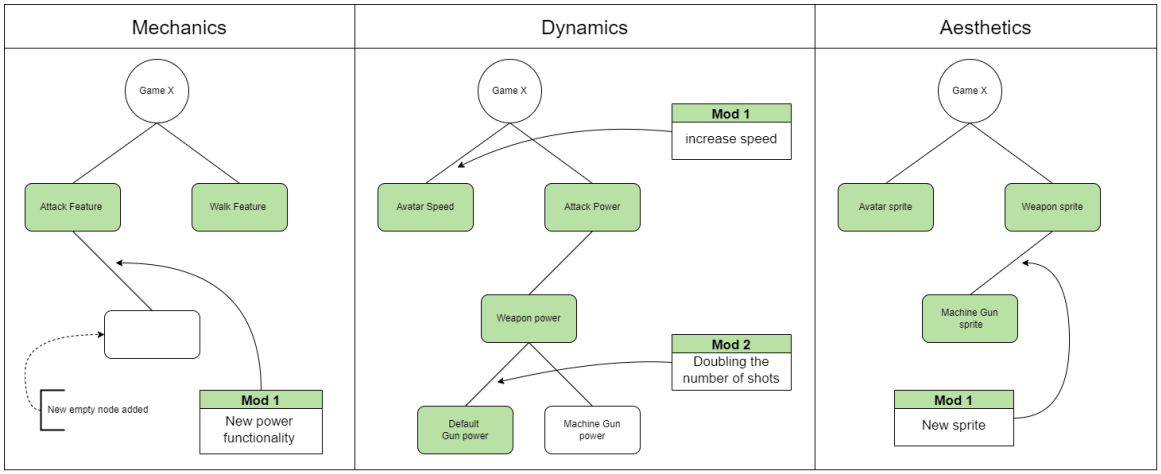

Figure 5.1: Example of feature trees.

To make it possible to carry out what has been presented so far, the platform will have an IDE for building games with several features that allows a video game to evolve. Among the platform's key features are:

- **Pattern check**: The code must follow a specific pattern so that the feature tree and modifiers are created, used, and viewed by the platform. This pattern will be automatically verified by the platform, demonstrating the problems. All the code will be written in javascript, a language that has already been widespread in the community.

- **Feature tree**: Visual demonstration of all game features. Through this tab it will be possible to select all the characteristics of the game to be created, as well as apply the modifiers in a certain characteristic. It is worth remembering that there will be 3 tabs, one for each part of elemental tetrade.

- **modifiers**: The platform will have a list of pre-made modifiers that can be applied to the features of the games to modify them. However, it will also be possible to create extra modifiers. All modifiers can be viewed through the modifiers tab.

Figure 5.2 demonstrates a wireframe of the platform screen, including all of its subscreens. Next, each of them will be described.

- **File list**: List of files for the game being developed.

- **File**: Game file being edited.

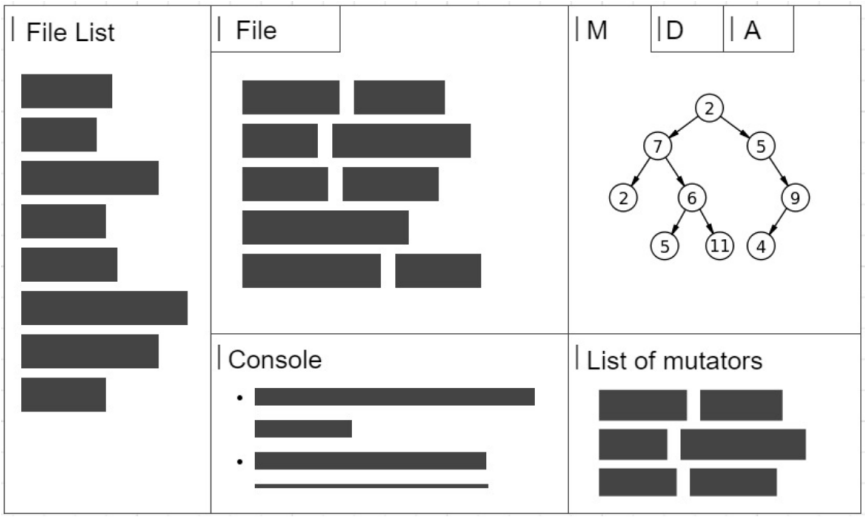

Figure 5.2: Platform idea wireframe

- **elemental tetrade**: Tree of game features that will be generated automatically. Through them the game characteristics can be selected, excluded or modified by the list of modifiers. It is worth remembering that there are 3 trees, one for each part of elemental tetrade.
- **Console**: Terminal that will demonstrate the game's compilation errors if the game doesn't follow the tool's development patterns.
- **List of modifiers**: List of modifiers that can be used to modify the game's feature tree.

From the development of this platform, some results and contributions to the video games area are expected. Next, each one of them will be described.

- **Contributions**
  - **Viewing game features**: Every game created on the platform will have its characteristics separated automatically according to its mechanics, second level mechanics, and aesthetics, facilitating the understanding of the game's characteristics and even serving as documentation.
  - **Fast mod evolution**: The game will be developed only once and N versions of it can be generated, through the choice of characteristics and their modifications through modifiers.
  - **Less time and cost to develop a video games**: the platform simplifies the process of a game's evolution, resulting in a cost reduction.
- **Expected results**

- **A greater number of video games created by the community**
- **Reduced mod development costs**
- **Facilitate the process of building games**
- **Decrease the evolution time of a mod**

In order to achieve the aforementioned results and contributions, the development of the platform must satisfy certain requirements. It is important to note that all platform requirements were derived from the Chapter 3 review. The following table details each platform requirement and the origin from which the review was conducted.

