CIMAT

Centro de Investigación en Matemáticas A.C.

Método para el Desarrollo Ágil de Videojuegos Guiado por Experiencias

TESIS

para obtener el grado de

Doctor en Ciencias

con orientación en

Ciencias de la Computación

PRESENTA:

Mario González Salazar

DIRECTOR DE TESIS:

Dr. Cuauhtémoc Lemus Olalde

CODIRECTORES DE TESIS:

Dr. Hugo Arnoldo Mitre Hernández

Dr. José Luis González Sánchez

Marzo 18 del año 2016 Guanajuato, Gto.

An Experience-driven Method for Video Game Agile Development

by

Mario González Salazar

DISSERTATION
Presented to the Faculty of
the Center for Research in Mathematics
in Partial Fulfillment of the Requirements
for the Degree of

DOCTOR OF PHILOSOPHY IN
Computer Science

Center for Research in Mathematics
Guanajuato, México

March 2016

I dedicate this dissertation to my wife Soledad, my son Mario, my mother Carmen, my Father Jesus Ma., and my sister Alejandra, who have always supported and believed in all my dreams and have allowed me to live my biggest dream which is to share my life with them.

ACKNOWLEDGEMENTS

I want to thank God for allowing me to study a field that I love. Thank my family who have always believed and supported me, to my supervisor Dr. Cuauhtémoc Lemus that has guided and challenged me in my research, to my co-directors Dr. Hugo A. Mitre whose support allowed better structure the thesis and streamline the time spent in my research and Dr. José Luis González whose extensive knowledge of my subject of study allowed me to better understand the details and directives of the thesis.

I also want to thank my colleagues at the CIMAT for all the ideas, questions, support and motivation particularly to Dr. Carlos A. Lara Alvarez. I want to thank to the external researchers too, especially to Dr. Diego Martin whose valuable advice helped to validate the proposal.

Part of my doctoral studies was developed at the University of Granada at Granada, Spain. I would like to express my gratitude to Dr. Francisco Luis Gutiérrez Vela and all the members of the Software Specification, Development and Evolution Research Group (GEDES for its acronym in Spanish) whose valuable contributions help me enrich my contributions.

I gratefully acknowledge the financial support for my research provided by CONACYT and CIMAT during my doctoral studies.

An Experience-driven Method for Video Game Agile Development

Mario González Salazar

Center for Mathematical Research CIMAT, 2015

Supervising Professors: Cuauhtémoc Lemus Olalde, Hugo Arnoldo Mitre Hernández, José Luis González Sánchez.

ABSTRACT

The video game industry today provides many facilities to micro and small businesses to create and publish their games, this thanks to the opening of new markets, mainly the social networking and mobile technology.

These emerging companies do not always have the knowledge and experience in game development and must rely on tools that facilitate and guide the process of game development. While the market has a number of tools, there is little to support the design of a video game. In literature you can find books on game design, but the level of detail is insufficient, and none of them relate the desired experiences that the game should evoke on the player and how to design the game according to these experiences.

The objective of this research is to provide a method that allows the use of the desired experiences to develop a video game with an agile approach, whose design is properly structured and have an appropriate level of detail for implementation. Achieving this objective will reduce rework while developing a game and help the final product to evoke the desired experiences in the player.

This research is motivated by the need of a method to create game design that can be used to implement the game and the need to take into account the experience that the player should have to create this design.

This research produced a method to transform the idea of a video game on a concept, define the experiences you want the game evoke on the player based on this concept and from these experiences guide the design and development of the game for better compliance of its objectives.

During testing it was found that the use of the proposed method reduces rework occasioned by poorly detailed descriptions and badly structured game design elements, also shown promising results in improving experiences that the players have while playing the game.

Keywords: Agile Development, Agile Game Development, Computer Game, Game Design, Game Design Document, Game Development, GDD, Human-Computer Interaction, HCI, Playability, Player Experience, Player Experience Evaluation, Px, Quality Attribute, Requirements Engineer, Rework, Software Engineer, Software Requirements, User Experience, User Experience Evaluation, Ux, Video Game.

TABLE OF CONTENTS

List of Figures

List of Tables

CHAPTER 1
INTRODUCTION

This chapter provides a summary of the entire dissertation. Additionally, this chapter includes the motivation behind this research effort, a description of the problem addressed, and an overview of the proposed solution, results, and future research works.

1.1 Motivation

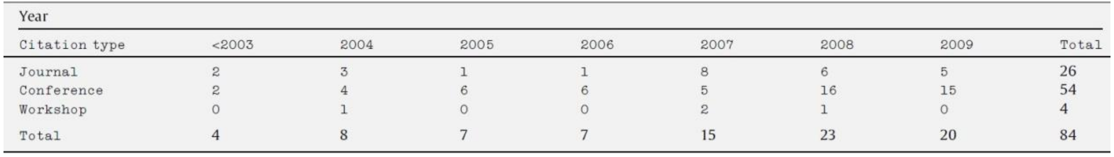
This section starts by discussing the relevance of video game industry as the main entertainment industry and as an important research area. Next, the new markets that mobile device and social networks open is presented. Finally, the complexity involving game development is discussed.

1.1.1 Importance of Video Games

Video games are important economically, as innovative leaders and as an alternative to resolve issues outside the entertainment arena. Video games constitute the main entertainment industry, with continuous growth since their appearance and billions of dollars in sales and revenues (*Essential Facts about the Computer and Video Game Indsutry*, 2015). Since its inception video games have evolved with technology, with every new generation of consoles bringing innovations. The search to generate new and better experience for gamers has created new ways of perceiving and interacting with games. Consoles like WII™, or the X-Box 360 Kinect™, have opened a new range of possibilities for interacting with a game, causing players to take a more active role by making them move more than their fingers to control the game. This technology has proven useful outside of the gaming area too, with applications in areas such as architecture, modeling or 3d modeling ("Top 10 Best Kinect Hacks," 2011). Video games have proven useful outside the entertainment area. Serious games have shown good results solving problems, training, diagnosing, predicting, and teaching among others (Jason, 2013).

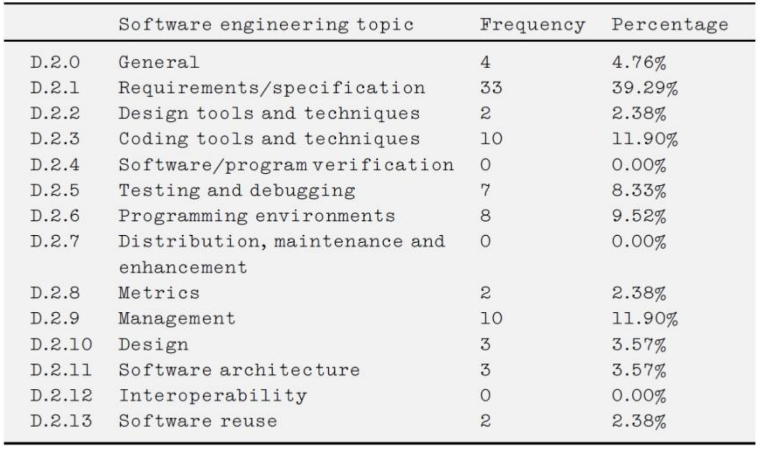
The video game domain as a research topic is growing too. Video games are a relatively new field, and at the moment there are no standards even in for a definition of what is a video game. But there is some common agreement in the use of some terms, from software engineering or other domains like the movie industry. Table 1-1 shows that in the last decade the number of published papers on ACM, IEEE, Springer and Elsevier related to video games has grown from 4 in 2003 to 20 in 2009. Table 1-2 shows the area in which this research has focused. The areas of interest for this research proposal are D.2.1 Requirements/specification and D.2.2 Design tools and techniques.

Table 1-1 Research Activity per Year and Citation Type(Ampatzoglou & Stamelos, 2010)



Year								
Citation type	<2003	2004	2005	2006	2007	2008	2009	Total
Journal	2	3	1	1	8	6	5	26
Conference	2	4	6	6	5	16	15	54
Workshop	0	1	0	0	2	1	0	4
Total	4	8	7	7	15	23	20	84

Table 1-2 Software Engineering in Games Research Topics (Ampatzoglou & Stamelos, 2010)



	Software engineering topic	Frequency	Percentage
D.2.0	General	4	4.76%
D.2.1	Requirements/specification	33	39.29%
D.2.2	Design tools and techniques	2	2.38%
D.2.3	Coding tools and techniques	10	11.90%
D.2.4	Software/program verification	0	0.00%
D.2.5	Testing and debugging	7	8.33%
D.2.6	Programming environments	8	9.52%
D.2.7	Distribution, maintenance and enhancement	0	0.00%
D.2.8	Metrics	2	2.38%
D.2.9	Management	10	11.90%
D.2.10	Design	3	3.57%
D.2.11	Software architecture	3	3.57%
D.2.12	Interoperability	0	0.00%
D.2.13	Software reuse	2	2.38%

1.1.2 Video Games and New Markets

The rise of social networks and mobile devices have opened new markets for the gaming industry (*Essential Facts about the Computer and Video Game Indsutry*, 2015, “Infographic: Mobile Gaming Statistics 2011,” 2011). Today it has become easier to sell video games digital copies via Internet; online stores allow independent developers to sells their apps without the need of belonging to a company. This has encouraged the emergence of new companies and independent developers trying to compete in these markets. Even companies like Sony™ and Microsoft™ have favored the publication of independent developers for their consoles. Despite this boom, a lot of games have problems that make the games unsuccessful or even unfinished (Carless, 2009). These problems can be described as challenges that can be addressed by software engineering in the field of game development (Kanode & Haddad, 2009).

1.1.3 Video Games Complexity

Game development has unique features that make it complex. While sharing many similarities with traditional software development, its ludic and multi-disciplinary nature hinders game development. Creating better experiences for gamers has become an important topic of research in recent years for both fields the human computer interaction and the requirements engineering (Ampatzoglou & Stamelos, 2010; *Evaluating User Experience in Games: Concepts and Methods (Human-Computer Interaction Series)*, 2010; J. González, Padilla, & Gutiérrez, 2009; Korhonen, Paavilainen, & Saarenpää, 2009; Nacke, Drachen, Kuikkaniemi, & Kort, 2009). Of the three major phases in game development (pre-production, production and post-production), it is in pre-production where the main challenges and unique characteristics emerge. This has resulted in most research on game development focusing on pre-production (Ampatzoglou & Stamelos, 2010).

Video games have evolved since their popularization in the seventies. Now we have many different games and many different platforms. Figure 1-1 presents the evolution of video games delineating the different trends that video games have had over time. This evolution has not only been in types of games or platforms but in complexity too. The time to produce a game, the number of people with in different disciplines needed and the number of work products needed have increased also. What one programmer could do in a couple of months in the past now may require years and hundreds of people. The complexity it is not only because the games are larger, but also because the different disciplines involved in producing a game have evolved. Table 1-1 can help illustrate how complex a game can be. Every choice of genre, number of players, platform, etc., can be combined bringing different disciplines needed. As example, a Role Playing Game (RPG) may need a professional writer. If multiple players on a Local Area Network (LAN) can play the game, it may need some network expertise. If its graphics will be in a 3-Dimensional environment it may need a 3D modeling expert. And so on. The problems emerging from this complexity lead to new opportunities for research topics.

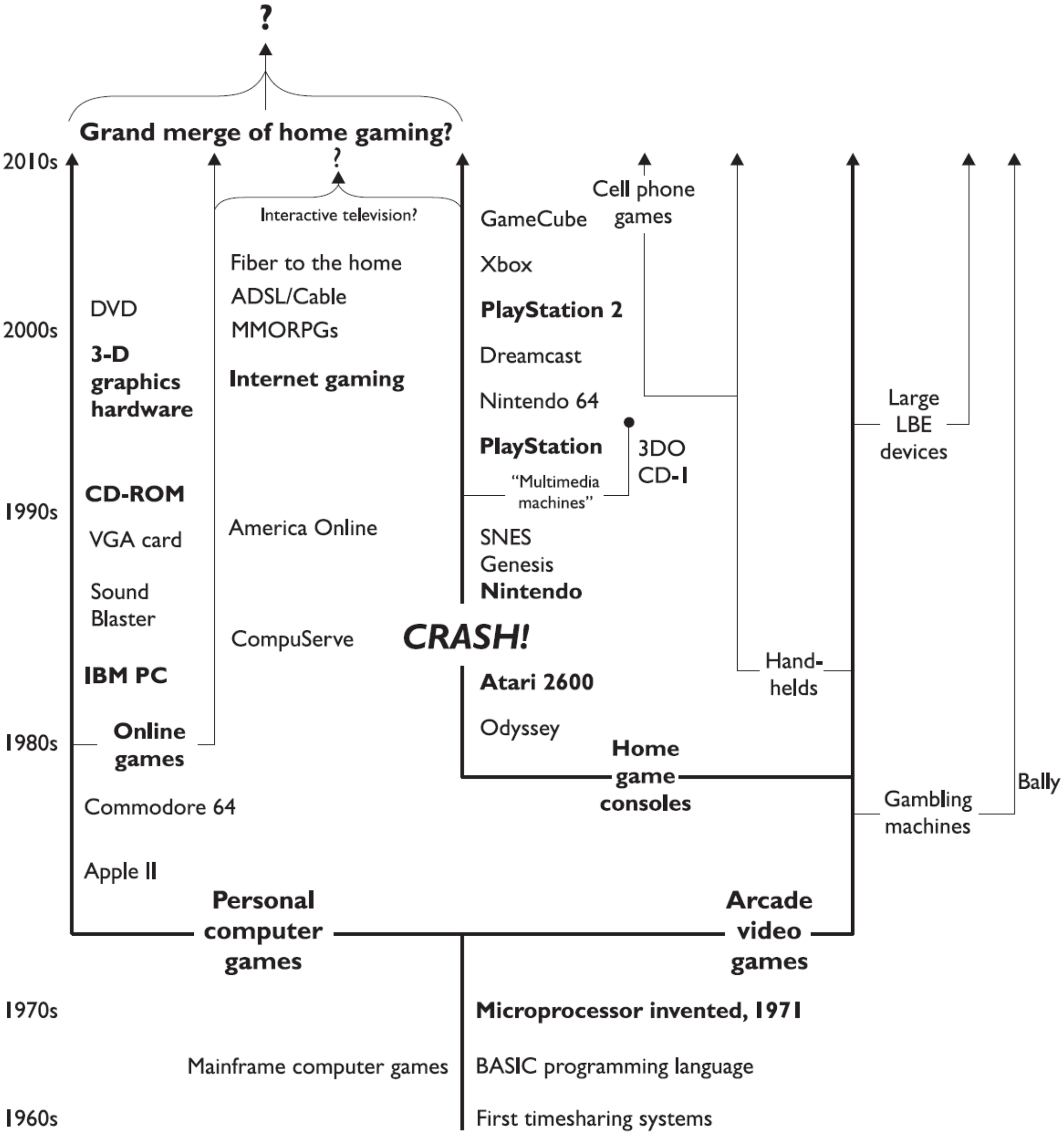
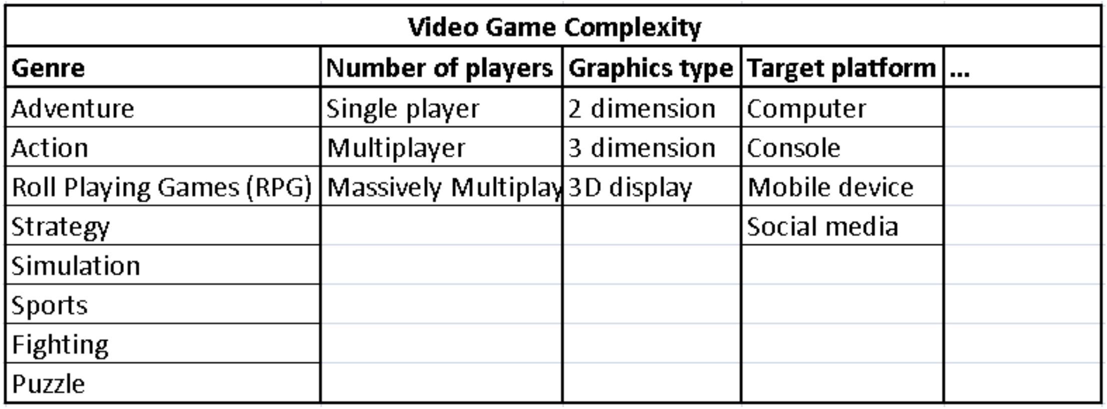


Figure 1-1 Video Games Evolution (Adams, 2003)

Table 1-3 Video game complexity



Video Game Complexity				
Genre	**Number of players**	**Graphics type**	**Target platform**	**...**
Adventure	Single player	2 dimension	Computer	
Action	Multiplayer	3 dimension	Console	
Roll Playing Games (RPG)	Massively Multiplay	3D display	Mobile device	
Strategy			Social media	
Simulation				
Sports				
Fighting				
Puzzle				

1.2 Problem Description and Research Questions

This section begins by defining the basic concepts used in all the chapters. The research context, the problem description, and the research significance of this work are discussed.

1.2.1 Basic Definitions

A video game is a software created to entertain, with a specific set of rules, based on the interaction between one or more persons and an electronic device associated with the software (Tavinor, 2008).

Pre-production is the stage in video game development mainly focusing on creating the game concept and game design (Bethke, 2002) .

Production is the stage in video game development that sets its sights on software creation and validation of all the game details defined in the pre-production stage. Work products such as sounds, music, cinematic, need to be integrated in a video game product which is later tested (Bethke, 2002).

Post-production is the stage in video game development focusing on video game distribution, maintenance and feedback management coming from different sources, like specialized video game reviewers, and video game forums. Sometimes an analysis called a postmortem is performed in order to determine the do´s and don'ts during game development (Bethke, 2002).

User Experience (Ux) is "every aspect of the user's interaction with a product, service, or company that makes up the user's perceptions of the whole. User experience design as a discipline is concerned with all the elements that together make up that interface, including layout, visual design, text, brand, sound, and interaction. UE works to coordinate these elements to allow for the best possible interaction by users" (The User Experience Professionals' Association (UXPA), 2012).

Player's Experience (Px) is the sensations experienced by a player while playing a video game (J. González & Padilla, 2008).

Gameplay is the degree and nature of the interactivity that the game includes. It describes how the players interact with the game and how the game responds to them. The gameplay is not influenced by the story or the setting in which the game take place (Rouse, 2004).

Playability is "a set of properties that describe the Px using a specific game system whose main objective is to provide enjoyment and entertainment, by being credible and satisfying, when the player plays alone or in company" (J. González & Padilla, 2008).

A game design driver is high-level property that the game should have in order to generate the intended experience in the player.

A game design guideline is a description of how game elements need to be created in order to achieve the intended experience established in the game design drivers.

Game mechanics is the description of the game elements and the rules by which they interact.

Game dynamics is the description of the challenges, goals, rewards, and interfaces among others that define the possible interaction of the player with the game.

Game aesthetics is the description of what the player perceives with his senses, especially visual and auditory aspects.

1.2.2 Context

The work proposed targeted mainly the pre-production stage, where goals and game design is produced.

The work proposed is best suited for:

- Creating video games that have a progression path with a beginning and end, divided by levels, missions or chapters.
- Small teams that create games with a high level of uncertainty in the development process.

- Emerging video game companies or software companies interested in video game development, with little experience in game design.

1.2.3 Problem Statement and Research Questions

The problem addressed by this dissertation is the lack of formal and detailed documentation and the lack of Px handling in game design in the pre-production stage.