5.2 Schedule of activities

The research and exploratory studies supported the construction of the platform for game evolution through SPL. The main activities that will be carried out for the development of this work will be listed below. Figure 5.4 demonstrates the activity schedule.

	DEC (2022) - SEPT (2023)			OCT 2023	NOV 2023	DEC 2023	JAN 2024	FEB 2024	MAR 2024	APR 2024	MAY 2024	JUNE 2024
Surveys with specialists												
Development activities												
Evaluate the platform (Pilot study)												
Implement improvements												
Evaluate the platform												
Implement improvements												
Write the thesis												
Defend the thesis												

Figure 5.3: Schedule of activities.

Development activities (Detailing)										
	DEC 2022	JAN 2023	FEB 2023	MAR 2023	APR 2023	MAY 2023	JUNE 2023	JULY 2023	AUG 2023	SEPT 2023
Specify game build standards										
Develop the IDE's backbone										
Build pattern checking on the platform										
Generate the IDE compiler										
Develop the feature tree structure										
Develop the structure of mutators										
Integrate the feature tree with mutators										

Figure 5.4: Schedule of development activities.

- **Development activities**
  - **Surveys with specialists:** Literature reviews and previous experiments have demonstrated that the solution can be developed. Once this is confirmed, it is time to consult with specialists in the field regarding compliance and approval for the platform's development. For this, a survey will be conducted with specialists to see their comments and support on the idea to be built. (Questionnaire to be defined).
  - **Specify game build standards:** Specify the design standards that must be followed for building games through the tool. This will help the platform to comprehend how the game was constructed, hence enabling the building of the feature tree.
  - **Develop the IDE's backbones:** Develop the visual structure of the IDE: terminal, file list, text editor, visual tree structure and list of modifiers.
  - **Build pattern checking on the platform:** Develop the code checker that will read the project files and verify that the projects conform to what was specified.
  - **Generate the IDE compiler:** Integrate the compiler with the IDE. This will run the Javascript language compiler and make code run.
  - **Develop the feature tree structure:** Develop the tab that will demonstrate the game's feature tree. This step will check the project files, separate them into characteristics and generate the tree.
  - **Develop the list of modifiers:** Create the tab that will show the list of modifiers. In this step, the basic modifiers and the modifier architectural pattern will be developed, which will allow the creation of new modifiers by the developer.
  - **Integrate the feature tree with modifiers:** Integrating the modifier list with the features tree will enable the modifiers to be utilized to change new games by modifying the tree.
- **Evaluate the platform (Pilot study):** The pilot study will have two types of evaluation
  - The first type of evaluation will use a comparative method to validate the ease and time of development by the platform. Some games already created by other engines will be recreated on the platform to verify their usability.

  - In the second evaluation, a pilot study will be carried out with some students to validate the final questionnaire (to be developed) and to catch possible problems in the tool.

- **Implement improvements:** Develop the possible improvements found in the questionnaire or on the platform.

- **Evaluate the platform:** Evaluate the tool through the modified TAM questionnaire for the platform context (to be defined). This evaluation will include diverse groups of game developers with varying degrees of proficiency: expert, students and community developers. Some participants will utilize the tool to develop games, while others will not. The identical activities will be presented to both groups in order to evaluate the tool's usability, development time, and acceptance.

- **Implement improvements:** Develop the improvements found through the evaluation.

- **Write the thesis:** Conduct an analysis and write of results found through the development of the platform and evaluation.

- **Defend the thesis:** Create the thesis presentation slides, send the thesis to the board. Defend the PhD.

References

ABBOTT, D., 2018, "Modding Tabletop Games for Education", *Proceedings of International Conference on Games and Learning Alliance*, pp. 318–329.

AGARWAL, S., SEETHARAMAN, P., 2015, "Understanding Game Modding through Phases of Mod Development." *Proceedings of ICEIS*, pp. 114–121.

ÅKESSON, J., NILSSON, S., KRÜGER, J., et al., 2019, "Migrating the android apo-games into an annotation-based software product line". In: *Proceedings of the 23rd International Systems and Software Product Line Conference-Volume A*, pp. 103–107.

AL-WASHMI, R., BANA, J., KNIGHT, I., et al., 2014, "Design of a math learning game using a Minecraft mod". In: *European conference on games based learning*, v. 1, p. 10. Academic Conferences International Limited.

ALBASSAM, E., GOMAA, H., 2013, "Applying software product lines to multi-platform video games". In: *2013 3rd International Workshop on Games and Software Engineering: Engineering Computer Games to Enable Positive, Progressive Change (GAS)*, pp. 1–7. IEEE.

AOUADI, N., PERNELLE, P., AMAR, C. B., et al., 2016, "Models and mechanisms for implementing playful scenarios". In: *2016 IEEE/ACS 13th International Conference of Computer Systems and Applications (AICCSA)*, pp. 1–8. IEEE.

ARAÚJO, J. C., ARAÚJO, M. C., NEIVA, F. W., 2022, "Uma abordagem de aprendizado integrada e interdisciplinar através de um processo de desenvolvimento cross-platform", *ANALECTA-Centro Universitário Academia*, v. 7, n. 2.

AYALA, I., PAPADOPOULOS, A. V., AMOR, M., et al., 2021, "Prodspl: Proactive self-adaptation based on dynamic software product lines", *Journal of Systems and Software*, v. 175, pp. 110909.

BALDASSARRE, M. T., CAIVANO, D., ROMANO, S., et al., 2021, "PhyDSLK: a model-driven framework for generating exergames", *Multimedia Tools and Applications*, v. 80, n. 18, pp. 27947–27971.

BEATTIE, A., 2020. "How the Video Game Industry Is Changing". `https://www.investopedia.com/articles/investing/053115/how-video-game-industry-changing.asp`. Online; accessed 29 May 2022.

BILIŃSKA, K., DEWALSKA-OPITEK, A., HOFMAN-KOHLMEYER, M., 2020, "To Mod or Not to Mod—An Empirical Study on Game Modding as Customer Value Co-Creation", *Sustainability*, v. 12, n. 21, pp. 9014.

BLOIS, A., 2006, "Uma abordagem de Projeto Arquitetural baseado em Componentes no Contexto de Engenharia de Domínio", *Unpublished doctoral dissertation). Universidade Federal Do Rio De Janeiro, Rio de Janeiro, Brazil*.

BOAVENTURA, F., SARINHO, V. T., 2017, "Mendiga: A minimal engine for digital games", *International Journal of Computer Games Technology*, v. 2017.

BOURQUE, P., FAIRLEY, R. E., 2014, *Guide to the Software Engineering Body of Knowledge SWEBOK*. 3rd ed. , IEEE Computer Society. ISBN: 978-0-7695-5166-1.

CASTRO, D., WERNER, C., 2021, "Rebuilding games at runtime". In: *2021 36th IEEE/ACM International Conference on Automated Software Engineering Workshops (ASEW)*, pp. 73–77. IEEE.

CASTRO, D. C. B., 2020, *USO DE JOGOS COMO ESTRATÉGIA PARA O ENSINO DE REUTILIZAÇÃO DE SOFTWARE*. Tese de Mestrado, Universidade Federal do Rio de Janeiro.

CHAMPION, E., 2013, *Game mods: design, theory and criticism*. `Lulu.com`.

CHEUNG, G., HUANG, J., 2012, "Remix and play: lessons from rule variants in texas hold'em and halo 2". In: *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work*, pp. 559–568.

CIGNONI, G. A., 2001, "Reporting about the Mod software process". In: *European Workshop on Software Process Technology*, pp. 242–245. Springer.

CLAUDE CHAUNIER, TORBEN MOGENSEN, B. T. "Mutators". `http://www.di.fc.ul.pt/~jpn/cv/mutators.htm`. Online; accessed 04 March 2021.

CLELAND, N. "Mod (video games)". https://en.wikipedia.org/wiki/Mod_(video_games). Online; accessed 04 March 2021.

CUSKER, J., 2013, "Elsevier Compendex and Google Scholar: a quantitative comparison of two resources for engineering research and an update to prior comparisons", *The Journal of Academic Librarianship*, v. 39, n. 3, pp. 241–243.

DACONCEICAO, R., LOCKE, C., COOPER, K., et al., 2013, "Semi-automated serious educational game generation: A component-based game engineering approach", *Proceedings of CGAMES'2013 USA*, pp. 222–227.

DAMAŠEVIČIUS, R., AŠERIŠKIS, D., 2017, "Visual and computational modelling of minority games", *TEM J*, v. 6, n. 1, pp. 108–116.

DAVIS, F. D., 1993, "User acceptance of information technology: system characteristics, user perceptions and behavioral impacts", *International journal of man-machine studies*, v. 38, n. 3, pp. 475–487.

DEBBICHE, J., LIGNELL, O., KRÜGER, J., et al., 2019, "Migrating Java-based apo-games into a composition-based software product line". In: *Proceedings of the 23rd International Systems and Software Product Line Conference-Volume A*, pp. 98–102.

DERAKHSHANDI, M., KOLAHDOUZ-RAHIMI, S., TROYA, J., et al., 2021, "A model-driven framework for developing android-based classic multiplayer 2D board games", *Automated Software Engineering*, v. 28, n. 2, pp. 1–57.

DEY, T., MASSENGILL, J. L., MOCKUS, A., 2016, "Analysis of popularity of game mods: A case study". In: *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play Companion Extended Abstracts*, pp. 133–139.

DJAOUTI, D., ALVAREZ, J., JESSEL, J.-P., 2011, "Classifying serious games: the G/P/S model". In: *Handbook of research on improving learning and motivation through educational games: Multidisciplinary approaches*, IGI global, pp. 118–136.

DORMANS, J., 2011, "Simulating mechanics to study emergence in games". In: *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, v. 7, pp. 2–7.

DOS SANTOS, R. P., 2016, *Managing and monitoring software ecosystem to support demand and solution analysis.* Tese de Doutorado, Ph. D. Dissertation. Universidade Federal do Rio de Janeiro.

EHRMANN, J., LEWIS, C., LEWIS, P., 1968, "Homo ludens revisited", *Yale French Studies*, , n. 41, pp. 31–57.

FENSKE, W., MEINICKE, J., SCHULZE, S., et al., 2017, "Variant-preserving refactorings for migrating cloned products to a product line". In: *2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, pp. 316–326. IEEE.

FURTADO, A. W., SANTOS, A. L., RAMALHO, G. L., 2011, "SharpLudus revisited: from ad hoc and monolithic digital game DSLs to effectively customized DSM approaches". In: *Proceedings of the compilation of the co-located workshops on DSM'11, TMC'11, AGERE! 2011, AOOPES'11, NEAT'11, & VMIL'11*, pp. 57–62.

GAROUSI, V., FELDERER, M., MÄNTYLÄ, M. V., 2019, "Guidelines for including grey literature and conducting multivocal literature reviews in software engineering", *Information and Software Technology*, v. 106, pp. 101–121.