The first part of the problem is the lack of formal and detailed documentation, pointed out by Callele et al. in his work " Requirements engineering and the creative process in the video game industry " (Callele, Neufeld, & Schneider, 2005) and later confirmed in " A report on select research opportunities in requirements engineering for videogame development" (Callele, Neufeld, & Schneider, 2011) where they indicate the need of a way to describe the game design with enough formality and detail so it can be used as an Software Requirements Specification (SRS) document to implement the software part of the game. Petrillo et al. expose similar problems in "Houston, we have a problem...: A Survey of Actual Problems in Computer Games Development " and “What Went Wrong? A Survey of Problems in Game Development" (Fabio Petrillo & Pimenta, 2009; Fábio Petrillo & Pimenta, 2008) where they analyze game post mortems and find that 65% of them report game design problems related to unspecified or ambiguous requirements in game design. When this problem appears in the production stage the developers have two options: ask the game designer for clarification of the missing information or make his or her best assumption and implement based on that. Either way necessitates rework by defining again the requirement to be implemented or by finding in latter stages that the assumptions about the requirements were wrong and the necessity of having to make changes.

The second part of the problem is the lack of Px handling in early stages in game development. There are some efforts related to evaluating the Ux in games, in works like the one by González Sánchez et al. (J. L. González, Montero, Padilla, & Guitiérez, 2009) where the authors use the playability concept to characterize the Px in different attributes and for each attribute they define a set of metrics in order to extend the concept of quality in use to integrate playability, or by Takatalo et al. (Takatalo, Häkkinen, Kaistinen, & Nyman, 2010) where they breakdown the concepts of presence, involvement and flow into smaller concepts that are evaluated by the EVEQ-GP (Experimental Virtual Environment Experience Questionnaire-Game), which needs to be applied in a final or semi-final stage of the game. Most of the work related to Px or Ux in games is focus on evaluating the experience and in many cases this evaluation occurs when the game is final or semi-final, which means that changing the game at this point to improve the experience is more expensive than doing it in early stages. This problem has been identified by McAllister and White (McAllister & White, 2010) where the authors point out that there is a need for handling Ux in early stages in game development. Evaluating experience is not enough; it is necessary to manage the experience that the developer wants the game to bring to the player and design the game taking into account these desired experiences. Callele et al. (Callele et al., 2011) has identified this problem as a research opportunity in the field of requirements engineering in video games.

The main research questions derived from the previous problem are stated as follows:

1. Is it possible to decrease rework while developing a game, by formalizing the design with a Game Design Document (GDD) that has a clearly defined structure, relations and details, and incorporates the SRS best practice?
2. Is it possible to improve the player experience that the game brings by managing, handling and tracking the desired player experiences in early stages of game development?

To answer these questions this dissertation aims to create a solution that can be integrated into the game development process that can be useful to small and emerging game development teams that need guidance in creating a video game while avoiding the problem mentioned in this section.

1.3 Structured Video Game Design Based on Player's Experience

This section describes an overview of the solution proposed to answer the research questions stated in Section 1.2.3. First, the assumptions about the context where the solution can be used are described. This information is complemented with a description about the scope of this work and the rationale behind the proposal. Next, an overview of the proposal is described. Finally, it is shown how the proposal can be used to improve quality, usefulness and Px in game design.

1.3.1 General Assumptions and Scope

The assumptions and constraints defined for this work are classified into two groups: information about the development team and information about the game to be created.

The assumption about the development team is:

- There is a complete game development team that can fulfill the roles required to create the game.

The assumption about the game to be created:

- The gameplay of the game can be described at least in terms of challenges and objectives. A game without one of these elements may be hard to design with the proposal.

The scope of this research focuses on providing a guide for creating the game design and handling the desired experiences that the game brings to the player. The following are out of scope:

- Provide a specific tool to measure the experience. Since the ways to measure the experience can vary depending on many variables such as the type of game, the experience to be evaluated, the resources of the team, to mention a few. It is better to let the team to choose their own Px evaluation tool.
- Evaluate non-player experience related areas. The proposal does not provide support for evaluating other areas related to quality assurance, which means that it cannot replace important activities like the testing phases (unit test, integration test, alpha and beta test).

These are the general assumptions and scope. Other specific assumptions and restrictions are discussed as required in each chapter.

1.3.2 Rationale

The game designs is reflected in the GDD, but despite the large amount of information that exists on game design, is not easy to find examples or templates of this document and the few templates that exist are very different from each other. These GDDs do not satisfy the formality and structure needed to implement the game in the production stage.

A GDD based on a taxonomy that integrates the key practices in game design and complements it with proven practices from requirements engineering, can help solve the problem of informality and structure that exists in the game design.

The current practice is to test the Px at the end of production or in post-production and they focus on evaluating whether or not the experience is positive or negative. Learning that the Px that the game brings is not the desired one when the game is finished may result in further expense and prove hard to fix.

The creation of a method that is able to manage the desired experiences that the designer wants the player to have while playing the game, a method which is capable of tracking and validating them through the game elements in the game development process and shape the game design to bring these experiences in early stages of game development, can help to solve the problem of difficulty and cost when the game does not generate the desired experiences.

The ideas described in this section represent a novel strategy for addressing the problem described in Section 1.2.3. Just as requirements engineering has tools to handle quality attributes to handle requirements, this work proposes the use of a methodology that can handle Px and then shape the game design into a structured GDD so the game can bring the desired experiences.

1.3.3 The Structured Video Game Design Based on Player's Experience Overview

The elements of our proposal have the following relationship:

- The development of a game is done hoping to achieve certain goals.
- These goals should help determine which is the desired Px that is reflected in game design drivers as properties of the game.
- These drivers are detailed in guidelines that describe how specific parts of the game should be created in order to achieve the driver property.
- The guidelines are related directly to parts of the GDD in order to track them when the related game elements are created and validate them when the game elements are finished.
- The player experiences are evaluated with different test cases in different stages to confirm that the game is achieving the goals, desired experiences and the guidelines are helping to achieve this end.

Figure 1-2 shows these relations between concepts.

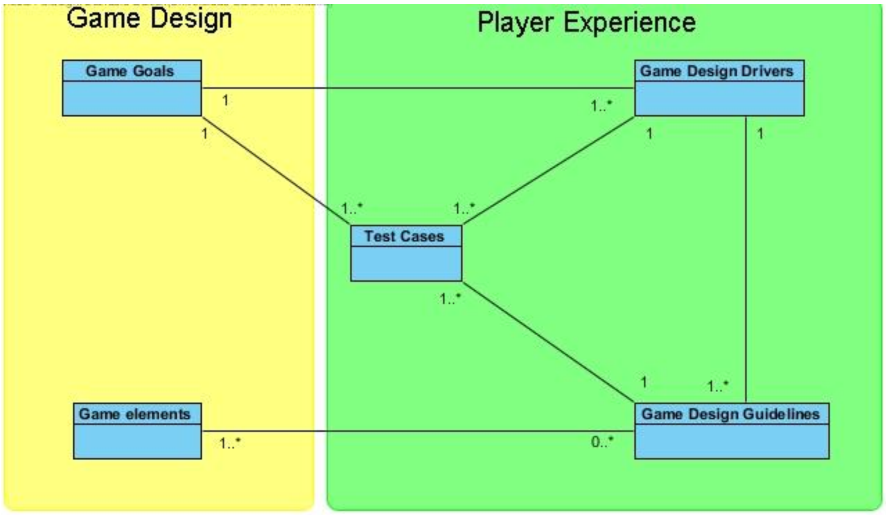


Figure 1-2 Main Concepts Relationship

Doing an analysis of several available GDDs found in the literature, which were contrasted with the Software Requirements Specification (SRS) best practices, created the GDD. The improved GDD incorporates a common understanding of terms, quality assurance, decision-making, traceability, definition of relations, boundaries, limitations and knowledge of game elements. Finally, to validate the proposal: the improved GDD was put side by side with a commercial GDD to be compared, tested by representing all game design elements of an existing video game and tested by creating a game using the proposed GDD and other well know GDD template and comparing them.

The game experience proposal was created by analyzing and comparing different proposals of Px in video games and identifying relevant quality attribute handling proposals. The proposal details the Px to a level, which can be directly related to game elements. We propose a method to manage, track and measure user experience through the game development process. Finally the proposal was tested by creating a game using it and comparing it with other Px handling proposal.

To make the proposal more feasible, an agile development methodology was adapted, which has proven useful in the context to which the proposal is directed. The scrum pattern Software Development Project Patterns (SDPP) was adapted for game development and to describe the proposal use in each phase described in the pattern.

1.4 Discussion and Conclusions

The previous section describes how the proposal addresses the research questions stated in Section 1.2.3. A more detailed experimentation is described in chapter 7.

The main contributions derived from answering the research questions are:

- A game design taxonomy, based on an analysis on GDDs proposal and requirements engineering best practices.
- A GDD template with instructions and an example of its use, based on an analysis on GDDs proposal and requirements engineering best practices.
- A method to manage, track and validate Px, based on analyzing and comparing different proposals of Px in video games and identifying relevant quality attribute handling proposals.
- A software development Project Pattern (sdPP) that adapts and integrates the previous template and method in to the Scrum framework, to do agile games development.
- Two international publications: at the 17th International Conference on Computer Games (CGAMES) in 2012 (Gonzalez, Mitre, Lemus, & Gonzalez, 2012) and the 4th International Conference on Software Process (CIMPS) in 2015 (Mitre-Hernández, Lara-Alvarez, González-Salazar, & Martín, 2015).
- A prototype of a software tool to improve the use of the method proposed by this thesis.

Future research will address the following areas:

- game design ontology,
- relationship between game design and software architecture,
- Player-centric practices in the Px evaluation mainly in defining the player profile,
- evaluating how an educational model can be integrated in the mechanics and dynamics in game design,
- investigating the different data sources that can be used to evaluate the Px and identify the advantages and disadvantages of each one,
- Extending the proposal in order to use the Px to create video games that dynamically adapt to the player.

The results from evaluating the proposal help to confirm that it reduces rework while creating a game. The results also show promising data in improving the Px that a game brings, while comparing it against the counterproposal, where there was a significant improvement in several areas related to the Px.

This proposal is unique in that it addresses the problem of the lack of formal and structured documentation and Px handling in early stages in game development, by analyzing current practices and supplementing its deficiencies with proven tools in other areas, such as requirements engineering, and adapting the proposal to current game development practices such as Scrum.

The proposal provides the foundations for creating a software tool that enables a better application and validation of game design guided by desired Px.

1.5 Dissertation Structure

The dissertation is organized into seven chapters. Chapter 2 presents an overview of the state of the art work related to the problem addressed in this dissertation. Chapter 3 presents the details on how the improved GDD was created. Chapter 4 presents the details of how the Px handling proposal was created. Chapter 5 presents the integration of the proposal with Scrum. Chapter 6 validates the entire proposal by describing the experiment performed. Finally Chapter 7 presents the conclusions and future work.

To provide a guide on how to read the rest of this dissertation, Figure 1-3 shows the structure and how each chapter is connected. The connections can be described as follow: Chapter 2, sets the basis on which the proposal is created; Chapter 3 proposes a GDD to address the problem of lack of formality in game design; Chapter 4 provides a method for guiding the GDD creation based on handling the desired player experiences; Chapter 5 integrates the GDD and the player experience handling method with current game development models; Chapter 6 evaluates and validates the proposal and Chapter 7 shows the conclusions for future research areas resulting from this work.

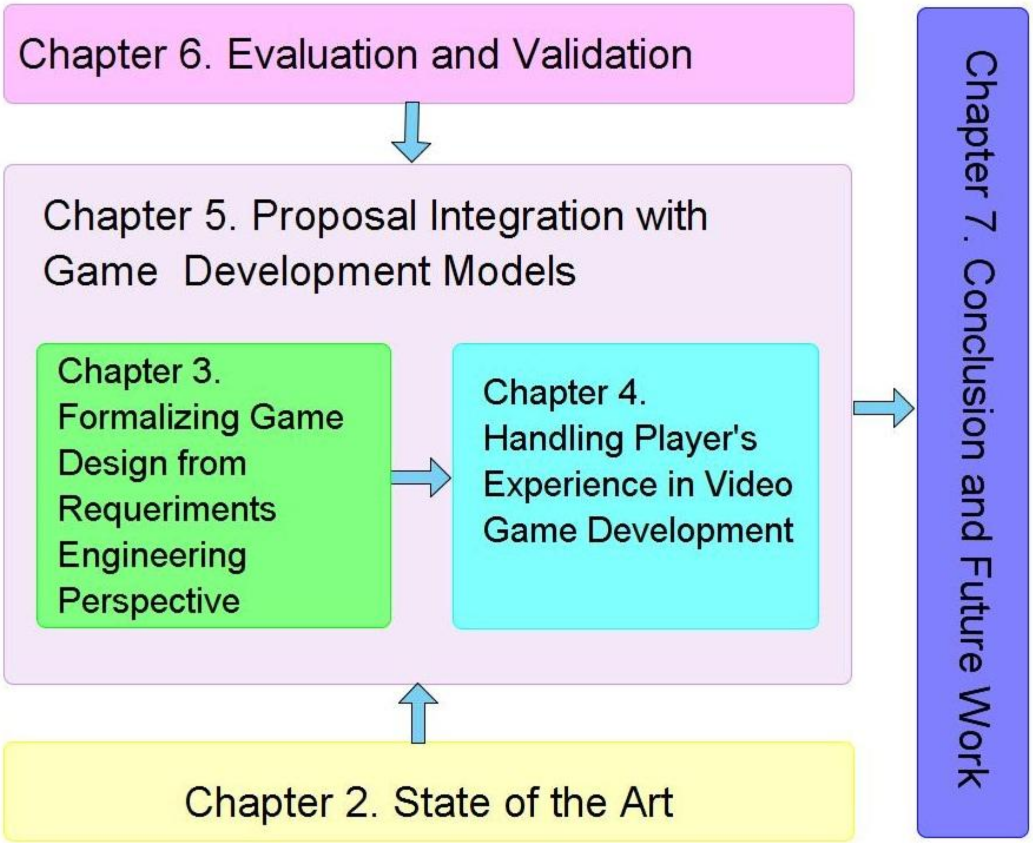

Chapter 6. Evaluation and Validation
Chapter 5. Proposal Integration with Game Development Models
Chapter 3. Formalizing Game Design from Requeriments Engineering Perspective
Chapter 4. Handling Player's Experience in Video Game Development
Chapter 2. State of the Art
Chapter 7. Conclusion and Future Work

Figure 1-3 Chapter Structure

CHAPTER 2
STATE OF THE ART

This chapter describes the state of the art work related to the problem addressed in this dissertation. The organization of this chapter is as follows: First, we give an introduction of game development, its process and roles. Second, we analyze work about game design, its key elements, its relation with requirements engineering, where game design is documented and the problems of structure, formality and relations when documenting the game design. Third, works about Player Experience (Px) are described, and we analyze different proposals on how to address player experience and point out the flaws and benefits of each one. Fourth, works about game development and agile game development are explored. Fifth, work about process patterns is presented. Finally, a summary of the most relevant discoveries is presented.

2.1 Introduction

This section describes the process, and main roles of game development. These concepts are basic to understand the rest of the document.

2.1.1 Game Development Process

There are three main stages in game development: pre-production, production and post-production. The names come from the similarity with game development and the creation of a movie. Most authors agree on these three stages in the development of video games (Adams, 2003; Bates, 2004; Bethke, 2002; Brinkkemper, Weerd, & Weerd, 2007; Callele et al., 2005; Keith, 2010; Sanchez, 2010), and while the processes described are similar, the boundaries between the activities in each stage varies from one author to another. In this dissertation, game design is included in the pre-production stage, and testing in the production stage. Figures 2-1 to 2-4 show different representations of these stages.

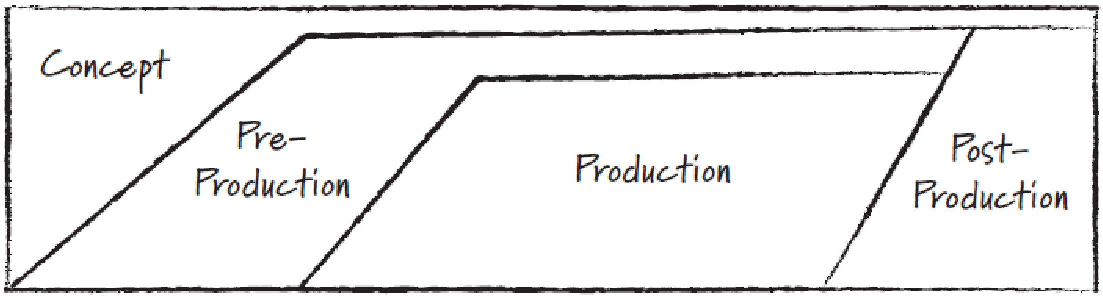


Figure 2-1 Overlapping Game Development Stages (Keith, 2010)

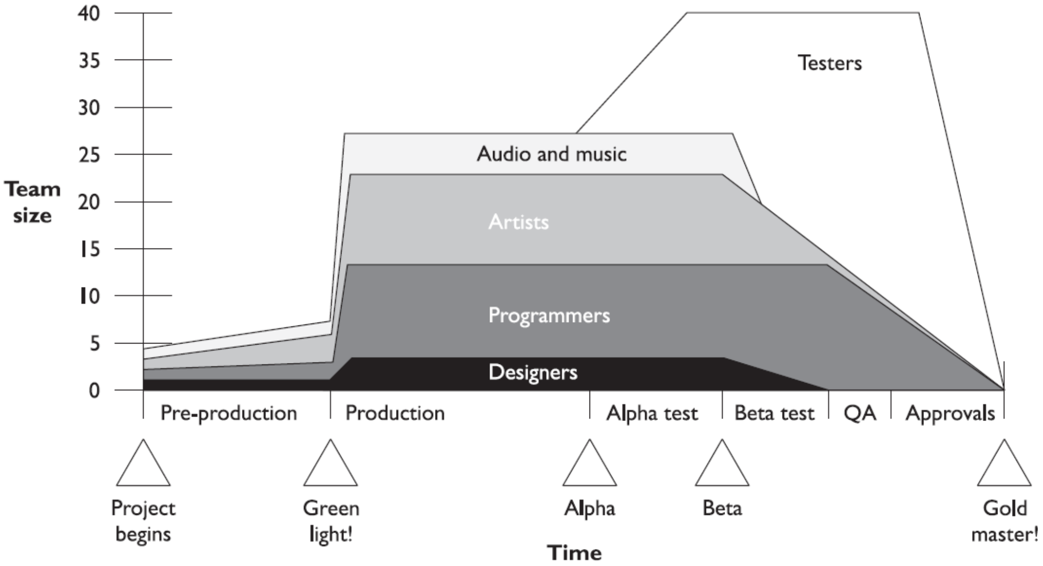


Figure 2-2 Game Development Stages (Adams, 2003)