GEORGE, S., LAVOUÉ, É., MONTERRAT, B., 2013, "An environment to support collaborative learning by modding". In: *European Conference on Technology Enhanced Learning*, pp. 111–124. Springer.

GEORGIEV, A., GRIGOROV, A., BONTCHEV, B., et al., 2016, "The RAGE advanced game technologies repository for supporting applied game development". In: *International Conference on Games and Learning Alliance*, pp. 235–245. Springer.

GOUWS, L. A., BRADSHAW, K., WENTWORTH, P., 2013, "Computational thinking in educational activities: an evaluation of the educational game light-bot". In: *Proceedings of the 18th ACM conference on Innovation and technology in computer science education*, pp. 10–15.

GUANA, V., STROULIA, E., NGUYEN, V., 2015, "Building a game engine: A tale of modern model-driven engineering". In: *2015 IEEE/ACM 4th International Workshop on Games and Software Engineering*, pp. 15–21. IEEE.

GUARDIOLA, E., 2016, "The gameplay loop: a player activity model for game design and analysis". In: *Proceedings of the 13th International Conference on Advances in Computer Entertainment Technology*, pp. 1–7.

GUO, H., TRÆTTEBERG, H., WANG, A. I., et al., 2015a, "A workflow for model driven game development". In: *2015 IEEE 19th International Enterprise Distributed Object Computing Conference*, pp. 94–103. IEEE, a.

GUO, H., TRÆTTEBERG, H., WANG, A. I., et al., 2015b, "Lessons from Practicing an Adapted Model Driven Approach in Game Development". In: *International Conference on Entertainment Computing*, pp. 451–456. Springer, b.

HOFMAN-KOHLMEYER, M. M., 2019, "PLAYERS AS CONTENT CREATORS. THE BENEFITS OF GAME MODDING ACCORDING TO POLISH USERS", *International Scientific Journal News*, v. 2, pp. 8–26.

HUNICKE, R., LEBLANC, M., ZUBEK, R., 2004, "MDA: A formal approach to game design and game research". In: *Proceedings of the AAAI Workshop on Challenges in Game AI*, v. 4.

JÄÄSKELÄINEN, R., 2010, "Think-aloud protocol", *Handbook of translation studies*, v. 1, pp. 371–374.

JARRETT, J., 2021, "Gaming the gift: The affective economy of League of Legends 'fair'free-to-play model", *Journal of Consumer Culture*, v. 21, n. 1, pp. 102–119.

KANG, K. C., COHEN, S. G., HESS, J. A., et al., 1990, *Feature-oriented domain analysis (FODA) feasibility study*. Relatório técnico, Carnegie-Mellon Univ Pittsburgh Pa Software Engineering Inst.

KHORRAM, F., TAROMIRAD, M., RAMSIN, R., 2021, "SeGa4Biz: Model-Driven Framework for Developing Serious Games for Business Processes." In: *MODELSWARD*, pp. 139–146.

KITCHENHAM, B., BRERETON, O. P., BUDGEN, D., et al., 2009, "Systematic literature reviews in software engineering–a systematic literature review", *Information and software technology*, v. 51, n. 1, pp. 7–15.

KOLBERT, J. "Unofficial patch". `https://en.wikipedia.org/wiki/Unofficial_patch`. Online; accessed 09 March 2021.

KRUEGER, C. W., 1992, "Software reuse", *ACM Computing Surveys (CSUR)*, v. 24, n. 2, pp. 131–183.

KRÜGER, J., FENSKE, W., THÜM, T., et al., 2018, "Apo-games: a case study for reverse engineering variability from cloned Java variants". In: *Proceedings*

of the 22nd International Systems and Software Product Line Conference-Volume 1, pp. 251–256.

KYAW, A. S., 2013, *Unity 4. x Game AI programming.* Packt Publishing Ltd.

LACERDA, D. P., DRESCH, A., PROENÇA, A., et al., 2013, "Design Science Research: método de pesquisa para a engenharia de produção", *Gestão & produção*, v. 20, n. 4, pp. 741–761.

LEE, D., LIN, D., BEZEMER, C.-P., et al., 2020, "Building the perfect game–an empirical study of game modifications", *Empirical Software Engineering*, pp. 1–34.

LIMA, C., ASSUNÇÃO, W. K., MARTINEZ, J., et al., 2019, "Product line architecture recovery with outlier filtering in software families: the Apo-Games case study", *Journal of the Brazilian Computer Society*, v. 25, n. 1, pp. 1–17.

MAGIOLADITIS, M. "Video game conversion". `https://en.wikipedia.org/wiki/Video_game_conversion`. Online; accessed 07 March 2021.

MAIA, N., BACELO, A. P. T., WERNER, C. M. L., 2007, "Odyssey-MDA: A Transformational Approach to Component Models." *Proceedings of Internacional Conference on Software Engineering and Knowledge Engineering*, pp. 9–14.

MARCHAND, A., HENNIG-THURAU, T., 2013, "Value creation in the video game industry: Industry economics, consumer benefits, and research opportunities", *Journal of interactive marketing*, v. 27, n. 3, pp. 141–157.

MARQUES, E., BALEGAS, V., BARROCA, B. F., et al., 2012, "The RPG DSL: a case study of language engineering using MDD for generating RPG games for mobile phones". In: *Proceedings of the 2012 workshop on Domain-specific modeling*, pp. 13–18.

MATALONGA, S., RODRIGUES, F., TRAVASSOS, G. H., 2017, "Characterizing testing methods for context-aware software systems: Results from a quasi-systematic literature review", *Journal of Systems and Software*, v. 131, pp. 1–21.