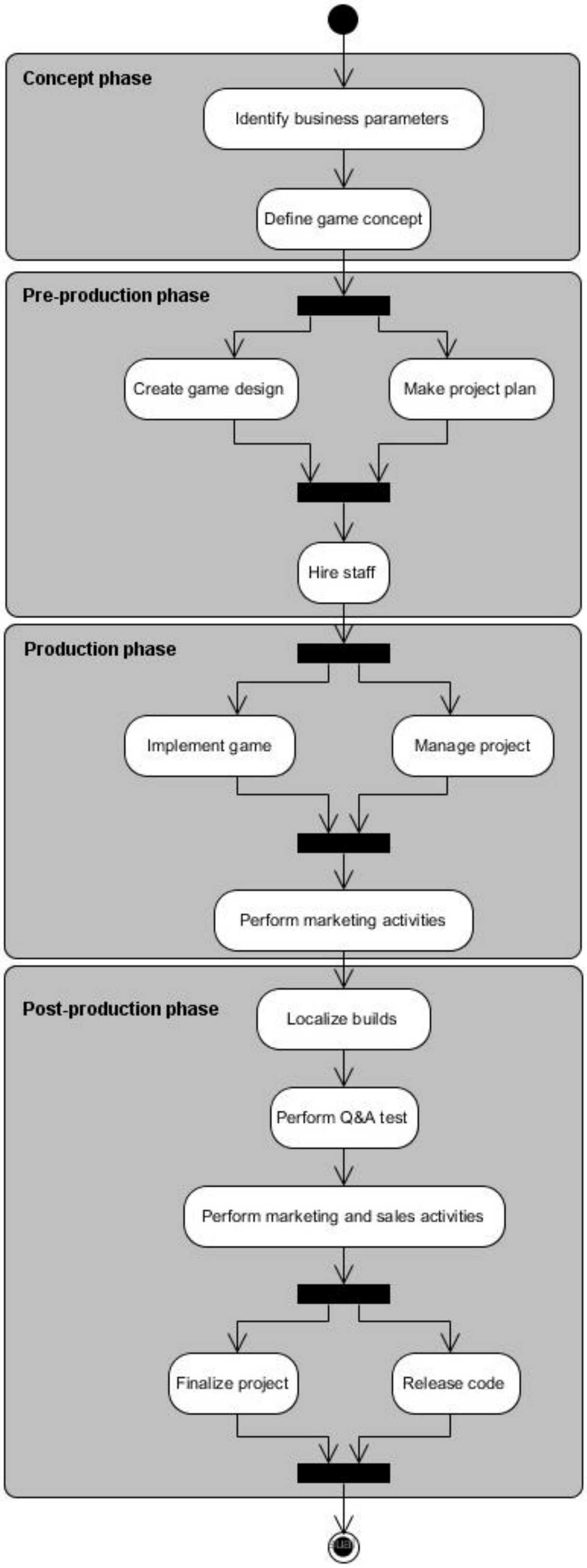


Figure 2-3 Game Development Stages and Activities(Brinkkemper et al., 2007)

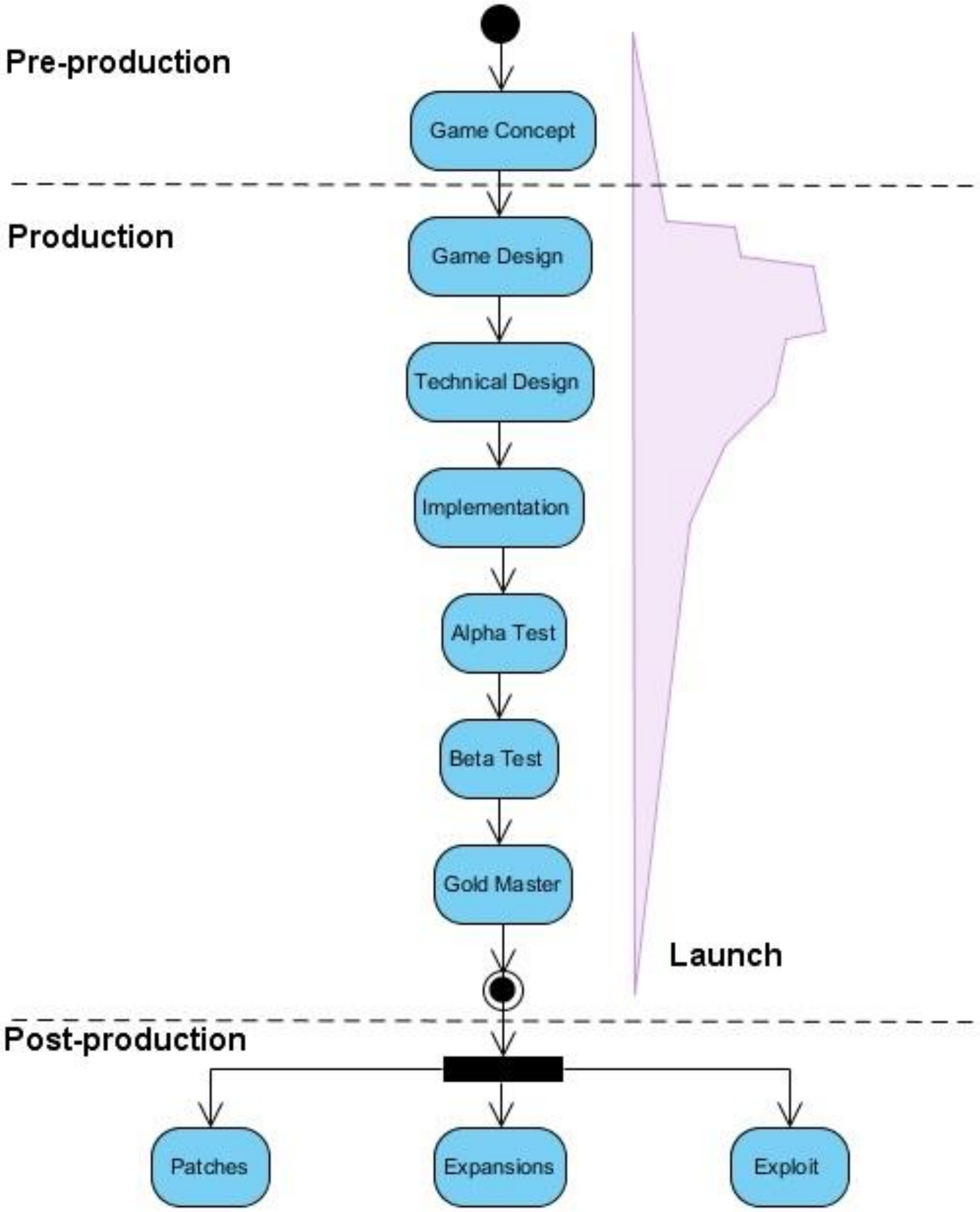
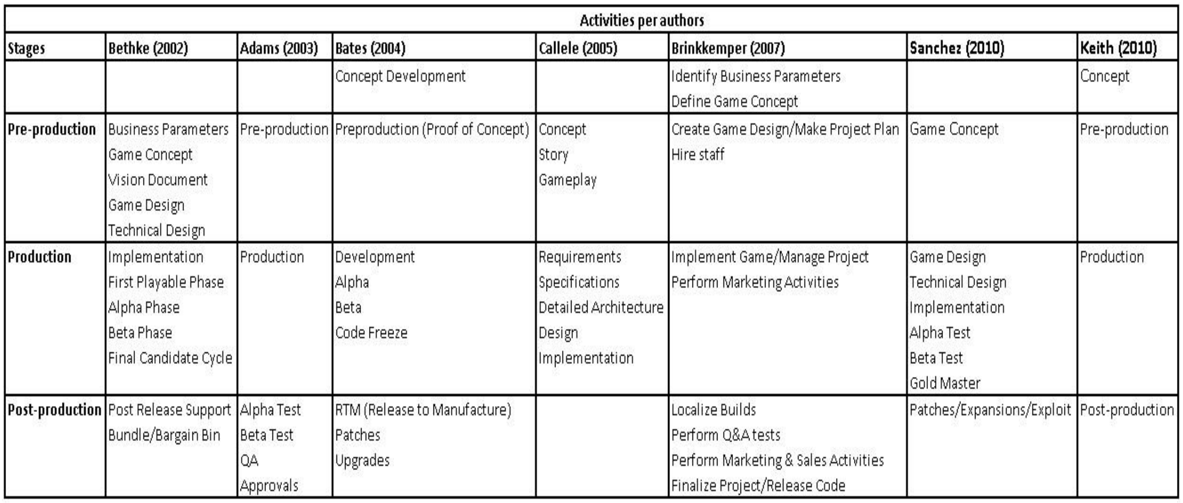


Figure 2-4 Game Development Stages and Activities (Sanchez, 2010)

This difference between stages and activities is show in Table 2-1, the main differences are: the inclusion or exclusion of the creation of the concept activity in the pre-production stage and the inclusion of the testing activities in production or post-production. The next subsections will describe the way that each stage will be treated on this dissertation.

Table 2-1 Game Development Stages and Activities Comparison



	Activities per authors						
Stages	**Bethke (2002)**	**Adams (2003)**	**Bates (2004)**	**Callele (2005)**	**Brinkkemper (2007)**	**Sanchez (2010)**	**Keith (2010)**
			Concept Development		Identify Business Parameters Define Game Concept		Concept
Pre-production	Business Parameters Game Concept Vision Document Game Design Technical Design	Pre-production	Preproduction (Proof of Concept)	Concept Story Gameplay	Create Game Design/Make Project Plan Hire staff	Game Concept	Pre-production
Production	Implementation First Playable Phase Alpha Phase Beta Phase Final Candidate Cycle	Production	Development Alpha Beta Code Freeze	Requirements Specifications Detailed Architecture Design Implementation	Implement Game/Manage Project Perform Marketing Activities	Game Design Technical Design Implementation Alpha Test Beta Test Gold Master	Production
Post-production	Post Release Support Bundle/Bargain Bin	Alpha Test Beta Test QA Approvals	RTM (Release to Manufacture) Patches Upgrades		Localize Builds Perform Q&A tests Perform Marketing & Sales Activities Finalize Project/Release Code	Patches/Expansions/Exploit	Post-production

2.1.1.1 Pre-production

The pre-production stage strives for detail in the game to be produced. How should it be seen? How should it sound? How should it be played? Why is the game entertaining? Which set of rules and goals will guide the interaction?

The main work product resulting from this stage is the Game Design Document (GDD) that needs to answer the above questions by describing the behavior and environment of the game. An important activity in pre-production is to describe user experience, which validates why someone wants to play the game. Pre-production should remove uncertainty from the video game development project by providing enough detail to create fairly accurate planning for the production stage. The detailed information should consider game mechanics (the elements of the game and the rules that guide the elements), game dynamics (the behavior of the game as a system), and game aesthetics (the intended experience on the user) as described in (Hunicke, LeBlanc, & Zubek, 2004).

In this stage a game development team mainly works on: the concept of the game, the overall summary of the main features of the game, the game design and then usually creates a prototype that can test the main features of gameplay. A basic goal to be reached in pre-production is to find the fun in the game (Keith, 2010).

2.1.1.2 Production

The production stage sets its sights on software creation and validation of all the game details defined on the pre-production stage. Work products such as sound effects, music, and cinematics, need to be integrated in a video game product, which is later tested.

The main work product of this stage is the game itself. In video game industry jargon, the gold master is the version of the game ready to be released. In this stage a game development team mainly works on: software architecture and design, coding, alpha test (the game is tested with all its features included) and beta test (the game is tested with no known bugs) (Bethke, 2002).

2.1.1.3 Post-production

The post-production stage focuses on the video game distribution, maintenance and feedback management coming from different sources, like specialized video game reviewers, and video game forums. Sometimes an analysis called postmortem is performed in order to determine the dos and don'ts learned from the game developed.

The main objectives of this stage are: monitor the performance of the game, so that any error or problem that the game that might be present can be corrected, and review game sales. Other than these objectives, post-production goals and activities can greatly vary, with a strong dependence on the platform for which this is developed. As an example there are many post-production activities involved in creating a console or computer game that can be purchased physically, as opposed to publishing a game for mobile devices in digital media. The development team is not always involved in all post-production activities in the case of a publisher; he undertakes many of these activities.

2.1.2 Game Development Roles

Roles in video game development can be categorized by discipline. Here we explain what the major disciplines are, and what kinds of role titles they hold. Aside from the main roles, the amount and variety of roles that a game can have is dependent on the size and type of game to be developed. Figure 2-5 shows some areas and roles that a game can have in its development based on Bethke work (Bethke, 2002), there is a distinction in the areas and roles that are directly related with this dissertation. A game publisher can cover some areas and roles, but if the game development studio does not have a publisher the studio should cover those areas and roles. The main disciplines in game development are: game designer, programmer, artist, producer, audio designer and quality assurance tester (Ferrier, 2008).

The game designers are responsible for the detailed vision of the game in the GDD, so that the game can be constructed from the description made in the GDD. They are responsible for defining the elements of the game, their attributes and potential actions that can be taken, so as to identify possible actions for both the player and the game to take. They create the challenges and rewards of the game as well as the flow and development in history that this should have.

The programmer or software engineers work at the coding level to make a game work. They're responsible for implementing the details described in the GDD and integrating them with the assets provided by the artists. They patch together the individual pieces of the game into what will hopefully become a fully playable piece of software by the end of the production cycle.

The artist brings to the players' eyes the vision set out in the GDD. Some of the roles that an artist can take are character design artist, user interface artist, animation artist, concept artist, and environment artist. Depending on the size of the project, a single artist can assume several of these roles or the project can have an artist responsible for each role.

The game producer is essentially the project manager of game making. Their job is to organize and facilitate the game's production. Producers create and enforce schedules and budgets. They serve as mediators between departments, and sometimes also between the studio and the publisher. They assign tasks, make sure deadlines are adhered to, and generally make sure the team has everything it needs to make the game.

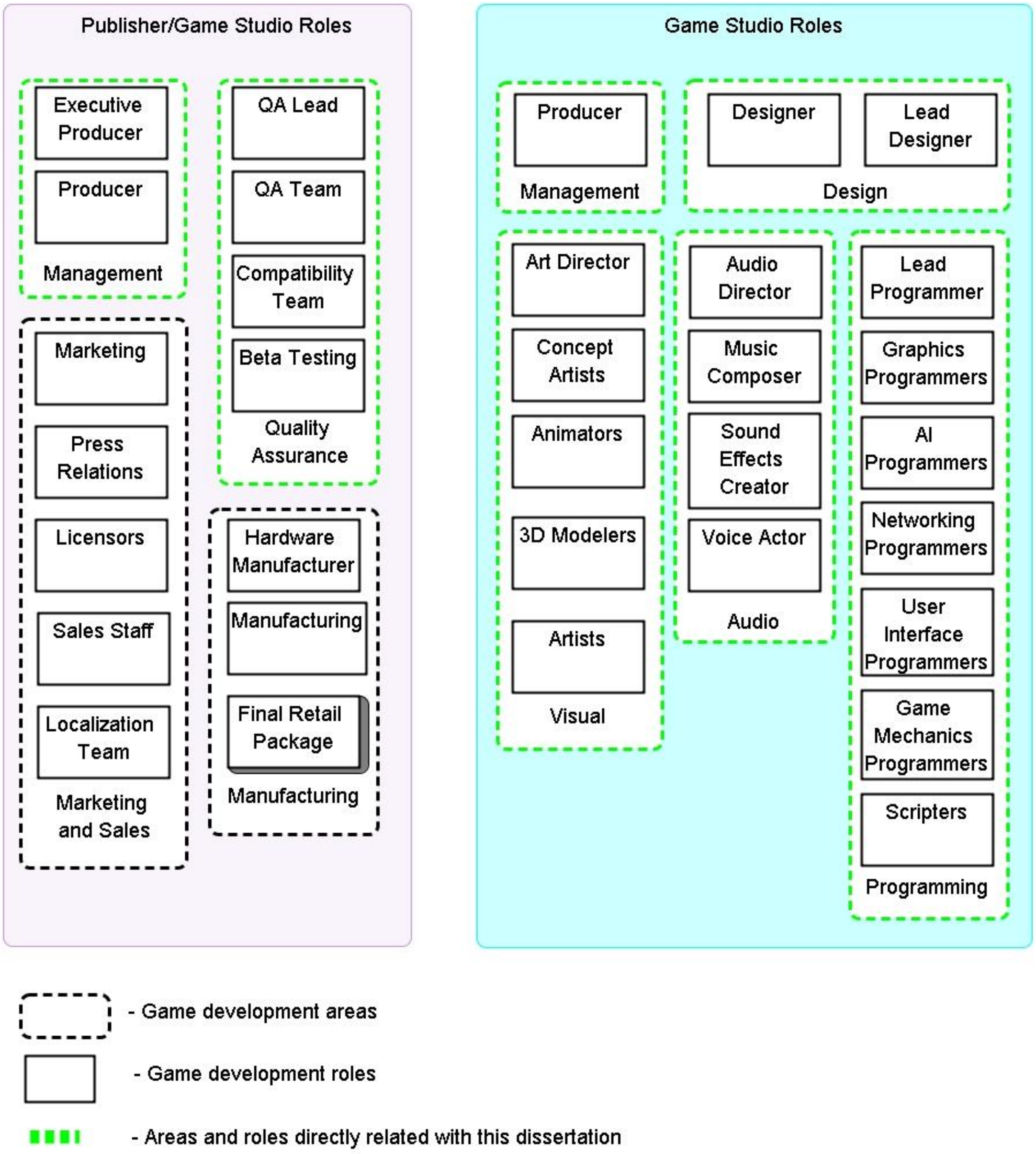


Figure 2-5 Game Development Areas and Roles

The audio designer is responsible for creating the sounds and music to match the visuals of a game. The audio designer’s objective is to give the game a unique and distinct sound, like a game’s visual style. The job is one part creative aesthetic, and one part technical. Game audio people can also be composers, writing and recording original music for the projects they work on.

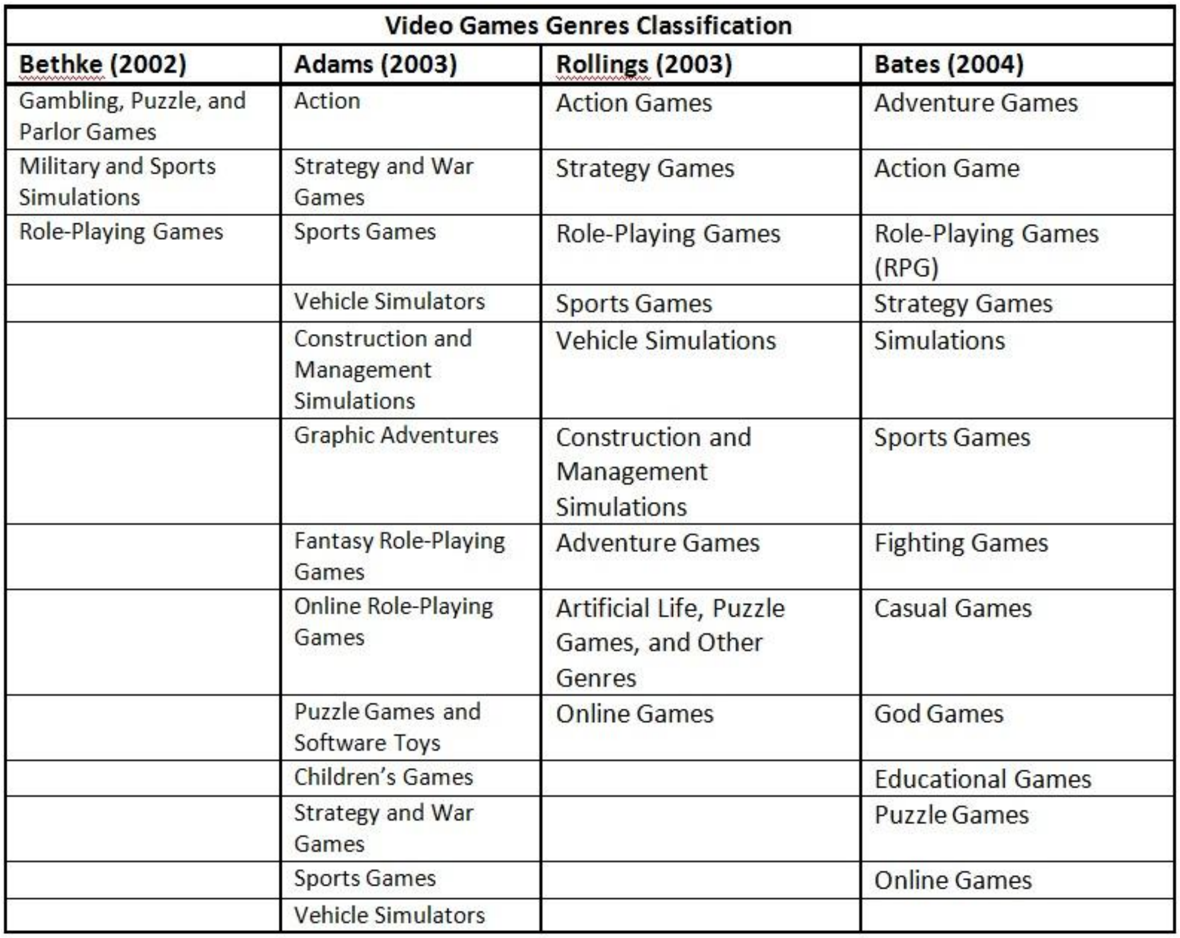
The quality assurance tester is responsible for playing the game or portions of the game while looking for and recording bugs, glitches, or other major problems. When they find a bug, they test to see if they can repeat it, and if they can, they record their bugs in writing for the programmers or artists to fix later.