MCARTHUR, V., TEATHER, R. J., 2015, "Serious mods: A case for modding in serious games pedagogy", *Proceedings of IEEE Games Entertainment Media Conference (GEM)*, pp. 1–4.

MENDONÇA, W. D., ASSUNÇÃO, W. K., LINSBAUER, L., 2018, "Multi-objective optimization for reverse engineering of apo-games feature models". In: *Proceedings of the 22nd International Systems and Software Product Line Conference-Volume 1*, pp. 279–283.

MORAES, T. M., SOUZA, A. D., 2011, "Revisão sistemática sobre a comunicação dentro do processo de desenvolvimento de software", *Universidade Federal de Goiás-GO*, p. 57.

MORALES, L., MÉNDEZ-ACUNA, D., MONTES, W., 2011, "Model-driven game development-case study. a mtc for maze-game s prototyping", *Revista electrónica en construcción de software PARADIGMA*, v. 5, n. 3, pp. 1–15.

MORISIO, M., EZRAN, M., TULLY, C., 2002, "Success and failure factors in software reuse", *IEEE Transactions on software engineering*, v. 28, n. 4, pp. 340–357.

MOTTA, R. C., DE OLIVEIRA, K. M., TRAVASSOS, G. H., 2016, "Characterizing Interoperability in Context-aware Software Systems", *Proceedings of VI Brazilian Symposium on Computing Systems Engineering (SBESC)*, pp. 203–208.

NETO, J. P., TAYLOR, W., "Game Mutators for Restricting Play", *Game & Puzzle Design, vol. 1, no. 2, 2015 (Colour)*, p. 64.

NIEBORG, D. B., 2005a, "Am I mod or not?-An analysis of first person shooter modification culture". In: *Creative Gamers Seminar—Exploring Participatory Culture in Gaming, University of Tampere, Finland (14–15 January)*, a.

NIEBORG, D. B., 2005b, "Am I mod or not?—An analysis of first person shooter modification culture". In: *Creative Gamers Seminar—Exploring Participatory Culture in Gaming, University of Tampere, Finland (14–15 January)*, b.

NÚÑEZ-VALDEZ, E. R., GARCÍA-DÍAZ, V., LOVELLE, J. M. C., et al., 2017, "A model-driven approach to generate and deploy videogames on multiple platforms", *Journal of Ambient Intelligence and Humanized Computing*, v. 8, n. 3, pp. 435–447.

PASHKOV, S., 2021, "Video game industry market analysis: Approaches that resulted in industry success and high demand", .

PATTERSON, R. F. "Mod (video gaming)". https://civilization.fandom.com/wiki/Mod_(video_gaming). Online; accessed 05 March 2021.

PETRI, G., VON WANGENHEIM, C. G., BORGATTO, A. F., 2016, "MEEGA+: an evolution of a model for the evaluation of educational games", *INCoD/GQS*, v. 3, pp. 1–40.

PETTICREW, M., ROBERTS, H., 2008, *Systematic reviews in the social sciences: A practical guide*. John Wiley & Sons.

POLITOWSKI, C., PETRILLO, F., GUÉHÉNEUC, Y.-G., 2021, "A Survey of Video Game Testing". In: *2021 IEEE/ACM International Conference on Automation of Software Test (AST)*, pp. 90–99. IEEE.

POOR, N., 2014, "Computer game modders' motivations and sense of community: A mixed-methods approach", *New media & society*, v. 16, n. 8, pp. 1249–1267.

PORETSKI, L., ARAZY, O., 2017, "Placing value on community co-creations: A study of a video game'modding'community". In: *Proceedings of the 2017 ACM Conference on Computer Supported Cooperative Work and Social Computing*, pp. 480–491.

POSTIGO, H., 2007, "Of mods and modders: Chasing down the value of fan-based digital game modifications", *Games and Culture*, v. 2, n. 4, pp. 300–313.

QURESHI, M. R. J., 2012, "Agile software development methodology for medium and large projects", *IET software*, v. 6, n. 4, pp. 358–363.

RAMADAN, R. "Does game modding require programming?" https://www.quora.com/Does-game-modding-require-programming. Online; accessed 09 March 2021.

RAMADAN, R., WIDYANI, Y., 2013, "Game development life cycle guidelines", *2013 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pp. 95–100.

RANDELL, B., 1979, "Software Engineering in 1968". In: *Proceedings of the 4th International Conference on Software Engineering*, ICSE'79, p. 1–10. IEEE Press.

RINCÓN, L., MARTÍNEZ, J.-C., PABÓN, M. C., et al., 2018, "Creating a software product line of mini-games to support language therapy". In: *Colombian Conference on Computing*, pp. 418–431. Springer.

SAMETINGER, J., 1997, *Software engineering with reusable components*. Springer Science & Business Media.

SÁNCHEZ, K., GARCÉS, K., CASALLAS, R., 2015, "A dsl for rapid prototyping of cross-platform tower defense games". In: *2015 10th Computing Colombian Conference (10CCC)*, pp. 93–99. IEEE.

SARINHO, V. T., APOLINÁRIO, A. L., ALMEIDA, E. S., 2012, "A feature-based environment for digital games". In: *International Conference on Entertainment Computing*, pp. 518–523. Springer.

SARINHO, V. T., DE AZEVEDO, G. S., BOAVENTURA, F. M., 2018, "Askme: A feature-based approach to develop multiplatform quiz games". In: *2018 17th Brazilian Symposium on Computer Games and Digital Entertainment (SBGames)*, pp. 38–3809. IEEE.

SCACCHI, W., 2011a, "Modding as an open source approach to extending computer game systems", *Proceedings of IFIP International Conference on Open Source Systems*, pp. 62–74.