2.1.3 Video Game Genres

A video game genre is a classification based on their gameplay interaction. A video game genre is defined by a set of gameplay challenges. They are classified independent of their setting or game-world content, unlike other works of fiction such as films or books (Apperley, 2006).

Using the game genre to create various tools to support the development of video games is an idea already explored in articles like "Using Genres to Customize Usability Evaluations of Video Games" (Pinelle, Wong, & Stach, 2008b)or products as "RPG Maker™"(“RPG Maker,” n.d.). It is important to note that despite the widespread use of the classification by gender in video games, there is no standard classification; games have evolved with the creation of new peripherals like Kinect™, or multiple games with hybrid or unique classifications. This evolution makes it difficult to create a standard classification. Given this ambiguity we do not recommend the use of genre to create methodologies and tools for game development. Table 2-2 illustrates this ambiguity by showing how the game genres classification can vary depending on the author.

Table 2-2 Game Genres Classification



Video Games Genres Classification			
Bethke (2002)	**Adams (2003)**	**Rollings (2003)**	**Bates (2004)**
Gambling, Puzzle, and Parlor Games	Action	Action Games	Adventure Games
Military and Sports Simulations	Strategy and War Games	Strategy Games	Action Game
Role-Playing Games	Sports Games	Role-Playing Games	Role-Playing Games (RPG)
	Vehicle Simulators	Sports Games	Strategy Games
	Construction and Management Simulations	Vehicle Simulations	Simulations
	Graphic Adventures	Construction and Management Simulations	Sports Games
	Fantasy Role-Playing Games	Adventure Games	Fighting Games
	Online Role-Playing Games	Artificial Life, Puzzle Games, and Other Genres	Casual Games
	Puzzle Games and Software Toys	Online Games	God Games
	Children’s Games		Educational Games
	Strategy and War Games		Puzzle Games
	Sports Games		Online Games
	Vehicle Simulators		

2.1.4 Game Development Challenges

Scacchi and Cooper do a review of studies, findings and practices that identify problems in the Computer Games and Software Engineering (CGSE) (Scacchi & Cooper, 2015). The authors present a list of challenges and problems in these areas:

- Using games to solve challenge problems in large-scale software engineering
- Game software requirements engineering
- Game software design
- Game software testing
- Teamwork processes and game jams in CGSE
- Global software development and global CGSE
- Game-based software engineering education

They present other research areas not as challenges or problems, but as opportunities:

- Automated generation of computer games
- Cloud-based game software services
- Game software repositories and data management services

Some of these opportunities were reported on past studies, some new ones are the result of emerging technology and the evolution of the game industry. The proposal presented in this work aims to contribute in the game software requirements engineering, the game software design and the game software testing challenges and the game software repositories and data management services opportunity.

2.2 Game Design Structure and Formality

This section gives an overview of game design, its relation with requirements engineering, the relation between game experience and game design and where the game design is documented.

2.2.1 Game Design Overview

The game design is the central part in game development. This activity transforms an idea into a detailed description of the game to create. It is this design that serves as a blueprint for creating the different assets that make up the game.

While there is no standard definition of game design, in this work, we use the following definition given by Ernest Adams and Rollings Andrew:

"Game design is the process of imagining a game, defining the way it works, describing the elements that make up the game (conceptual, functional, artistic, and others), transmitting that information to the team that will build the game" (Rollings & Adams, 2003).

Next relevant game design concepts are described. The concepts discussed are: gameplay, narrative, heuristics, balance and flow, and game design documentation.

2.2.1.1 Gameplay

There is no universally accepted definition of gameplay, for this dissertation the definition given by Rollings and Adams(Rollings & Adams, 2003) will be use as base to understand gameplay, they define gameplay as:

"One or more causally linked series of challenges in a simulated environment".

The gameplay is the main differentiator of video games with other entertainment industries, since it describes the interaction that the player will have with the game on terms of challenges, making the player an active member in the course of the game unlike music or film where this interaction does not exist. Among the main works on video game design, there are differences regarding what they consider key elements of the game design, but the gameplay is a common element that is repeated in all. All authors agree that interaction defined by the gameplay, is key to a successful game design (Bates, 2004; Crawford, 2003; Oxland, 2004; Pedersen, 2009; Rogers, 2010; Rollings & Adams, 2003; Rouse, 2004; Schell, 2008).

2.2.1.2 Narrative

Another recurring theme is the narrative (Bates, 2004; Crawford, 2003; Oxland, 2004; Rogers, 2010; Rollings & Adams, 2003; Rouse, 2004; Schell, 2008), it can be consider as the way the game and its history are presented to the player from beginning to end. Every game has a narrative and a story that can be either implicit or explicit. Implicit stories are simple and are usually present on games that don't use history as a main feature. An example is Tetris™ where the story tells how a person places figures of different shapes that are falling in a fixed space, and when they align horizontally, all sections of the aligned figures disappear. Explicit stories can go from simple to really complex, and they can be told by different means, like cinematic, audio narration or text dialogs.

2.2.1.3 Heuristics

The heuristics are elements often used to evaluate the design of video games. A heuristic can be defined as "a design guideline which serves as a useful evaluation tool" (Desurvire, Clapman, & Toth, 2004). The game heuristic can address general topics such as usability (Desurvire & Wiberg, 2009; Pinelle, Wong, & Stach, 2008a; Pinelle et al., 2008b) and gameplay evaluation (Desurvire et al., 2004; Korhonen et al., 2009) or more specific issues such as heuristics for evaluating mobile games (Korhonen & Koivisto, 2006, 2007). While heuristics are easy to apply and inexpensive, because of their generic nature, they may not always be appropriate to the game to be developed. Feedback from developers said many times the heuristics are not very useful because they are too generic (McAllister & White, 2010). The heuristics on gameplay should be treated with special care, and to apply these heuristics may be beneficial for certain games, but cannot contribute much or even harm the gameplay in other games. For example, a heuristic like "Player's fatigue is minimized by varying activities and pacing During Game play"(Desurvire et al., 2004) is not applicable to games like Tetris™ or Bubbles IQ™, because in both games the type of activity does not change. We conclude game heuristics can be used to check for omissions and to look for areas of improvement in game design, but, given its generic nature is not advisable to use them as an evaluation tool in game design.

2.2.1.4 Balance and Flow

One key element in game design is the balance of the game. This means that the challenges of the game have the appropriate difficulty. To balance the game there is a concept in video games known as the flow (Chen, 2007), which says that the difficulty of the challenges and the skills of the player should grow at the same rate. If the challenges' difficulties grow faster than the players' skills, the player gets frustrated, but on the contrary if the players' skills grow faster than the challenges' difficulties, the player gets bored. Figure 2-6 illustrates this concept.

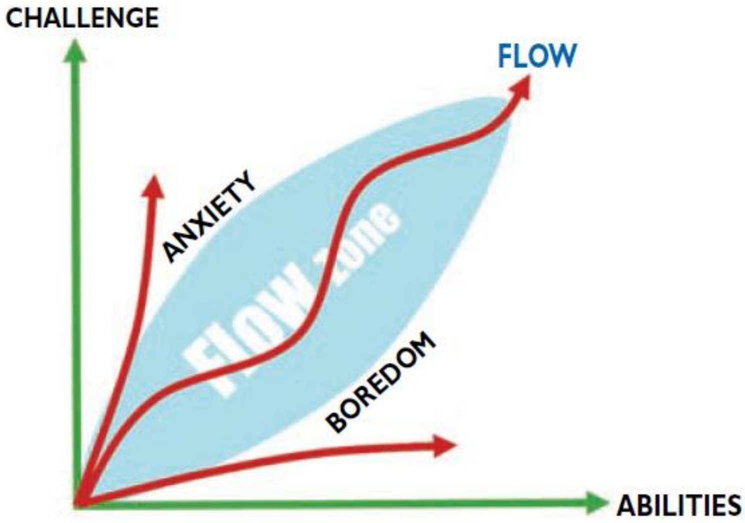


Figure 2-6 Flow Concept (Chen, 2007)

Other authors use the concept of flow in conjunction with other elements to evaluate player enjoyment (Sweetser & Wyeth, 2005). The problem with this model is that results may vary from one game genre to another. Some elements of the model may be relevant in some genres and less relevant in others.

2.2.1.5 Game Design Documentation

The game documentation mainly occurs in pre-production, which is reflected in the game design document, and the game designer is primarily responsible for the documentation. The GDD can be considered the requirements document, since it is where most of the specifications of the game assets are obtained. The game designer should be concerned not only with the game features, but with the experience that the game will bring to the player. This experience arises from the interaction between the player and the game. Game design documentation will be covered in more detail in Section 2.2.2 and 2.2.3, and player experience will be covered in Section 2.3.

2.2.2 Requirements Engineering and Game Design

In this section we talk about work on requirements engineering and its relation with game design. First, we present information on the growth in research in requirements specification in game development. Second, we talk about problems found when moving from pre-production to production from a requirements engineering perspective. Third, we analyze a proposal to incorporate player experience to quality models by characterizing it in to playability. Finally we talk about research opportunities found in the game development and requirements engineering area.

The software engineering research focused on game development has increased in recent years, has been the requirement engineering on which most research has focused (Ampatzoglou & Stamelos, 2010) as shown on the previous chapter in Table 1-1 and 1-2.

The Callele et al. proposal does an analysis of the creative process in games and the main challenges in requirements engineering (Callele et al., 2005). In Callele's work they point out the difficulties of moving from pre-production to production because the GDD fails to meet the formality, detail and an adequate structure that a Software Requirements Specification (SRS) document needs to be useful to the software engineers that create the software on production. Figure 2-7 illustrates this problem. Callele points out the need for a method that helps to solve this problem without hindering the creativity of the game designer. In the work the authors does not propose a solution, they just the problem and the need of research in the subject.

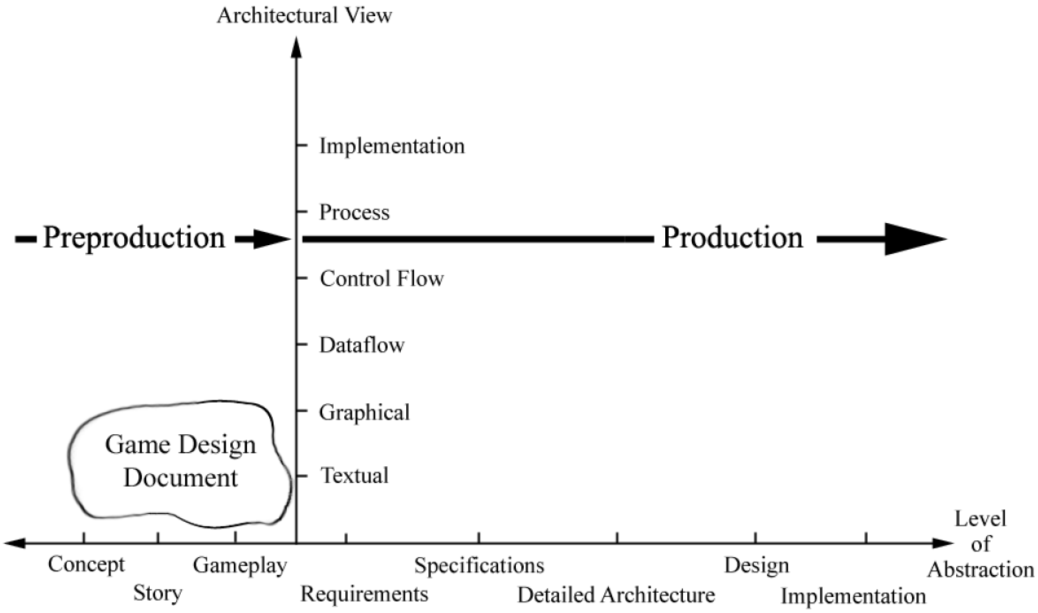


Figure 2-7Game Development (Callele et al., 2005)

In their work on quality attributes on game development (J. L. González et al., 2009) propose the incorporation of playability as an attribute of quality in use in the standard ISO/IEC 25010-3 (“ISO/IEC 25010-3: Systems and software engineering: Software product quality and system quality in use models.,” 2009), since most of the standards and models that handle quality do not consider the ludic nature that a game brings to player experience. The model representation is shown in Figure 2-8.

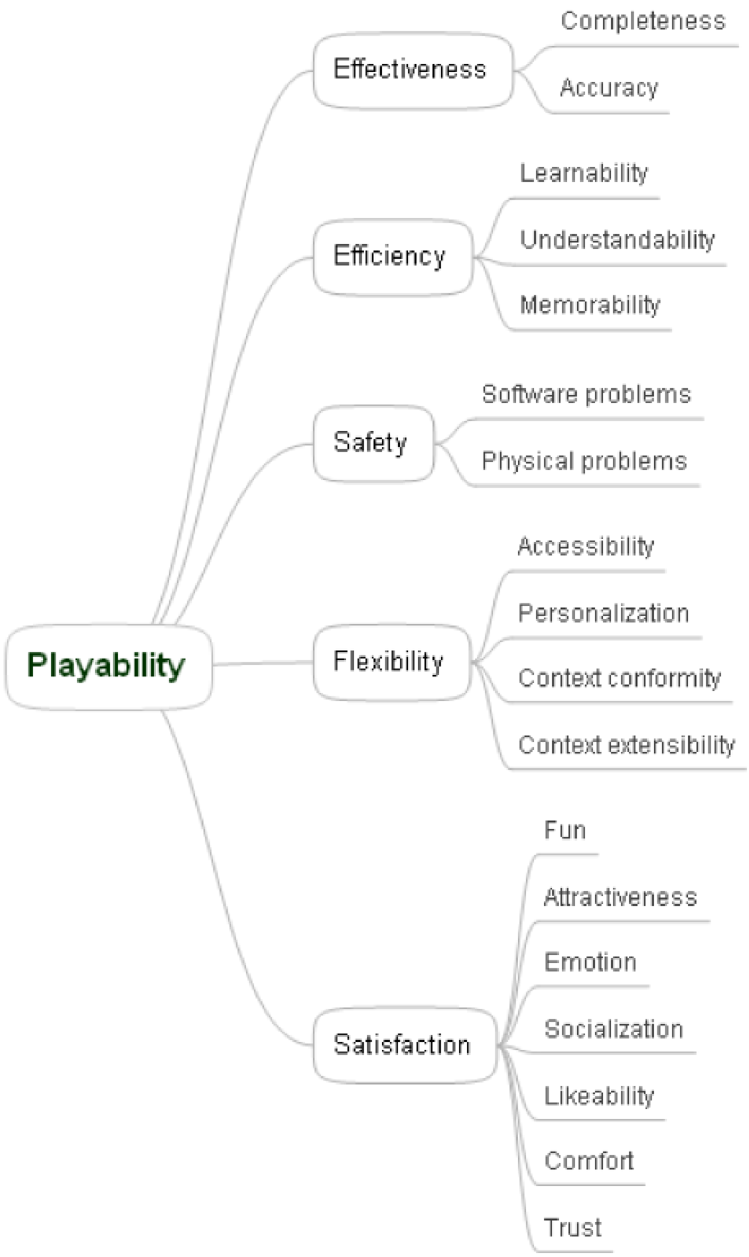


Figure 2-8 Playability Quality Model

The requirements engineering on game development have some promising research topics, as pointed out by (Callele et al., 2011). The problem that Callele et al. discussed in their previews article (Callele et al., 2005) still persist and is pointed as a research opportunity. The research opportunities presented in this work are the following:

- Process management
- Requirements engineering in games
- Experience requirements

- Experience requirements and interactions with other requirement types user experience design, game design
- Film and other creative / media productions
- Value and economic analysis
- Language and ontology

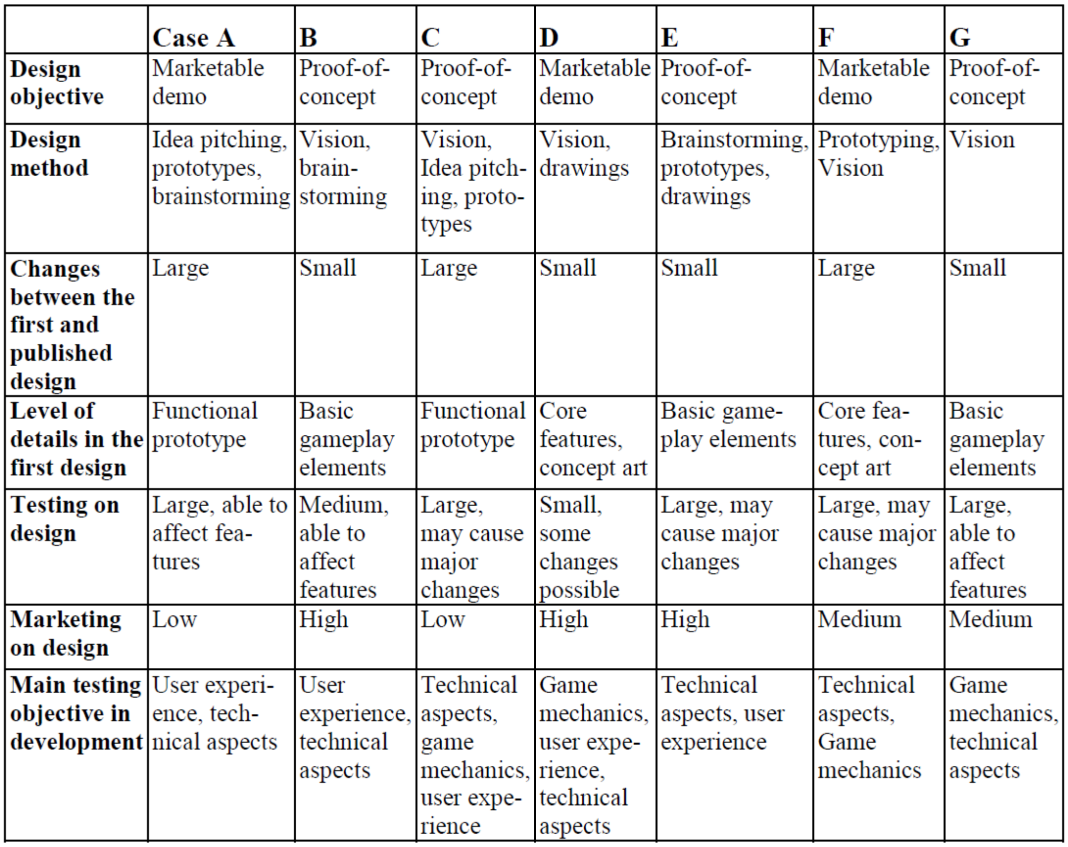
Guo et al. propose the Game Creation with Customized Tools (GCCT) based on the Model Driven Development (Guo, Gao, Krogstie, & Trætteberg, 2015). GCCT have four main tasks: the first three corresponding to the tools customization: game feature customization, game editor customization and game code generator customization; the last task is the Game creation. Each of these tasks is mapped to a MDD task. The author don´t explain in detail how these tasks adapt to create a game and there is no case of study or example to understand how the GCCT works. The evaluation of GCCT is based on a questionnaire to a group of people that after seen a video explaining how the GCCT works, they answer the questionnaire about their opinion on the usefulness of the GCCT. The authors' proposal may be useful, but is hard to determinate whit out a case of study or an experiment that gives data on the use of the GCCT to create games.

Kasurinen, Maglyas and Smolander conduct an investigation based on interview with game development professionals, their goals was to find out if Requirements Engineering (RE) was useful in game development and how industry was applying these SE practices. They interview companies with different maturity, size, and target platforms. They analyze the date and made the following findings:

- Game developers need to manage plans and product requirements, as the product may vary greatly between the first design and release.
- Game products can be changed significantly based on the feedback from marketing and testing.
- Requirement analysis is conducted mostly with user tests and usability testing.
- Game developers try to minimize the amount of functional requirements that should be implemented.

In addition to these main findings the authors discover seven main categories shown in Table 2-3 where the RE have relevance and concluded that using RE practices does not hinder creativity, but these practices should be used after the game concept has been defined because when the game idea hasn´t been conceptualized it can be a lot of changes. The proposal presented on this work has influence on all the seven categories and is guided by the fun (Player Experience).

Table 2-3 Requirements engineering relevance in game development categories (Kasurinen, Maglyas, & Smolander, 2014)



	Case A	B	C	D	E	F	G
Design objective	Marketable demo	Proof-of-concept	Proof-of-concept	Marketable demo	Proof-of-concept	Marketable demo	Proof-of-concept
Design method	Idea pitching, prototypes, brainstorming	Vision, brainstorming	Vision, Idea pitching, prototypes	Vision, drawings	Brainstorming, prototypes, drawings	Prototyping, Vision	Vision
Changes between the first and published design	Large	Small	Large	Small	Small	Large	Small
Level of details in the first design	Functional prototype	Basic gameplay elements	Functional prototype	Core features, concept art	Basic gameplay elements	Core features, concept art	Basic gameplay elements
Testing on design	Large, able to affect features	Medium, able to affect features	Large, may cause major changes	Small, some changes possible	Large, may cause major changes	Large, may cause major changes	Large, able to affect features
Marketing on design	Low	High	Low	High	High	Medium	Medium
Main testing objective in development	User experience, technical aspects	User experience, technical aspects	Technical aspects, game mechanics, user experience	Game mechanics, user experience, technical aspects	Technical aspects, user experience	Technical aspects, Game mechanics	Game mechanics, technical aspects

Washburn et al. analyze post-mortem of published games where the developers put what went right and what went wrong while creating the game (Washburn Jr., Sathiyanarayanan, Nagappan, Zimmermann, & Bird, 2016). They divided the information in four main categories: product, development, resources and customer facing, from these categories they create sub-categories and reviewed the post-mortems. Figure 2-9 show the percentage of incidence on what went right and Figure 2-10 show the same for what wrong.

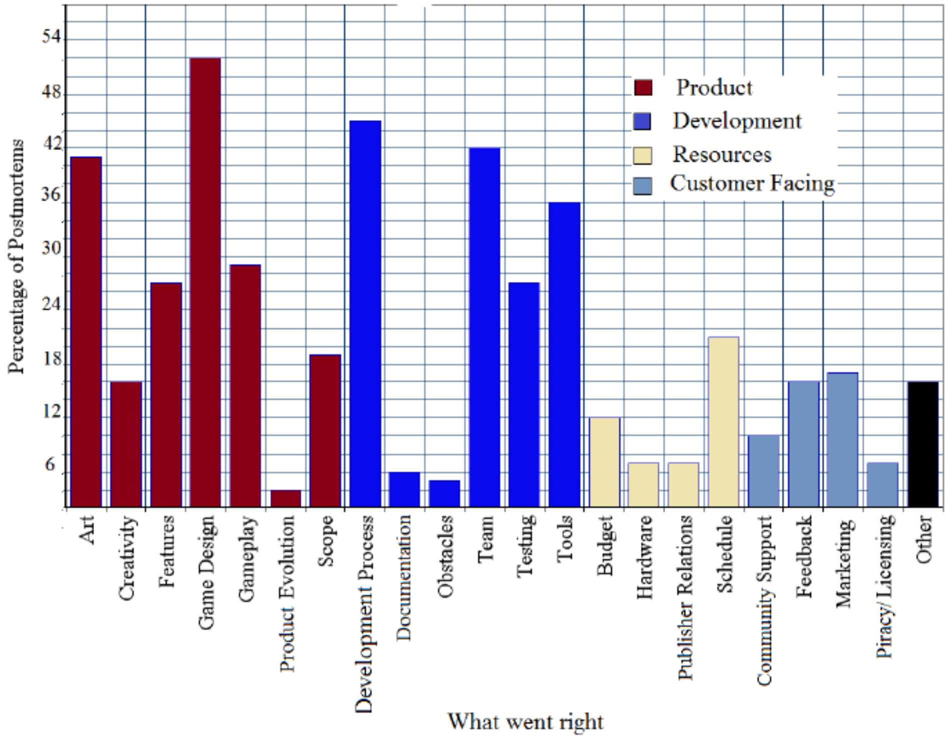


Figure 2-9 What went tight on post-mortem (Washburn Jr. et al., 2016)

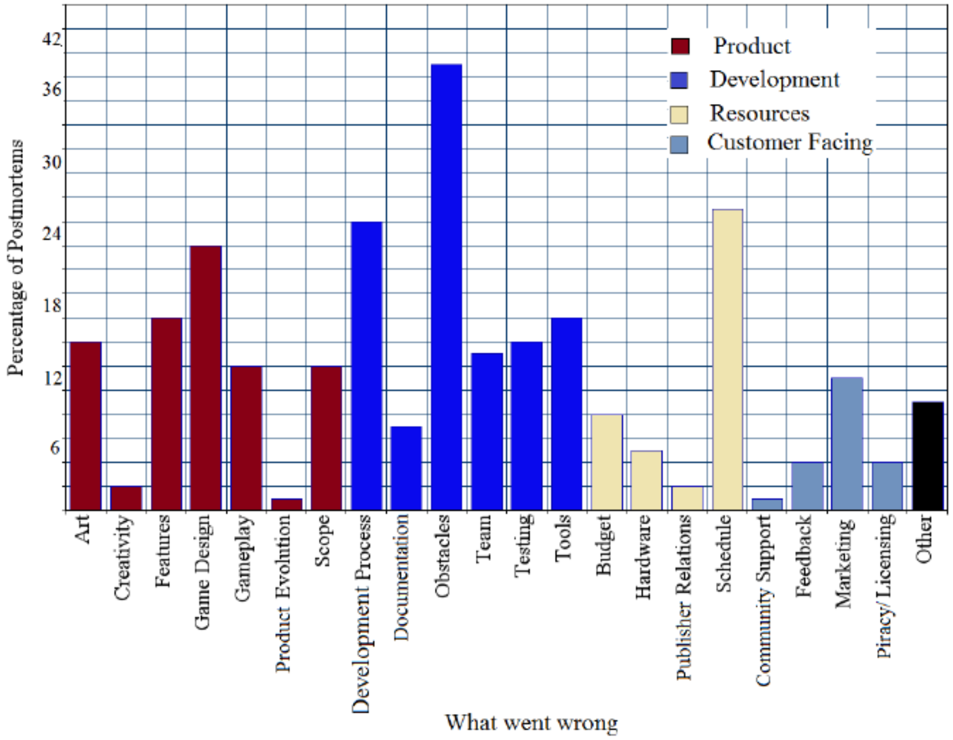


Figure 2-10 What went wrong on post-mortem (Washburn Jr. et al., 2016)

Additionally they compare different criteria to see if there is a difference or coincidence in the rights and wrongs: small teams (twenty or less) and large teams, publisher and no publisher, and multiplatform and single platform. On the small versus large team criteria they find out that small teams tends to encounter more unexpected obstacles than large teams. On the publisher versus no publisher they find out that teams that publish their own games may be affected more often by unexpected obstacles. On the multi versus single platform they find out that game with multiples platforms tends to have better marketing. In this work proposal we contribute to the most frequent sub-category that went wrong on product: game design and to the second and third most frequent sub-categories that went wrong on development: development process and tools.

The works previously analyzed presents the main problems to do requirements engineering in game development, proposed some solution to these problems and opens new research opportunities, to solve this problems. These opportunities especially focus on requirements communication between pre-production and production and handling the player experience.

2.2.3 Game Design Documentation

This section is about game design and how it is documented. First, work on different approaches to the structure suggested for a game design is presented. Second, proposals and advice given by different authors about the GDD content and structure is presented. Finally, the conclusions on the game design documentation problems and opportunities are discussed.

Robin Hunicke et al.'s work on Mechanics, Dynamics and Aesthetics (MDA) framework (Hunicke et al., 2004) presents an iterative approach to design and tuning video games. The framework has three levels of abstraction. It starts by defining the aesthetics of the game, which are similar to the player's experience defined in this dissertation. Then, the intended game interaction is defined in dynamics trying to fulfill the aesthetics. Finally, the game elements that can bring the interaction are created. Figure 2-9 shows the three level, its base concept and how close the relation is with developers and players.

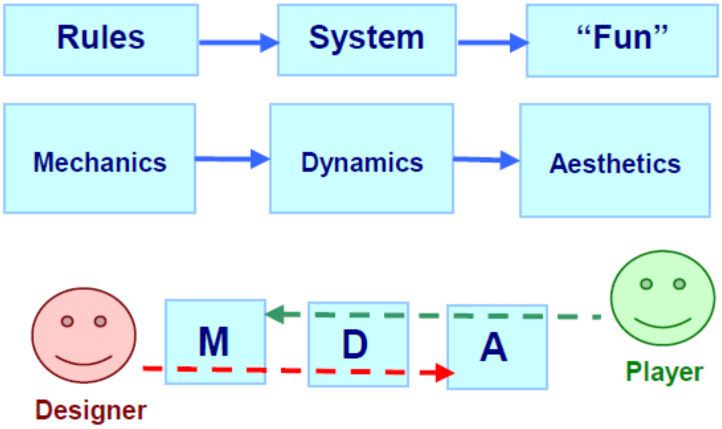


Figure 2-11 MDA Concept (Hunicke et al., 2004)

We believe that the use of aesthetics to refer to player's experience may be confusing, since aesthetics is usually used to describe the look and feel of the game and it does not necessarily cover issues like challenges. The framework offers an interesting way to design and tune games, but does not provide the tools or methods to construct the details of such mechanics, dynamics and aesthetics. Dynamics and mechanics may require more information not only in its logic but in how they are related to the aesthetic part, such as how they should look, and how they should sound.

Several authors talk about the GDD as the place where the game design is documented, but even if all agree in general, the details in the document content may differ or may not be detailed from author to author.

Andrew Rollings and Ernest Adams mentioned three different documents to document the design of the game(Rollings & Adams, 2003). The first is called high concept and contains the main features of the game; it aims to sell the idea of the game. The second document is called game treatment and has a greater level of content which aims to give more detail to someone who is already interested in the game and wants to know more about it. The last document is called the script game, and is what we call in this work a GDD, which is the document that should specify extensively the detailed game, and which will serve as a reference for creating the game. They do not provide an example of a GDD, but make reference to Taylor's template (Taylor, 1999), which is one of the GDD templates analyzed later on chapter 3.

Chris Crawford (Crawford, 2003) presents a series of lessons learned throughout his career as a game designer. While these lessons are based on a long career and the lessons listed are useful, there is no significant contribution on how to document the game design. The examples are simple and they lack detail, and he does not provide an example or template of a GDD. It is hard to document the design of a game by taking his work as a guide.

Bob Bates (Bates, 2004) gives an introduction to game design, then describes game design from the perspective of the different roles, then describes the process and the documentation of the game design, finally analyzing game design from a business viewpoint. Bates describes game design documentation in a similar way as Rollings and Adams (Rollings & Adams, 2003), that is, different documents with different purposes, but they agree in that the document that contains the details of the game design is the GDD. Bates uses game genres to explain different styles in game design, he gives a template of a GDD for an action genre so it can be used to document an action game or be adapted to document other genres. He recognizes the GDD as a vital asset in game development and suggests creating it online in a similar fashion as a wiki. This can help to keep the document updated easier than a static document. Bates gives no examples of a GDD and his template is tailored to a specific genre. Nowadays, it is hard to classify the games by genre due to the lack of standardization and the new genres and multi-genre games.

Erik Bethke in his work (Bethke, 2002) gives an extensive and detailed picture of game development and production. Bethke's work is full of real life examples used to illustrate his ideas. On game design, Bethke describes what a GDD should contain, the process of creating the GDD and the amount of work that each stage of the process may require. Bethke does not provide a GDD template but following the description of the content of the GDD that he gives a template may be elaborated. He covers the structure of the GDD, but he does not go too deeply into the detail of each part, and how each part is related.

Richard Rouse III (Rouse, 2004) covers the main topics about game design, offering his theory of concepts and complements this theory with his own personal experience applied to known games and with interviews with experienced game designers. He provides two samples of different GDDs, but he does not provide a template and both examples are difficult to relate to in terms of a basic structure and how they elements are related.

Kevin Oxland's work (Oxland, 2004) gives one of the most detailed descriptions on how to document game design. He uses a game as a sample and as he covers different subjects in game design he illustrates how his sample game will look with reference to that subject. He does not provide a

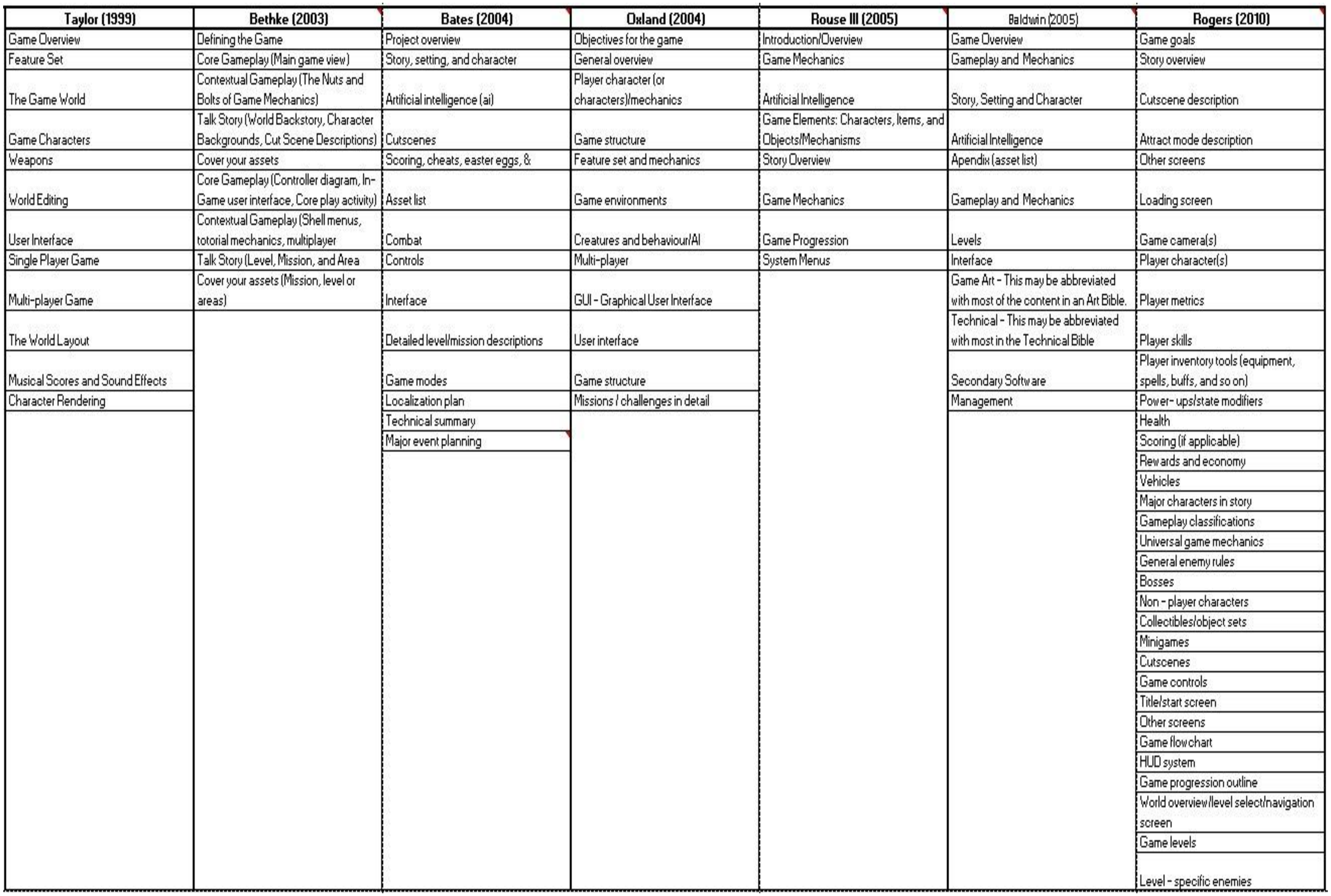
GDD template or a full example of a GDD. Since he does not provide a GDD, it is hard to know the structure or how the elements are related. But his work can be used as a reference on how to go in deep when documenting game design.

Scott Roger (Rogers, 2010) concurs with other authors on the progression of the game design documentation, commencing by shaping the game idea on one sheet, then detailing it into ten pages and finally creating the GDD. He uses drawings to clarify his concepts. He proposes a chart based on levels to give structure to the game in the GDD. He provides a GDD template but it is too specific to games with particular characteristics, which leaves out many types of games.

Jesse Schell (Schell, 2008) maps the elements that interact in the process of playing a game. He describes progressively all the relevant elements and then he talks of ways he had found to address these elements. The main ideas of each topic are described in lenses, which can be used as guides when designing. He does not provide a GDD template or sample. His work does not gives detailed examples of game design, it focuses more on explaining how games and game design work than on how to document the game design.

Table 2-3 show the comparison between different authors GDD proposal, the section each author proposes vary widely from one author to other.

Table 2-4 GDD Proposals Comparison



Taylor (1999)	Bethke (2003)	Bates (2004)	Oxland (2004)	Rouse III (2005)	Baldwin (2005)	Rogers (2010)
Game Overview	Defining the Game	Project overview	Objectives for the game	Introduction/Overview	Game Overview	Game goals
Feature Set	Core Gameplay (Main game view)	Story, setting, and character	General overview	Game Mechanics	Gameplay and Mechanics	Story overview
The Game World	Contextual Gameplay (The Nuts and Bolts of Game Mechanics)	Artificial intelligence (ai)	Player character (or characters)/mechanics	Artificial Intelligence	Story, Setting and Character	Cutscene description
Game Characters	Talk Story (World Backstory, Character Backgrounds, Cut Scene Descriptions)	Cutscenes	Game structure	Game Elements: Characters, Items, and Objects/Mechanisms	Artificial Intelligence	Attract mode description
Weapons	Cover your assets	Scoring, cheats, easter eggs, &	Feature set and mechanics	Story Overview	Apendix (asset list)	Other screens
World Editing	Core Gameplay (Controller diagram, In-Game user interface, Core play activity)	Asset list	Game environments	Game Mechanics	Gameplay and Mechanics	Loading screen
User Interface	Contextual Gameplay (Shell menus, totorial mechanics, multiplayer	Combat	Creatures and behaviour/AI	Game Progression	Levels	Game camera(s)
Single Player Game	Talk Story (Level, Mission, and Area	Controls	Multi-player	System Menus	Interface	Player character(s)
Multi-player Game	Cover your assets (Mission, level or areas)	Interface	GUI - Graphical User Interface		Game Art - This may be abbreviated with most of the content in an Art Bible.	Player metrics
The World Layout		Detailed level/mission descriptions	User interface		Technical - This may be abbreviated with most in the Technical Bible	Player skills
Musical Scores and Sound Effects		Game modes	Game structure		Secondary Software	Player inventory tools (equipment, spells, buffs, and so on)
Character Rendering		Localization plan	Missions / challenges in detail		Management	Power- ups/state modifiers
		Technical summary				Health
		Major event planning				Scoring (if applicable)
						Rewards and economy
						Vehicles
						Major characters in story
						Gameplay classifications
						Universal game mechanics
						General enemy rules
						Bosses
						Non - player characters
						Collectibles/object sets
						Minigames
						Cutscenes
						Game controls
						Title/start screen
						Other screens
						Game flow chart
						HUD system
						Game progression outline
						World overview/level select/navigation screen
						Game levels
						Level - specific enemies

Is concluded that no author covers the three main problems on game design documentation identified in (Callele et al., 2005): formalism in detail, structure and relations. Chapter 3 covers more in depth the relations between formalism, structure and relations and our proposal.