SCACCHI, W., 2011b, "Modding as a basis for developing game systems", *Proceedings of the 1st international workshop on Games and software engineering*, pp. 5–8.

SCHELL, J., 2008, *The Art of Game Design: A book of lenses*. CRC press.

SHIRATUDDIN, M. F., THABET, W., 2011, "Utilizing a 3D game engine to develop a virtual design review system", .

SIERRA, M., PABÓN, M. C., RINCÓN, L., et al., 2019, "A comparative analysis of game engines to develop core assets for a software product line of mini-games". In: *International Conference on Software and Systems Reuse*, pp. 64–74. Springer.

SOTAMAA, O., 2007, "On modder labour, commodification of play, and mod competitions", *First Monday*, v. 12, n. 9.

SOTAMAA, O., 2010, "When the game is not enough: Motivations and practices among computer game modding culture", *Games and Culture*, v. 5, n. 3, pp. 239–255.

TANG, S., HANNEGHAN, M., 2013, "A model driven serious games development approach for game-based learning". In: *Proceedings of the International Conference on Software Engineering Research and Practice (SERP)*, p. 1.

The Steering Committee of The World Congress in Computer Science, Computer ....

TANG, S., HANNEGHAN, M., CARTER, C., 2013, "A platform independent game technology model for model driven serious games development", *Electronic Journal of e-Learning*, v. 11, n. 1, pp. pp61–79.

TENGTRIRAT, T., PROMPOON, N., 2013, "Applying Exception Handling Patterns for User Interface Customization in Software Games Modification". In: *Proceedings of the International MultiConference of Engineers and Computer Scientists*, v. 1.

THILLAINATHAN, N., 2013, "A model driven development framework for serious games", *Available at SSRN 2475410*.

UNGER, A., 2012, "Modding as part of game culture", *Computer Games and New Media Cultures*, pp. 509–523.

VALDEZ, E. R. N., MARTÍNEZ, Ó. S., BUSTELO, B. C. P. G., et al., 2013, "Gade4all: developing multi-platform videogames based on domain specific languages and model driven engineering", *IJIMAI*, v. 2, n. 2, pp. 33–42.

VEGT, W. V. D., NYAMSUREN, E., WESTERA, W., 2016, "RAGE reusable game software components and their integration into serious game engines". In: *International Conference on Software Reuse*, pp. 165–180. Springer.

VEGT, W. V. D., NYAMSUREN, E., WESTERA, W., 2018, "Making Serious Games with Reusable Software Components". In: *Joint International Conference on Serious Games*, pp. 13–16. Springer.

WADA, H., SUZUKI, J., 2005, "Modeling turnpike frontend system: A model-driven development framework leveraging UML metamodeling and attribute-oriented programming". In: *International Conference on Model Driven Engineering Languages and Systems*, pp. 584–600. Springer.

WALT, S., 2010, "Computer game mods, modders, modding, and the mod scene", *First Monda*, v. 15, n. 5.

WEEKE, C., 2020, "Appropriation & Motivation in Game Modification", *Erasmus University Thesis Repository*.

WIDJMAN, T., 2021. "Global Games Market to Generate $175.8 Billion in 2021; Despite a Slight Decline, the Market Is on Track to Surpass $200 Billion in 2023". https://newzoo.com/insights/articles/global-games-market-to-generate-175-8-billion-in-2021-despite-a-slight-decline-the-market-is-on-track-to-surpass-200-billion-in-2023. Online; accessed 29 May 2022.

WOHLIN, C., 2014, "Guidelines for snowballing in systematic literature studies and a replication in software engineering". In: *Proceedings of the 18th international conference on evaluation and assessment in software engineering*, pp. 1–10.

XEXÉO, G., CARMO, A., ACIOLI, A., et al., 2013, "O que são jogos", *LUDES. Rio de Janeiro*, v. 1, pp. 1–30.

ZHU, M., WANG, A. I., 2019, "Model-driven game development: A literature review", *ACM Computing Surveys (CSUR)*, v. 52, n. 6, pp. 1–32.

ZHU, M., WANG, A. I., TRÆTTEBERG, H., 2016, "Engine-cooperative game modeling (ecgm) bridge model-driven game development and game engine tool-chains". In: *Proceedings of the 13th International Conference on Advances in Computer Entertainment Technology*, pp. 1–10.

ZUPPIROLI, S., CIANCARINI, P., GABBRIELLI, M., 2012, "A role-playing game for a software engineering lab: Developing a product line". In: *2012 IEEE 25th Conference on Software Engineering Education and Training*, pp. 13–22. IEEE.

Appendix A

TAM + MEEGA Questionnaire (English version)

A.1 Job description

Games is one of the industries that has grown significantly over the years, attracting enthusiasts of all ages, genres, and tastes and reaching a community of billions of consumers. However, game development can be time-consuming, with numerous participants and stages, which makes some titles to take years to complete. With such a large community, some customers cannot wait that long for the game to be released. As a result, they end up making his/her own versions of the game; this process of modifying an existing game to make a new one is known as a mod.

Although the development of mods is common in the gaming community, a study revealed some difficulties in the process, which stand out: the lack of specialized tools for building mods, the difficulty of understanding the original game's source code, and, at times, the need to recreate the original game from scratch.

The mod concept is very similar to the concept of opportunistic software reuse, in which specific software is copied and modified. Through a study on games and software reuse it was possible to conclude that Software Product Line would be one of the most recommended approaches for building mods. As a result, the goal of this work is to demonstrate the concept of a product line for building games using two existing games. The first aims to generate a new game automatically modifying the game's mechanics, dynamics, and aesthetics. The second lets the player design his/her own game by combining mechanics, dynamics, and aesthetics.