Zook and Riedl propose the Mechanics generation as a tool to create automatic game design (Zook & Riedl, 2014). The authors use a constraints solver to generate mechanics given specific requirements and a planner to evaluate if the generated mechanics meets specific playability goals. They use specific domain to help with the mechanics definition but give the freedom to give non domain playability and design requirements. The authors test their proposal with Role Playing Game (RPG) domain and platform domain and then they combine bot domains. The proposal may prove useful to generate challenges for specific type of game like infinite runners or specific mechanics of game like repetitive encounters on an RPG, but replacing all the game challenges may restrict and hinder creativity and disrupt the synergy that result of designing challenges to meet other goals than just present a difficult to the player.

Orita and Correa present a systematic review of game design methods and tools (Orita-Almeida & Correa-da-Silva, 2013)they classify the analyzed work into three main categories: a shared design vocabulary, game design methods and tool, and a design visual language. The authors identify problems that are still present regarding game design, like the lack of formality, flexibility in the design document, tools too general to be useful or too specific that hinders creativity. They identify to main users of these game design methods and tools. The first one is specialized team that want to improve productivity and quality this kinds of users may require tools and methods a more precise and formal approach sacrificing flexibility and creativity; The second one is small and emerging teams that need a guide on how to create the game design this team require tools and methods less formal an with a flexible approach that can be shaped on grow with the team. The proposal on this work is address to the second type of users. The authors conclude that there is a broad area in game design to contribute and that the academia and industry coincide in the needs but there is still a long road to understand how to cover these needs.

2.3 Player Experiences in Video Games

2.3.1 Overview

Player experience is a relevant concept in video games. Omitting, or delaying to final stages the handling of player experience may bring unexpected and undesired results such as unfulfilled goals and features unaligned with the desired experience. In this section current work on measuring, handling and tracing player experience is analyzed.

2.3.2 Proposals

Emily Brow's work (E. Brown, 2010) is a snapshot of the current method which the game industry uses to handle the player experience in the development process. She talks about what is actually used which is mainly based on the game designer experience. She concludes that there are big areas of opportunity to improve these practices, by creating new tools that can be used in a practical way in the game development for handling the player experience.

José Luis González et al. (J. L. González et al., 2009) characterize the player experience as playability and divide playability into a number of attributes that can be measured. The work proposes to incorporate playability as an extension of quality in use. Each attribute has associated several metrics, most of which can be obtained automatically inside the game (e.g. percentage completed of the world); this means that most attributes cannot be measured until the game is finished.

Katherine Isbister's work (Isbister, 2010) is a summary of work on social play based on reported best practice and previous publications. It offers suggestions for enabling and measuring social play, but it is still in the early stages and there is no specific method or methodology to follow.

Jari Takatalo et al.'s work (Takatalo et al., 2010) is based on the Experimental Virtual Environment Experience Questionnaire-Game EVEQ-GP. The game needs to be in the semifinal stage to be able to apply the questionnaire. Presence, involvement and flow are represented in latent variables and associated with 34 questionnaire items to be measured. These questionnaires are related to 15 factors of user experience in video games. The questionnaire items are the key to measuring the user experience in video games and they need at least a prototype to be able to apply these questionnaires.

Eduardo Calvillo et al.'s (Calvillo-Gámez, Cairns, & Cox, 2010) describes the core elements that a game must have in order for the player to have a positive experience, they proposes the Core Elements of the Gaming Experience (CEGE) to evaluate if the game have the core elements or not. The authors claim that by having these elements there is no guarantee that the experience will be positive but it won't be negative; and furthermore, that the lack of these elements will result in a negative experience. They use a questionnaire to evaluate the experience. The core elements are described but there is no explanation as to how to implement them in the game. The identified core elements can be used on early stages to guide the design.

P. Lemay & M. Maheux's (Lemay & Maheux-Lessard, 2010) proposes a questionnaire using semantics to measure how leisure activities are perceived, one of them is play video games. They compare the activity of playing video games with different profiles, like players versus non-players or men versus woman. This work proves useful in comparing different pairs of semantics to see how a specific game or genre is perceived, but does not address the specific experience generated by the game for the player.

H. Desurvire & C. Wiberg's work (Desurvire & Wiberg, 2010) proposes the Game Approachability Principles (GAP) that is a set of heuristics to create better tutorials, and by this, improving the player experience. It is true that teaching the players how to play the game is important, but is just a part of the many other topics relevant to player experience. It is important to point out that tutorials are just one of many tools available to game designers that can be used to teach the player about the game.

K. Poels et al. (Poels, IJsselsteijn, Kort, & Iersel, 2010) analyze the post-game experience in video games. They discovered several experiences that the players have after playing a game in short or long terms. The experiences categories found are: perceptions, emotions, cognitions, and behavior. This work is an initial approach to exploring the experiences that the players have after they play a game and how they affect their lives and dispositions to other games.

M. Lankes et al. (Lankes, Bernhaupt, & Tscheligi, 2010) study how emotions of characters in a game relate to the experience of the player. They found out that more than just having detailed and realistic emotions for characters, these emotions should relate to the context of the game. When the emotions of characters and context of the game are consistent, the player experience is positive, but when the character emotions and the context of the game are not consistent the player experience is lower or even negative. As a result of this study, it has been determined that the main focus when trying to express emotions in characters and transmitting them to the player should be emotions consistent with the context. This study applies mainly to games that can reflect emotions on characters' faces, which left the less detailed games out.

F. Mueller & N. Bianchi (Mueller & Bianchi-Berthouze, 2010) evaluate user experience in the new exertion games (how the combination of exerting bodily movements with computer gaming affects the user experience). They use different approaches to measure the user experience and find the variation in results in exertion games compared to traditional games, but emphasize the methods that must be applied immediately after the play occurs, since the player may be exhausted and in an altered emotional state. The authors discovered that the exertion games offer new ways of motivation mainly related to the health of the player. The work presented here is just the first approach to research of the main differences of exertions games player experience.

Brown et al. (M. Brown, Kehoe, Kirakowski, & Pitt, 2010) study the relation between user experience and game controllers. They used three different controllers to evaluate the player experience in terms of usability and functionality in an experiment. The study concludes that is important to take into consideration how the controllers will affect the player experience from the beginning of the game development, and evaluate it in short term (the period that the player adapts to the controllers) and long term (once the player is used to the controllers). This study shows evidence of the relationship of user experience and game controllers, but not details on how to establish the best controllers depending on the type of game.

Koeffel et al. (Koeffel et al., 2010) propose a framework of heuristics to evaluate user experience. They use existing heuristics as basis to propose their framework, and extend it so it can evaluate tabletop games too. The main reason for using heuristics as the main tool to achieve and evaluate user experience is that they are cost effective and can be apply in early stages in game development. They saw that games that follow heuristics bring better experiences than games that do not. The use of heuristics has the problem of balance between being too generic to be useful versus being too specific to be widely applicable.

Mirza et al. (Mirza-Babaei, Nacke, Gregory, Collins, & Fitzpatrick, 2013) study the benefits of biometrics-based user test against non-biometrics user test on game design. They ascertained that user test feedback significantly improves the game design and player experience, and that biometrics feedback may be translated into changes that improve game design and player experience over those of the non-biometrical user test. The biometrical user test offers a more accurate feedback when evaluating and shaping player experience, but the technology needed to create these tests may not be available to all game developers.

Elson, Breuer and Quandt propose the integrated model of player experience (IMP) framework (Elson, Breuer, & Quandt, 2014). It take into account the phases: pre use (choice), use (play), and post use (effects); personal (Player traits and states); media (game characteristics); and contextual (settings and social environment). The IMP comprises the gaming experience in three main elements: context, player and game. These elements can be analyzed in three main phases: pre-game, game, post-game. The authors explain the each elements and how is characterized in each phase. For example how context influence the player experience before playing the game if the country laws has low tolerance to violent games, or the context while playing the game with a poor internet service. By dimensioning the player experience into these three elements and phases, researchers studying player experience can classified and guided based on which elements and phases they impact. Methods and models related to player experience can be classified too into these three elements and phases this can help the researchers to selects the method and models based on their influence areas. In this work the proposal is related to the three main elements and phases but is mainly focus on the game elements in the pre-game and game phases.

Cairns, Cox and Nordin work analyze the work involving immersion as part of the player experience (Cairns, Cox, & Nordin, 2014). They conduct an experiment and find out that immersion is influenced by sensorial factors like music or by game mechanics factors like time limit, on the contrary other factors like been a 2-Dimensiona or 3-Dimensional game does not affect the immersion. Based on their research they do an attempt to describe the immersion a follows:

“Our best current understanding is that it is a confluence of different psychological faculties such as attention, planning and perception that when unified in a game lead to focused state of mind. In this state, players are less aware of the world around them and become immersed in the game. Moreover, this is a self-sustaining state because of the pleasures associated with being immersed in a game.”

Immersion is an important aspect if the player experience and is influenced by multiple factors, but there is work to be done to understand how it relate to other concepts like engagement or which attributes are relevant for immersion depending on the type of game. In a RPG narrative may be key achieve the desired immersion, but in games Tetris™ the mechanics may be more relevant to immersion. The importance of immersion may vary from one kind of game to another and the desired experience that each game want to evoke.

Caroux et al. did a systematic review about the player-video game interaction (Caroux, Isbister, Le Bigot, & Vibert, 2015). To classify their work they use two main areas: the player aspect and the video game aspect, then they create categories and sub-categories in each area Figure 2-12 show these categories and sub categories.

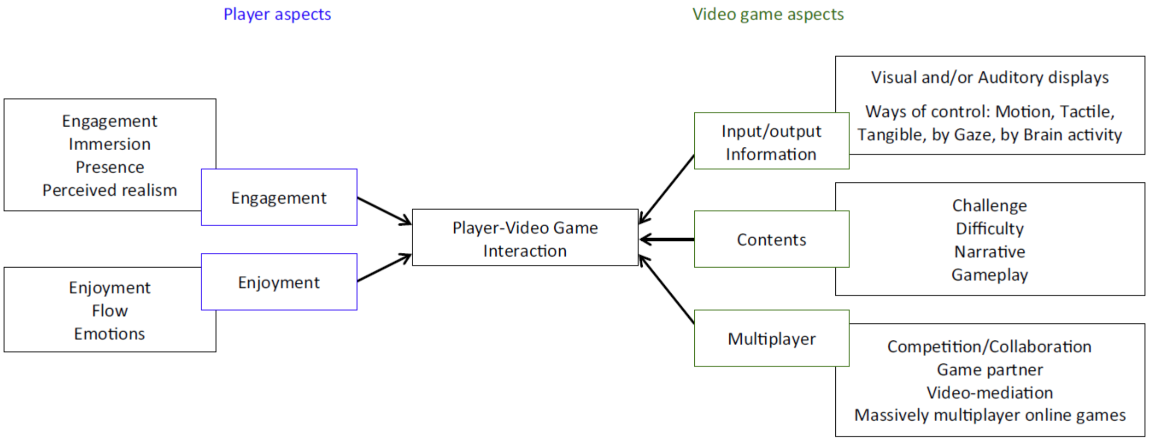


Figure 2-12 Player-video game interaction aspects (Caroux et al., 2015)

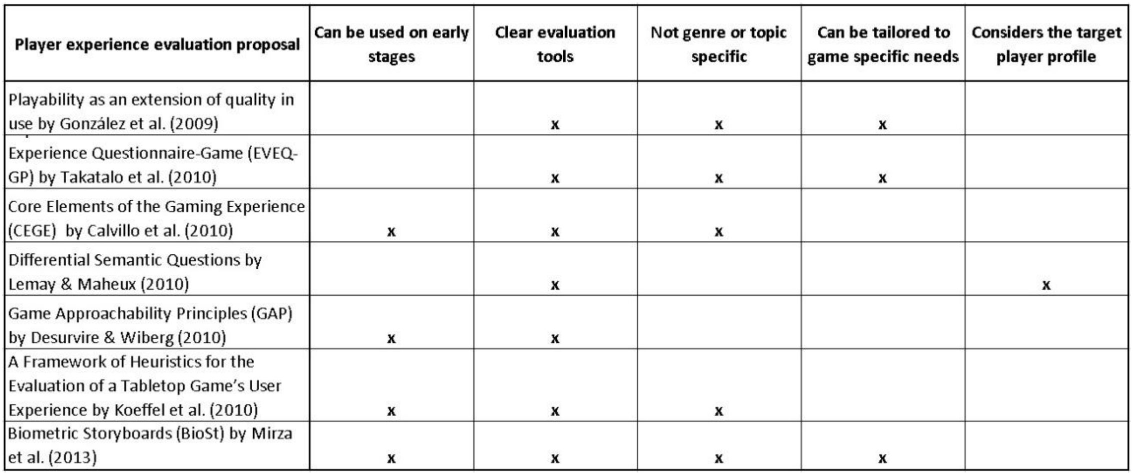
The authors create a definition for the Player-Video Game Interaction based on the results of the review:

“Player–video game interactions are interactions in which technical aspects of video games have influence on players’ engagement and enjoyment”

The authors also find that also find that interest for the study of player-video game interaction is recent, most of the studies that they analyze are from 2010 or early. They find two main problems with the current research: the first is the limitations of validity, this because in many studies they use questionnaires as evaluation tool; the second is the impact of the study, where the studies are aimed to a small area, like collaborative games, or racing games. They suggest using objective tools to evaluate the proposals especially in the player aspect with tools like eye tracking or physiological data, these tools can be an optimal complement to the questionnaires. The proposal presented in this work has an impact on both aspects player and video game and use subjective and objective tools to validate the results.

This section presented an analysis of different studies and proposal related to player experience or user experience in games. Table 2-4 present a comparison between the more mature methods analyzed to understand better its context of use.

Table 2-5 Player Experience Proposals Comparison



Player experience evaluation proposal	Can be used on early stages	Clear evaluation tools	Not genre or topic specific	Can be tailored to game specific needs	Considers the target player profile
Playability as an extension of quality in use by González et al. (2009)		x	x	x	
Experience Questionnaire-Game (EVEQ-GP) by Takatalo et al. (2010)		x	x	x	
Core Elements of the Gaming Experience (CEGE) by Calvillo et al. (2010)	x	x	x		
Differential Semantic Questions by Lemay & Maheux (2010)		x			x
Game Approachability Principles (GAP) by Desurvire & Wiberg (2010)	x	x			
A Framework of Heuristics for the Evaluation of a Tabletop Game's User Experience by Koeffel et al. (2010)	x	x	x		
Biometric Storyboards (BioSt) by Mirza et al. (2013)	x	x	x	x	

2.3.3 Player Experience Proposals Summary

We present several proposals and studies related to player experience. These proposals vary widely. We analyze each proposal's benefits and flaws, considering the following concerns: How can player experience be handled in early stages and traced to later stages in game development? How much effort is needed to handle player's experience in a game? How to define the intended player experience and trace it to the game elements? How will the desired player experience help to achieve the game goals? Chapter four will try to answer these questions.

2.4 Game Development Methodologies

This section analyzes the current game development methodologies and how they are applied. First we talk about traditional software and game development, second we talk about agile game development and finally we talk about patterns in game development.

2.4.1 Game Development Models

In this section we talk about traditional software development models and how they are related to game development.

J. Kasurinen, R. Laine & K. Smolander's work (Kasurinen, Laine, & Smolander, 2013) analyses the ISO/IEC 29110, Lifecycle profiles for Very Small Entities, and how applicable it is in the game development industry. Their study is based on questionnaires applied to seven game industry companies that fall in the very small category, and based on the results of the questionnaires see the suitability of the model in game development. The results show that the model is hard to apply as it currently stands. They identify some key issues that need to be modified, so the model can be used in game development: first, the game design needs to be open to modifications in the whole process; second, the model needs to support iterative development. In the study it is clear that small companies are more comfortable with an agile development approach. According to the results, six of the seven companies analyzed are using an agile development approach and three of them are using Scrum. At the end, they propose a model that covers some of the issues identified in the study, but the model needs to be tested.

E. Bethke's (Bethke, 2002) does an extensive cover of game development. He defines roles, key game development elements and how the roles create the game development elements. He mentions some software development models, like waterfall (Royce, 1987), iterative (Benington, 1956) and extreme(Beck, 1999). However, he does not suggest any software models, he proposes analyzing the project and selecting the best model to meet the project goals. This method is logical, but it may be difficult to follow, trying to change the game development paradigm of a whole team, because surely this will affect the learning curve.

A. Rollings and D. Morris (Rollings & Morris, 2003) analyze in detail game design and discuss software development models and their application to game development. They point out the main problem that a waterfall model may bring, which is the lack of flexibility. So the use of a more flexible model like the spiral model (Boehm, 1988) may help to solve this problem. They suggest the use of the software factory concepts to reuse many of the assets created. They recognize that a game may be too different from another to reuse the code, but point out that even if many parts of the game may be different there are many other that are not, like the menus or sound and music handling.

B. Bates' work (Bates, 2004) talks about the development models. He analyzes the waterfall model, the modified waterfall model that overlap activities and the iterative prototyping model. Bates concurs with most of the authors that the waterfall model has too many flexibility problems to be used on game development, and suggests that the modified waterfall where the activities overlap is better suited. However he points out that using the iterative prototyping model achieves the best results when doing game development. This is because the prototype gives a continues feedback, so the game design can be refined and go for the "find the fun first" concept, that states that the first goal in pre-production is to find, test and tune what makes the game fun.

J. Schell (Schell, 2008) discusses the waterfall model and the problems that it brings due to its lack of flexibility. He points out that in game development these problems tend to get bigger. He supports the spiral model as an alternative to the waterfall model, which is based on iterative cycles, each cycle focuses on mitigating the bigger risk to the project, which means each cycle will bring more certitude to the project.

Kolrva et al. presents a study on how resources are spent and workflow occurs in reality in game development (Koleva, Tolmie, Brundell, Benford, & Rennick-Egglestone, 2015). They conduct a study based on questionnaire and ethnographic practice on game development companies. The results show that tools are used across activities and stages, this is somehow expected because in game development a change of stage does not imply that an activity is finished, a game designer continue the design on production and a programmer can start programming in pre-production to create a functional prototype. Another finding in the study was that the different roles in game development have a poor interaction level with their clients, this results can be explained by understanding that in many cases there is no client, the producer may by the closest role to a client, potential players can be identified as clients to, but most of the time they are used to give feedback on the game in the final stages. As a final conclusion the authors writes:

“In particular there are a number of ways in which existing workflows are not yet well supported by tool design. The next important step is to see how the developers of such tools respond to these requirements”

The proposal in this work provides a workflow support problem approach.

In summary, software development models have been adapted to game development, and currently the experience shows that the waterfall may bring the same problems that software development has and create new ones. Based on the reviewed work, the modified waterfall or the iterative models are better approaches to game development. Modified waterfall models have the advantage of being able to overlap activities, which adds flexibility and a fast feedback, however iterative models like spiral or Scrum have the advantage of observing important parts of the final product in early stages, and are more flexible than modified waterfall and bring accurate feedback to the player. In iterative models it is easier to apply the concept "find the fun first".