Basic commands and notes:

- Game 1
  - Use the joystick to move the character

  - Avoid hitting obstacles

- Game 2
  - Within the feature tree, select all desired mechanics, dynamics, and aesthetics for the game. If you have two options for a trait, choose at least one.
  - Use the joystick to move the character
  - To attack or hold the boxes, press the buttons on the right.
  - To jump, press the left side button.
  - Complete the game objectives outlined on the splash screen to win.

A.2 Characterization questionnaire

Please answer the following questions based on your experience with the games. All data collected will be used exclusively for research purposes and will be published completely anonymously, without compromising the participant.

1. **Educational background:**
   - ☐ PhD.
   - ☐ PhD student.
   - ☐ Master's degree.
   - ☐ Master's student.
   - ☐ Graduate.
   - ☐ Graduate student.
   - ☐ Others (please specify): _____

2. **Age group:**
   - ☐ Less than 18 years
   - ☐ 18-28 years
   - ☐ 29-39 years
   - ☐ 40-50 years
   - ☐ Over 50 years

3. **Experience**

Consider: (1) no experience with the activity; (2) theoretical knowledge but no practice; (3) personal or classroom projects; and (4) industry projects. (5) Has extensive knowledge

	1	2	3	4	5
Digital games	☐	☐	☐	☐	☐
Game development	☐	☐	☐	☐	☐
Mods	☐	☐	☐	☐	☐
Software Reuse	☐	☐	☐	☐	☐
Software product line	☐	☐	☐	☐	☐

4. Time experience

Please provide details in your answer. Include how many months of experience you have in each of the knowledge areas.

Technology	Months
Digital games	
Game development	
Mods	
Software Reuse	
Software product line	

A.3 Evaluation questionnaire

5. Usability

	1	2	3	4	5
The game design is attractive (board, cards, interfaces, graphics, etc).	☐	☐	☐	☐	☐
Texts, colors and fonts match and are consistent.	☐	☐	☐	☐	☐
Learning to play this game was easy for me.	☐	☐	☐	☐	☐
I think that the game is easy to play.	☐	☐	☐	☐	☐
The game rules are clear and easy to understand.	☐	☐	☐	☐	☐
The fonts (size and style) used in the game are easy to read.	☐	☐	☐	☐	☐
The colours used in the game are meaningful.	☐	☐	☐	☐	☐

6. Experience

	1	2	3	4	5
This game is appropriately challenging for me.	☐	☐	☐	☐	☐
The game provides new challenges (offers new obstacles, situations, or variations) at an appropriate pace.	☐	☐	☐	☐	☐
The game does not become monotonous as it progresses (repetitive or boring tasks).	☐	☐	☐	☐	☐
I would recommend this game to my colleagues.	☐	☐	☐	☐	☐
I had fun playing the game.	☐	☐	☐	☐	☐
There was something interesting at the beginning of the game that captured my attention.	☐	☐	☐	☐	☐
I was so involved in my gaming task that I lost track of time.	☐	☐	☐	☐	☐
I forgot about the environment around me while playing this game.	☐	☐	☐	☐	☐
The game contents are relevant to my interests.	☐	☐	☐	☐	☐

7. Utility of the proposed tool:

	1	2	3	4	5
Did I easily understand how to use the SPL approach?	☐	☐	☐	☐	☐
Did I apply the strategy correctly? I designed the games I want to play.	☐	☐	☐	☐	☐
Do I understand what happened in the interaction with the tool?	☐	☐	☐	☐	☐
Have I noticed how simple it is to create a new game using Product Line?	☐	☐	☐	☐	☐
Would I use a tool to expand games if one were to be proposed?	☐	☐	☐	☐	☐

8. Have you identified any positive or negative aspects of using the game in your opinion? If so, which one(s)?

_____

_____

9. **Do you have any idea on how to improve the game or the platform? If so, please explain.**

---

10. **This question is for any additional comment (difficulties, criticisms, and/or suggestions) about the study. We depend on your help to improve the work.**

Thanks for your collaboration!

Diego Cardoso Borda Castro
Cláudia Maria Lima Werner

Appendix B

TAM + MEEGA Questionnaire (Portuguese version)

B.1 Descrição do trabalho

Uma das indústrias que mais vem crescendo ao longo dos anos é a de jogos, atraindo entusiastas de todas as idades, gêneros e gostos, chegando a ter uma comunidade de bilhões de consumidores. No entanto, o desenvolvimento de jogos pode ser algo demorado, com muitos participantes e etapas, o que faz com que alguns títulos levem anos até sua entrega final. Com uma comunidade de entusiastas tão grande, alguns consumidores não conseguem esperar tanto tempo até o lançamento do jogo. Devido a isso, acabam criando suas próprias versões do jogo. Esse procedimento de utilizar um jogo já existente para construção de um novo é conhecido como mod.

Apesar da construção de mods ser algo recorrente na comunidade de jogos, através de um estudo realizado, foi possível perceber algumas dificuldades nesse processo, onde se destacam: a falta de ferramentas especializadas para construção de mods, a dificuldade de entender o código fonte do jogo original ou até mesmo, em alguns momentos, a necessidade de recriar o jogo original do zero.

O conceito de mod se assemelha muito com o conceito de Reutilização de Software oportunista, onde um determinado software é copiado e modificado. Tendo isso em mente, foi realizado um estudo sobre jogos e Reutilização de Software, onde a abordagem de Linha de produto de Software se destacou entre as demais para a construção de mods. Devido a isso, esse trabalho visa demonstrar o conceito de linha de produto para construção de jogos por meio de dois jogos disponibilizados. O primeiro visa criar um novo jogo de forma automática de tempos em tempos através da modificação das mecânicas, dinâmicas e estéticas do jogo. O segundo permite que o próprio jogador construa seu jogo por meio da seleção das mecânicas,

dinâmicas e estéticas.