2.4.2 Agile Game development

This section discusses the agile paradigm and how it has been integrated with game development. Erickson et al. (Erickson J., Lyytinen, & Siau, 2005) defined the agile software development as "strip away as much of the heaviness, commonly associated with the traditional software-development methodologies, as possible to promote quick response to changing environments, changes in user requirements, accelerated project deadlines and the like".

Agile development emerges as a solution to common problems (mentioned in Section 2.4.1) in software development. The guiding concepts of the agile paradigm are noted in the agile manifesto (Fowler & Highsmith, 2001). In game industry the use of agile framework or methodologies like Scrum (Schwaber & Beedle, 2002) is growing (Kasurinen et al., 2013; Keith, 2010) as agile development seems better suited to the unique challenges of game development.

P. Abrahamsson et al. (Abrahamsson, Oza, & Siponen, 2010) compare the agile development methods over the last ten years. They conclude that current agile methods have some issues that require the attention of the scientific community, these issues are: clarifying their range of applicability and explaining the interfaces to the software development life-cycle not covered by the methods; producing more detail on project management; offering concrete guidance to support their solutions; efforts on creating ways the method can be adapted in different development situations.

C. Keith's (Keith, 2010) is about agile game development with Scrum, wherein he first explains the Scrum concepts and then explains how they can be used in game development. He gives a detailed description as to how each concept can be applied to the Scrum framework and how it can be complemented with other tools like user stories. He concludes by pointing out that what is in the book are current practices among game companies and not just theory.

A. Godoy & E. Barbosa (Godoy & Barbosa, 2010) suggest a similar adaptation to that of Keith where they use the Scrum framework and extreme programming to create the proposed Game-Scrum. They try to solve some specific challenges unique to the game development: artistic content, project scope, project management, team organization. They test Game-Scrum by developing a mini game, but the results show that some of the challenges still remain, and the proposed method needs to be refined and matured.

R. Kortmann & C. Harteveld (Kortmann & Harteveld, 2009) proposes the Triadic Game Design Development Model, which is based on agile development models and the design of three game components: reality, which determines the subjects, variables and definitions of the game; meaning, which incorporates aspects such as communications, learning, rhetoric and opinion; and play, which is affiliated to media studies, game design, and human-computer interaction. They test their model by conducting a workshop to create the draft concept of a game. Even if the draft concept can validate the model to some degree, a full game development iteration is required to observe the behavior of the model.

This section covers the agile game development and analyzes some cases and proposals on how to integrate the agile development paradigm with game development. We observe that Scrum is the most common used agile framework in game development, which may be because it is a framework instead of a method, so it can be adapted easily to the game development specific conditions.

2.4.3 Pattern in Game Development

This section discusses process pattern and how they can be used to do agile development. We took Alexander's (Alexander, 1979) description of a pattern "Each pattern describes a problem which occurs over and over again in our environment, and then describes the core of the solution to that problem, in such a way that you can use this solution a million times over, without ever doing it the same way twice".

The pattern approach in software engineering is an adaption of the work of Alexander et al. (Alexander et al., 1977) which was not intended for software engineering. This concept of pattern was refined by May & Taylor in their work (May & Taylor, 2003) when they describe each item required in a pattern, which are: name, context, problem, forces, solution, rationale, resulting, context, and related patterns.

Software Engineering patterns may be divided into the following categories: design of software solutions, process improvement and software configuration management. But we are analyzing process improvement. Within process improvement there are the following sub-categories:

- Improvements Patterns, which are used to "know how implement process improvement initiatives"(Landaeta, García, & Amescua, 2008).
- Collaboration patterns, which are used to "know how to implement computer supported collaborative work approach"(Schummer, 2004).
- Process pattern, which are used to "help software engineers to manage the knowledge related to the practices on how to develop software projects effectively" (Martín, Guzmán, Urbano, & Llorens, 2012).

As we find out that Scrum is suited for game development we decide to analyze work that has implemented Scrum with process pattern. Martin et al. (Martín et al., 2012) proposes the Software Development Project Pattern (sdPP) framework, which is a process pattern that "is composed of a data model for representing project patterns that include effective practices to develop software projects and a software tool to manage this type of knowledge objects". Figure 2-11 shows the description model of the sdPP.

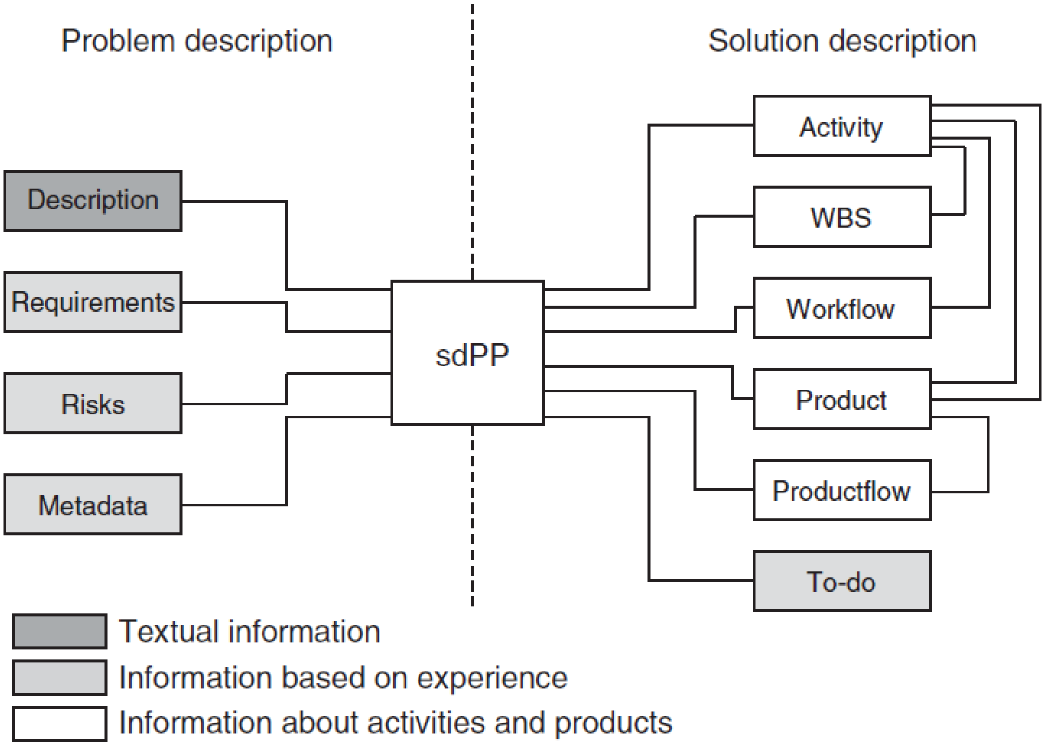


Figure 2-13 sdPP model (Martín et al., 2012)

To prove the sdPP they use software development methods such as extreme programming, Scrum, Rational Unified Process (Kroll & Kruchten, 2003) and Craig Larman (Larman, 2003) and create an instance of the sdPP for each of these models. They test these instances by using them in developing a web based software and compare them with the same web based software, but without the use of sdPP. The results indicate that the execution of sdPP contributes to increasing the quality of the artifacts.

2.5 Summary

We have presented the concept of game design in video games, its relation with requirements engineering and the current problems of game design faces. One of the main problems is the game design documentation in the GDD, since it lacks structure, formalism and relations.

We have examined the player experience and emphasize how important it is to the game's success. We analyze the methods to measure the player experience and to handle player experience. We conclude that measuring the player experience in late stages and finding out problems is costly, and the few techniques that can be applied in early stages like heuristics, are too generic to be useful.

We have presented some game development methodologies, and have seen how traditional software development models were adapted to game development. We find out that agile development is better suited to game development due to its iterative nature Table 5-1 in Chapter 5 will bring more evidence of this statement. We find that process pattern can be used to adapt software development models, and that the evidence shows that by using these patterns to adapt the model, the resulting artifacts have an increased quality.

CHAPTER 3
FORMALIZING GAME DESIGN FROM REQUERIMENTS ENGINEERING PERSPECTIVE

3.1 Problem Description

Major software companies are interested in video game development due to its high industry revenues and its growing capability ("Video Game Industry Statics," 2010). Game design is the cornerstone of video games companies. As seen in chapter 2, video game development has three main stages. Video games are designed at the pre-production stage, which generates a product commonly called the Game Design Document (GDD). In the production stage, the GDD is used for software design, development and validation. In the post-production stage video games are distributed and monitored after delivery, for the purpose of taking corrective action, along with analysis of the company´s expectations of the sales and performance of the video game product. Therefore, a GDD plays a key role throughout the video game development process.

The scientific community has addressed the complexity of video game design mainly by trying to formalize the pre-production stage. Some argue that applying requirements engineering best practices may avoid rework during production stage (Callele et al., 2005). Some point out that a lack of formality in the GDD cause problems that reduce ROI (Return On Investment) (Bethke, 2002). Others point out that one of the most common causes of failed development is ambiguity and lack of communication(Oxland, 2004). In our opinion, requirements engineering best practices may support pre-production and production stages, by bringing structure, detail and establishing relationships among video game elements in order to improve playing time experience.

A formal GDD may provide support for the transition between the pre-production stage and production stage, reducing rework. We propose an improved GDD based on a comparative analysis of GDD documents found in the literature. Furthermore, we propose the resulting GDD template be formalized with other well-known Software Requirements Specification (SRS) standards. Finally, our GDD proposal is then compared with a commercial GDD for feedback.

3.2 *Solution Approach*

Game design and the use of a GDD are common topics in the video game field. Regarding the GDD many authors agree that there is no established structure for a GDD, since there are significant differences from game to game, especially in the video game genre (Baldwin, 2005; Oxland, 2004; Rogers, 2010; Rouse, 2004). However, there is a set of common elements of game design. We use these common elements to create a structure for a GDD template and the SRS best practices to formalize this template.

3.2.1 Game Design Document Structure

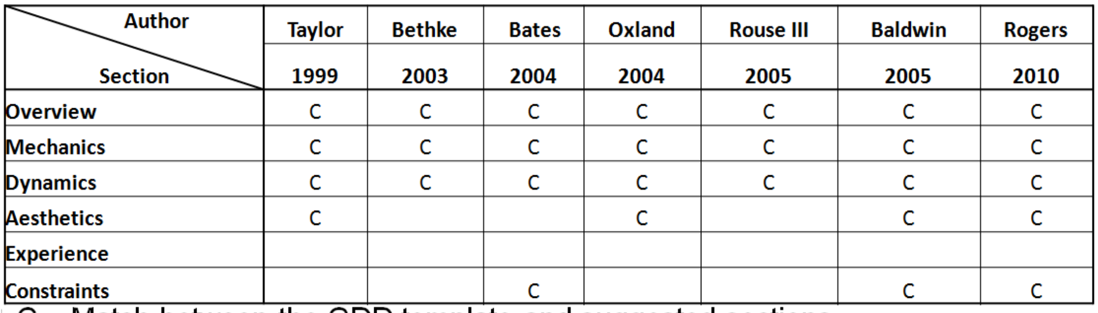
The purpose of this section is to identify the principal sections and structure of a GDD. There is little available evidence of GDD structures from the game industry, therefore, the elements identified from the proposals of (Baldwin, 2005; Bates, 2004; Bethke, 2002; Oxland, 2004; Rogers, 2010; Rollings & Adams, 2003; Rouse, 2004; Taylor, 1999) available in literature are the following: overview, mechanics, dynamics, aesthetics, experience and assumptions and constraints. Each element can be described as follows:

1) Overview Section: All authors suggest that a GDD should include a section that summarizes the key elements of the game to have a summary of the main features of the game and to remember why the game is being developed. (Baldwin, 2005; Bates, 2004; Bethke, 2002; Oxland, 2004; Rogers, 2010; Rollings & Adams, 2003; Rouse, 2004; Taylor, 1999). Some authors even include a subsection of goals or objectives of the game (Oxland, 2004; Rogers, 2010). We consider the overview a key section not only because it is a summary of the main features of the game, but because it helps transform the game idea into a game concept, which allows a better understanding of the size and viability of the game.
2) Mechanics Section: The term mechanics is used to describe game elements (e. g. player character) and intended interaction (e.g. a challenge). We decided to separate them in order to achieve a better game structure and increase reuse. Among the authors, the way of describing the game elements have common sections like mechanics, characters or assets list (Baldwin, 2005; Bates, 2004; Bethke, 2002; Oxland, 2004; Rogers, 2010; Rollings & Adams, 2003; Rouse, 2004; Taylor, 1999). The game mechanics do not describe the game directly, but rather define the parts that are going to build the game. They describe the characteristics of these mechanical elements, their behaviors and the way these elements interact with each other.

3) Dynamics Section: All authors have common sections that contain game interactions such as interfaces, levels or artificial intelligence proposals (Baldwin, 2005; Bates, 2004; Bethke, 2002; Oxland, 2004; Rogers, 2010; Rollings & Adams, 2003; Rouse, 2004; Taylor, 1999). The dynamics involve an interaction between the player and the game. It is here that the described mechanical elements are used to build the gameplay, such as where the interfaces to which the player will interact with the game are set and where the world, the flow and the story of the game are described.
4) Aesthetics Section: What the player perceives by his visual and auditory senses. Most authors cover the visual aspects in a document called the art bible. Mark Baldwin (Baldwin, 2005) suggests an art section abbreviating the art bible in his template. The auditory is mentioned by some authors (Oxland, 2004; Rogers, 2010; Taylor, 1999). The aesthetics of the game are mechanical and dynamic properties. These properties in particular are associated with the senses through which the player perceives, mainly (but not exclusively) visual and auditory.
5) Experience Section: Creating enjoyable player experience is fundamental for the game's success (Schell, 2008). Player experiences are enriched by mechanics, dynamics and aesthetics of the game. Playability can be used to link game design to player experience (Nacke et al., 2009). Therefore, defining the expectations of player experiences may lead to the improvement of the game and to the establishment of a base line to test the experiences in production. These experiences are not being considered in the surveyed GDD. Due to the complexity of managing the player experiences, this section was removed from our GDD, improved and replaced with the creation of a specific method to handle the player's experience, all of which is described in Chapter 4.
6) Assumptions and Constraints Section: The technical limitations are covered by some authors by including a summary of the technical bible in the GDD (Baldwin, 2005; Bates, 2004; Rogers, 2010). The assumptions and restrictions are an essential part in the design of video games. They allow the designers to understand the context and restrictions on which the game should be constructed so that the game does not exceed the context and constraints, preventing serious problems that can arise later for not adhering to them.

We present in Table 3-1 the suggested sections and the GDD templates that match the sections. As can be seen there is an agreement between authors on the overview, mechanics and dynamics sections. Aesthetics and constraints are not covered by half of the authors. And any author does not cover experience.

Table 3-1 Suggested GDD section and template section relation



Author	Taylor	Bethke	Bates	Oxland	Rouse III	Baldwin	Rogers
Section	1999	2003	2004	2004	2005	2005	2010
Overview	C	C	C	C	C	C	C
Mechanics	C	C	C	C	C	C	C
Dynamics	C	C	C	C	C	C	C
Aesthetics	C			C		C	C
Experience							
Constraints			C			C	C

C = Match between the GDD template and suggested sections

3.2.2 Requirements Engineering Applied to Game Design

In this section we discuss the GDD formality, understood as the structure, relations and detail it contains. We followed the IEEE Std 830-1998 (Engineering & Committee, 1998) (reaffirmed in 2009) for the purpose of comparing the SRS with a GDD.

Regarding the structure, the SRS has three main sections: a) an introduction that provides an overview of the SRS, b) an overall description that contains the general facts that affect the product and its requirements; it provides background for the requirements, and c) details of the specific requirements that a designer and a tester can use for designing and testing a software product.

With respect to relations, there are different types of requirements and the relations between them need to be clear, and cross-referenced to related documents. The specific requirements should be uniquely identifiable. Wiegers (Wiegers, 2003) gives a hierarchical description of the requirement types dividing them into business, user and system levels.

Concerning detail, all the specific requirements should be stated in conformance with the following characteristics: correct, unambiguous, complete, consistent, and ranked for importance and/or stability, verifiable, modifiable, and traceable.

As shown in Table 3-2 SRS structure, detail and relations may improve a GDD:

1) The structure of the specific requirements section introduces best practices, such as organizing specific requirements in which the object

organization described in the SRS standard can be used for bringing together the mechanics.

2) The document references from the introduction section would be an aid to game developers in identifying how documents are connected. That way, all decisions can be traced backward and forward along the development process.

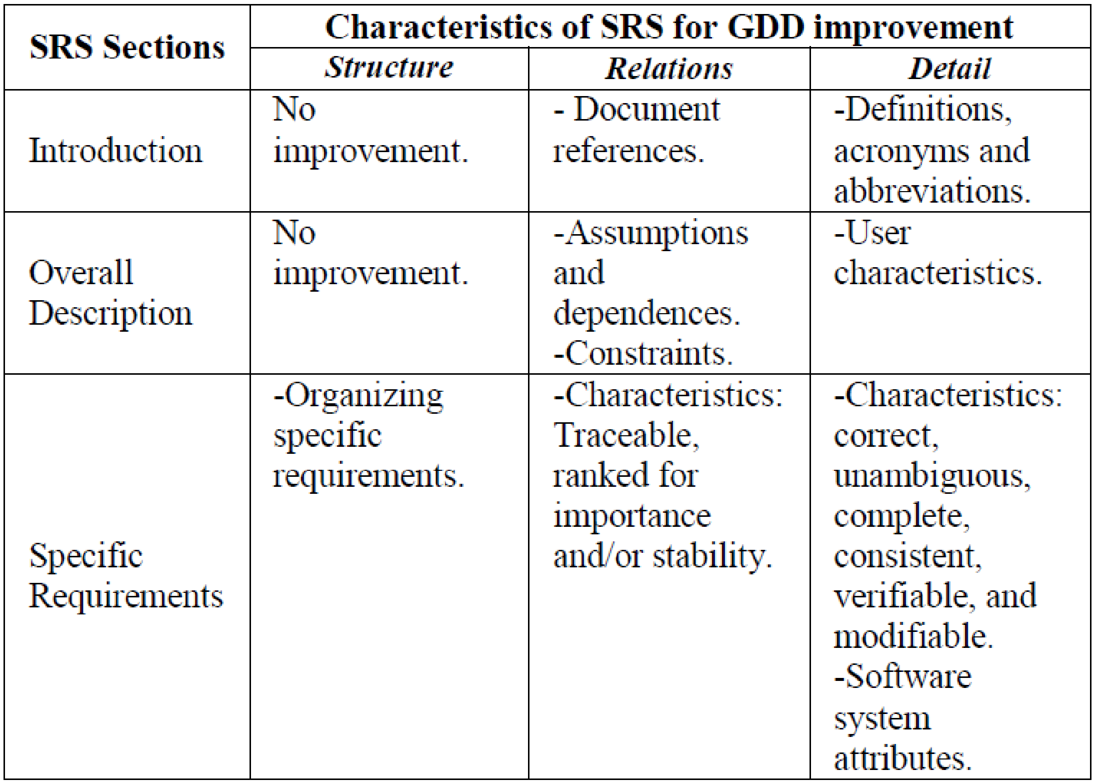
3) The SRS overall description section may enrich the relations of the GDD by making explicit descriptions of the assumptions and dependencies. In this way, developers would know which parts of the game have to be double-checked later in the project. The constraints can help the game designers to know the limitations or boundaries to take into account when designing the game.

The SRS specific requirements section may enhance the relations in a GDD. Traceability and stability over game elements may help to estimate the effort needed when changes occur. Ranking importance and stability of game elements is crucial for decision-making tradeoffs.

The GDD detail may be improved by the SRS with a definitions, acronyms and abbreviations section, making the document easy to read and establishing a common language for the creation of a GDD among stakeholders during the development process.