Comandos básicos e observações:

- Jogo 1
  - Utilize o joystick para movimentar o personagem
  - Evite bater nos obstáculos
- Jogo 2
  - Selecione todas as mecânicas, dinâmicas e estéticas desejadas para o jogo dentro da árvore de características. Se tiver duas opções para uma características, lembre-se de selecionar pelo menos uma.
  - Utilize o joystick para movimentar o personagem
  - Utilize os botões do lado direito para atacar ou segurar as caixas
  - Utilize o botão do lado esquerdo para pular
  - Para vencer, cumpra os objetivos do jogo descritos na tela inicial

B.2 Questionário de caracterização

Por favor, responda as questões abaixo com base na experiência que obteve ao utilizar os jogos. Todos os dados coletados são apenas para melhoria da pesquisa e serão publicados de forma totalmente anônima, não comprometendo o participante.

11. Formação Acadêmica:

- ☐ Doutorado concluído
- ☐ Doutorado em andamento
- ☐ Mestrado concluído
- ☐ Mestrado em andamento
- ☐ Graduação concluída
- ☐ Graduação em andamento
- ☐ Outro (Qual): _____

12. Faixa etária:

- ☐ Menos de 18 anos
- ☐ 18 a 28 anos
- ☐ 29 a 39 anos

□ 40 a 50 anos

□ Mais de 50 anos

13. Experiência

Conside: (1) Sem experiência com a atividade; (2) Possui conhecimento teórico, sem prática; (3) Pratiquei em projetos em pessoais ou em sala de aula; (4) Pratiquei em projetos na indústria; (5) Possui um vasto conhecimento

	1	2	3	4	5
Jogos digitais	☐	☐	☐	☐	☐
Desenvolvimento de jogos	☐	☐	☐	☐	☐
Mods	☐	☐	☐	☐	☐
Reutilização de Software	☐	☐	☐	☐	☐
Linha de produto de Software	☐	☐	☐	☐	☐

14. Tempo de experiência

Por favor, detalhe sua resposta. Inclua o número de meses de experiência para cada uma das áreas de conhecimento.

Tecnologia	Meses
Jogos digitais	
Desenvolvimento de jogos	
Mods	
Reutilização de Software	
Linha de produto de Software	

B.3 Questionário de avaliação

15. Usabilidade

	1	2	3	4	5
O design do jogo é atraente (tabuleiro, cartas, interfaces, gráficos, etc.).	☐	☐	☐	☐	☐
Os textos, cores e fontes combinam e são consistentes.	☐	☐	☐	☐	☐
Eu precisei aprender algumas coisas antes que eu pudesse jogar o jogo.	☐	☐	☐	☐	☐
Eu considero que o jogo é fácil de jogar	☐	☐	☐	☐	☐

As regras do jogo são claras e compreensíveis.	☐	☐	☐	☐	☐
As fontes (tamanho e estilo) utilizadas no jogo são legíveis.	☐	☐	☐	☐	☐
As cores utilizadas no jogo são compreensíveis.	☐	☐	☐	☐	☐

16. Experiência

	1	2	3	4	5
Este jogo é adequadamente desafiador para mim.	☐	☐	☐	☐	☐
O jogo oferece novos desafios (oferece novos obstáculos, situações ou variações) com um ritmo adequado.	☐	☐	☐	☐	☐
O jogo não se torna monótono nas suas tarefas (repetitivo ou com tarefas chatas).	☐	☐	☐	☐	☐
Eu recomendaria este jogo para meus colegas.	☐	☐	☐	☐	☐
Eu me diverti com o jogo.	☐	☐	☐	☐	☐
Houve algo interessante no início do jogo que capturou minha atenção.	☐	☐	☐	☐	☐
Eu estava tão envolvido no jogo que eu perdi a noção do tempo.	☐	☐	☐	☐	☐
Eu esqueci sobre o ambiente ao meu redor enquanto jogava este jogo.	☐	☐	☐	☐	☐
O conteúdo do jogo é relevante para os meus interesses.	☐	☐	☐	☐	☐

17. Utilidade da ferramenta a ser proposta:

	1	2	3	4	5
Eu compreendi facilmente como usar a abordagem de SPL?	☐	☐	☐	☐	☐
Eu usei a abordagem da maneira correta? Criei os jogos que gostaria	☐	☐	☐	☐	☐
Compreendi o que aconteceu na interação com a ferramenta?	☐	☐	☐	☐	☐
Eu notei a facilidade de criar um novo jogo por meio de Linha de produto?	☐	☐	☐	☐	☐

Caso existisse a ferramenta a ser proposta, eu usaria uma ferramenta dessa para expandir jogos?	☐	☐	☐	☐	☐

18. De acordo com sua opinião, foi identificado algum aspecto positivo / negativo da utilização do jogo? Se sim, qual(ais)?

_____

_____

19. Você possui alguma sugestão para melhoria do jogo ou da plataforma? Em caso positivo, por favor, especifique-a.

_____

_____

20. Este espaço é reservado para quaisquer comentários adicionais (dificuldades, críticas e/ou sugestões) a respeito do estudo executado. Contamos com sua contribuição para que o trabalho seja aprimorado.

_____

_____

Obrigado pela sua colaboração!

Diego Cardoso Borda Castro
Cláudia Maria Lima Werner