The User characteristics part of the overall SRS description may improve the GDD detail by allowing game designers to know who might play the game and to adjust the interface complexity for different gamer profiles.

Table 3-2 Improvement of GDD with SRS



SRS Sections	Characteristics of SRS for GDD improvement		
	Structure	*Relations*	*Detail*
Introduction	No improvement.	- Document references.	-Definitions, acronyms and abbreviations.
Overall Description	No improvement.	-Assumptions and dependences. -Constraints.	-User characteristics.
Specific Requirements	-Organizing specific requirements.	-Characteristics: Traceable, ranked for importance and/or stability.	-Characteristics: correct, unambiguous, complete, consistent, verifiable, and modifiable. -Software system attributes.

Most of the characteristics of the specific requirements from the SRS help to improve the detail in the GDD. A detailed GDD with correct, unambiguous, complete, consistent, verifiable and modifiable game elements is more suitable for designing the software of the game. Considering software system attributes in a GDD can help to identify requirements not based on functionality. Since video games have special requirements not addressed right now by any standard (Callele et al., 2005), additions may be made to consider requirements such as feelings (Callele, Neufeld, & Schneider, 2006) or any other requirement based on user experiences (J. González, Padilla, et al., 2009).

Up to this point, we have identified that SRS best practices improve the pre-production and consequently production and post-production stages through:

- Relations with other documents.
- Common language for common understanding.
- Knowledge of game parts for reviews.
- Decision-making based on tradeoffs of game parts.
- Limitations or boundaries of video game.
- Relation of complexity with gamer profile.
- Organization of game requirements.

- Requirements traceability for decision-making.
- Requirement characteristics as a means to formalize a GDD.
- Quality attributes of video games (e.g. feelings and experiences).

3.3 Solution Description

This section shows how the proposal is improved with SRS characteristics. Table 3-2 shows the initial proposal for a GDD based on the analysis of reviewed GDDs and incorporating identified SRS best practices. It should be noted that the i(improved)GDD is best suited for video games with a progression path with a beginning and end, divided by levels, missions or chapters. The iGDD targets emerging video game companies or software companies interested in video game development with little experience in game design. Next, we present the descriptions for each iGDD section.

Overview: There are two principal additions to this section. One is a reference subsection relating iGDD with others documents in the project. The second subsection establishes a common understanding for the document reader by adding definitions, abbreviations and acronyms.

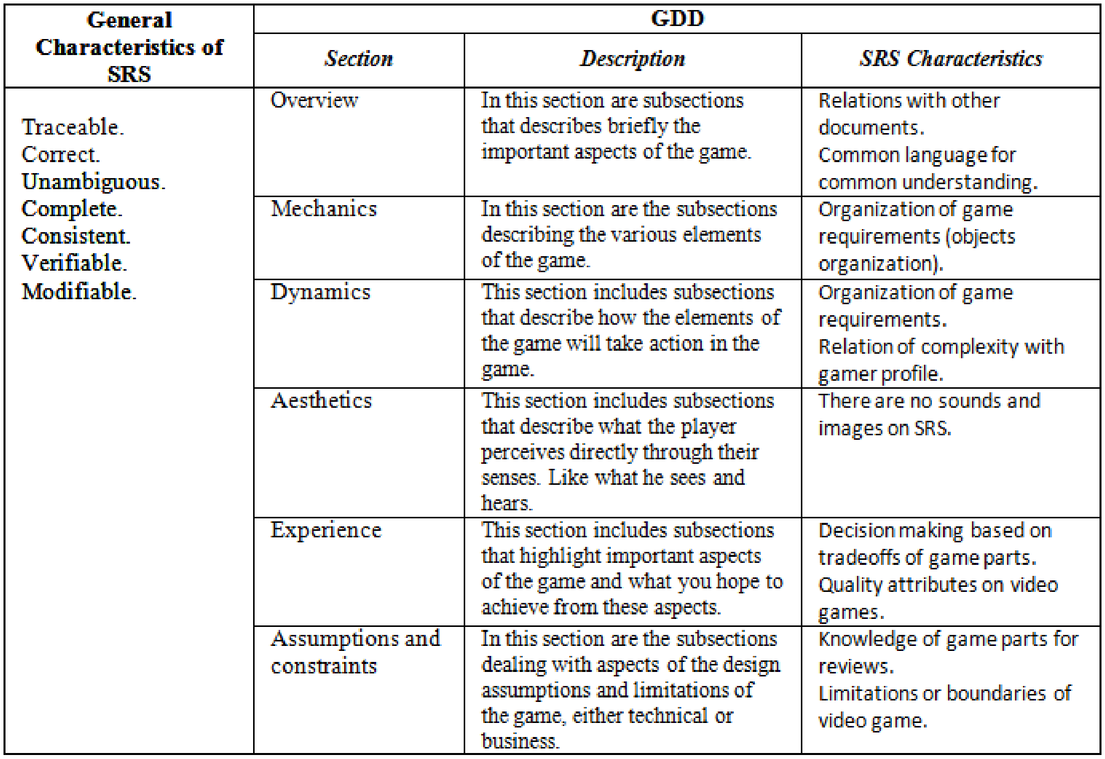
Mechanics: This section is used to describe objects in the game such as the player avatar or an enemy. Hence, by using the object organization of requirements, mechanics can be described in a practical way. Categories are used to classify and specify general attributes and behavior that share category elements such as "enemy". Elements of the game are described by defining their attributes, behavior and rules of how elements can interact with each other.

Dynamics: There are ways of organizing requirements that can be adapted to support the dynamics specification of the game, such as features, responses, functional hierarchy and system mode. The gamer profile added in this section may be used to adjust the interfaces and challenges of the game. The elements described in the mechanics section are added to a field or levels in the game world. Objectives, rewards and challenges are described, as well as how the player will learn to play the game and how the game can be balanced.

Aesthetics: There is no support of SRS elements for defining/capturing what a gamer will hear and see.

Experience (removed from the improved GDD and covered in chapter 4): In this section, on one hand, we add the importance and stability of different parts of the game. Therefore a decision change can be taken knowing the impact of it. On the other hand, we add quality attributes. Due to the fact that a video game is a software product, some of these attributes are taken into consideration using current standards, and other attributes may be incorporated from recent work on feelings and experiences attributes (J. L. González et al., 2009).

Table 3-3 Description of Resulting GDD



General Characteristics of SRS	**GDD**		
	Section	***Description***	***SRS Characteristics***
Traceable. Correct. Unambiguous. Complete. Consistent. Verifiable. Modifiable.	Overview	In this section are subsections that describes briefly the important aspects of the game.	Relations with other documents. Common language for common understanding.
	Mechanics	In this section are the subsections describing the various elements of the game.	Organization of game requirements (objects organization).
	Dynamics	This section includes subsections that describe how the elements of the game will take action in the game.	Organization of game requirements. Relation of complexity with gamer profile.
	Aesthetics	This section includes subsections that describe what the player perceives directly through their senses. Like what he sees and hears.	There are no sounds and images on SRS.
	Experience	This section includes subsections that highlight important aspects of the game and what you hope to achieve from these aspects.	Decision making based on tradeoffs of game parts. Quality attributes on video games.
	Assumptions and constraints	In this section are the subsections dealing with aspects of the design assumptions and limitations of the game, either technical or business.	Knowledge of game parts for reviews. Limitations or boundaries of video game.

A base model of the improved GDD is shown in Figure 3-1, this model gives a better understanding of the main concepts and how they are related to each other. Of importance is the relation between mechanics, dynamics and aesthetics. The gameplay is described in the "Game Modality Elements" class in the dynamics section. This Class is related to the "Mechanics Game Element" class from mechanics, which are the construction blocks for the "Game Modality Elements" class. Classes from mechanics and dynamics are related to the class "Aesthetics Property", which means they can have a visual, sound or other property that can be perceived with the senses. To illustrate these relations let's suppose we have an object with the name "Basic Enemy" which belong to the class "Mechanics Game Element" and we want to create an object with the name "Level 1 Challenge 1" from the class "Game Modality Elements". We decide that the "Level 1 Challenge 1" object will have 5 "Basic Enemy" objects. This means that every "Game Modality Elements" object will have inside one or more "Mechanics Game Element" Object. A more detailed example can be seen in section 3.4.

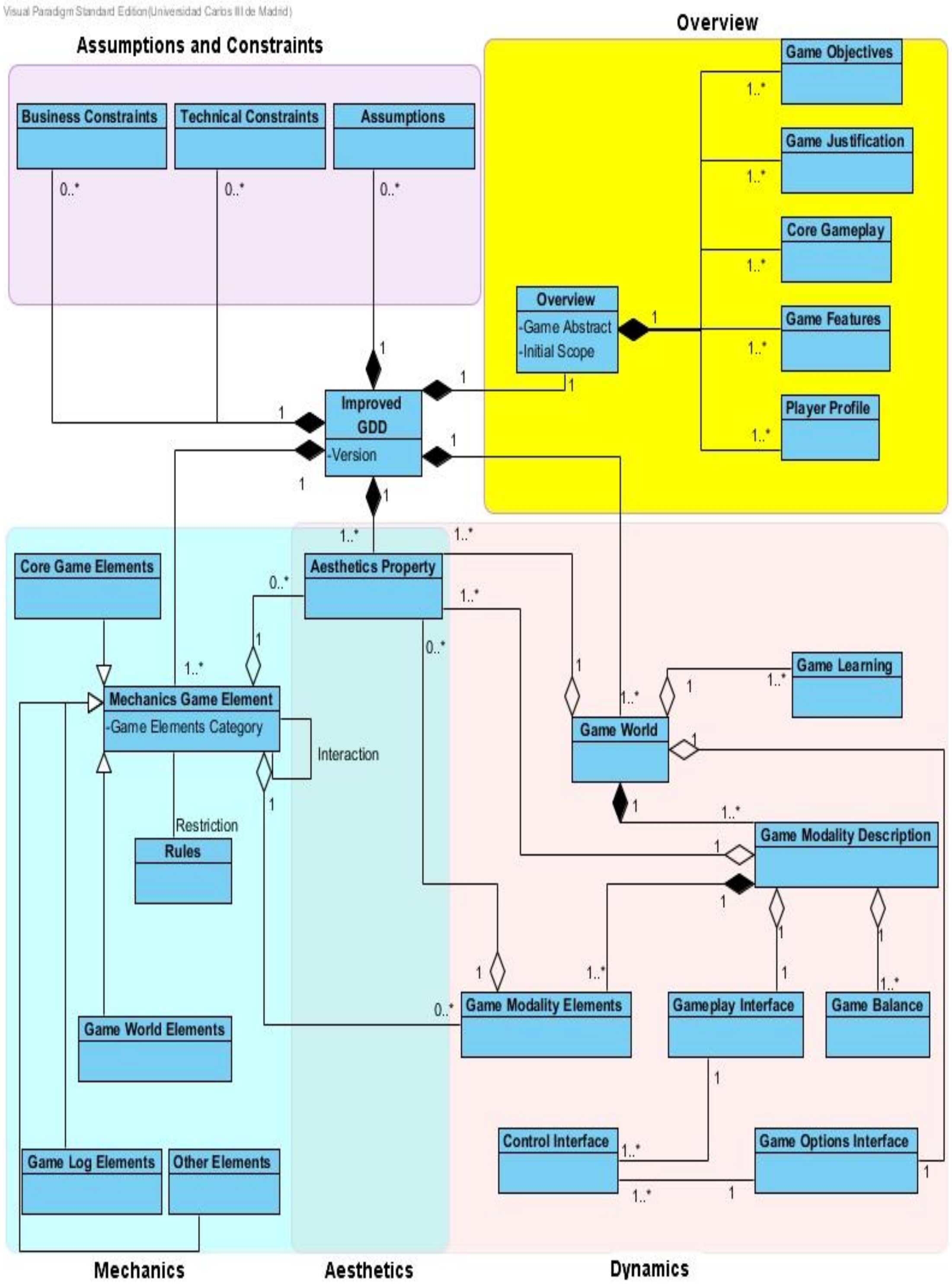


Figure 3-1 Improved GDD Model

3.4 Example

This section describes an example of our improved GDD to see how the game design of an already created game might be represented. The game Donkey Kong™ is used, to illustrate this example. The focus is mainly on the overview, mechanics and dynamics section of the iGDD. The section finishes by covering some illustrative samples of each iGDD section and gives the conclusions.

3.4.1 Example Overview

To create the overview section we have to make several assumptions given that we don't know some of this information, and thus we will focus on the information that we can validate in the game. The abstract of the game in the overview section is the following:

The game is about a carpenter who has to fight several obstacles to reach the top of the level. At the top he can rescue his girlfriend, who was kidnapped by a giant ape named Donkey Kong. The game will have several levels; at the end of each level Donkey Kong will take the carpenter's girlfriend and take her to the top of a new level.

The game core gameplay of the game is the following:

The player will start at the bottom of a series of platforms; he will have to fight several obstacles while working his way to the top of each of the platforms. The player will be able to move sideways, climb stairs and jump. The player will accumulate points based on the actions. Jumping obstacles and collecting certain objects are examples of how to earn points. There is a time limit for completing each level. Each level has different challenges due to the enemies and platforms that change. There are a limited number of attempts that the player can use to beat the level; these attempts are represented by the carpenter lives. If the player reaches the last level and rescues the carpenter's girlfriend, the game starts again but with increased difficulty.

The story summary is the following:

Donkey Kong is Mario's pet, the game's protagonist carpenter. Because of the abuses of Mario to his pet, it escapes and kidnaps the carpenter's girlfriend Pauline. The monkey climbs a platform with his victim. Mario goes to rescue his girlfriend, but on the way he must dodge several obstacles. On several occasions Mario is about to rescue Pauline, but when he is close to reaching her, Donkey Kong takes Pauline and climbs higher again. At the top, Mario removes several rivets to make Donkey Kong fall so he can rescue Pauline.

The initial scope of the game is the following:

It is intended that the game's levels are different from each other. Therefore the game will not have more than four levels. It is anticipated that a group of three people in four months will be able to finish the game.

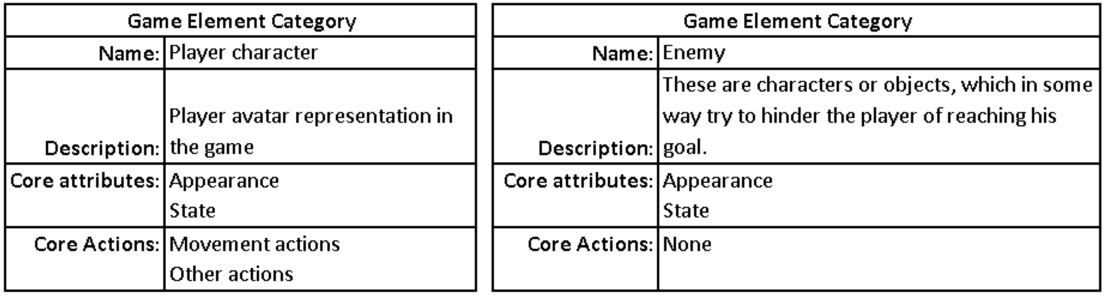
By reading the above information, we can conclude that the overview is not only the starting point to document a video game, but it is the best way of communicating the game idea among the stakeholders.

3.4.2 Example Mechanics

The mechanics were created from the final version of the game. We use all the game elements found in the game to complete the game mechanics. We will describe some illustrative samples of the game mechanics.

The game elements categories are a prerequisite for defining any game element. We will use two main categories to illustrate our example. One category is the "Player character" and the other is "enemy" Table 3-4 shows the detail of these categories.

Table 3-4 Game elements categories

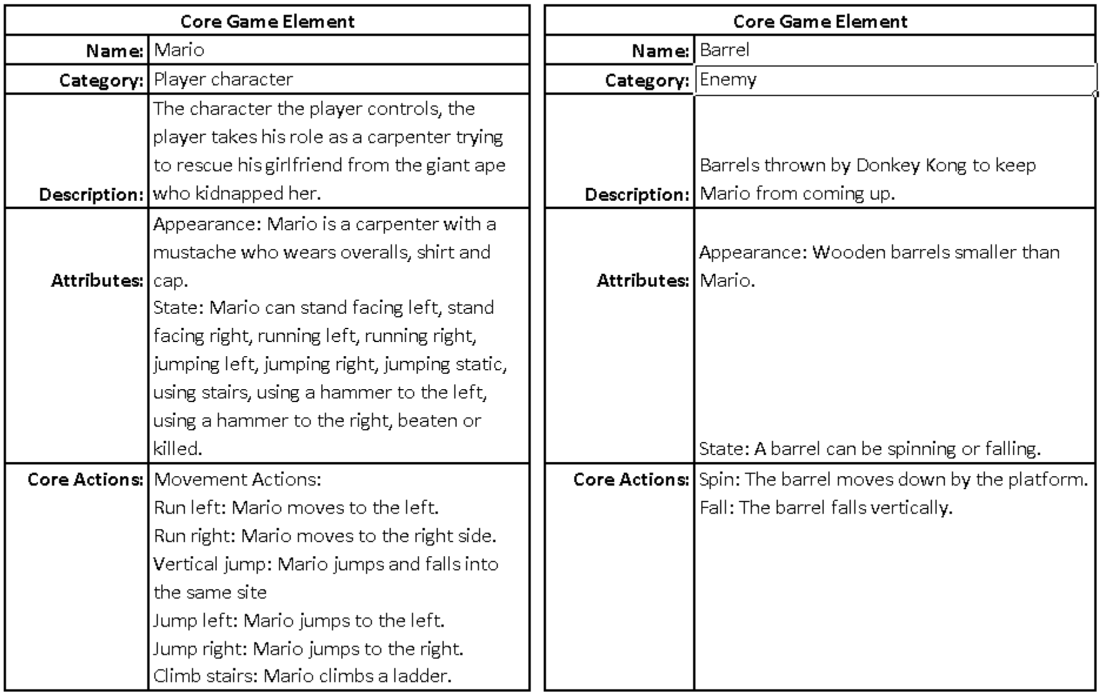


Game Element Category	
Name:	Player character
Description:	Player avatar representation in the game
Core attributes:	Appearance State
Core Actions:	Movement actions Other actions

Game Element Category	
Name:	Enemy
Description:	These are characters or objects, which in some way try to hinder the player of reaching his goal.
Core attributes:	Appearance State
Core Actions:	None

We define some game core game elements derived from these categories. From the "Player character" category we defined the core game element named "Mario". Mario is the only element that belongs to this category. From the "Enemy" category we defined several core game elements. Table 3-5 describes the core game element "Mario", as well as the core game element "Barrel" which belongs to the category "Enemy".

Table 3-5 Core game elements examples



Core Game Element	
Name:	Mario
Category:	Player character
Description:	The character the player controls, the player takes his role as a carpenter trying to rescue his girlfriend from the giant ape who kidnapped her.
Attributes:	Appearance: Mario is a carpenter with a mustache who wears overalls, shirt and cap. State: Mario can stand facing left, stand facing right, running left, running right, jumping left, jumping right, jumping static, using stairs, using a hammer to the left, using a hammer to the right, beaten or killed.
Core Actions:	Movement Actions: Run left: Mario moves to the left. Run right: Mario moves to the right side. Vertical jump: Mario jumps and falls into the same site Jump left: Mario jumps to the left. Jump right: Mario jumps to the right. Climb stairs: Mario climbs a ladder.

Core Game Element	
Name:	Barrel
Category:	Enemy
Description:	Barrels thrown by Donkey Kong to keep Mario from coming up.
Attributes:	Appearance: Wooden barrels smaller than Mario. State: A barrel can be spinning or falling.
Core Actions:	Spin: The barrel moves down by the platform. Fall: The barrel falls vertically.

The description of both core game elements can be used to begin the construction of different artifacts related to these elements, such as animations and sound effects.

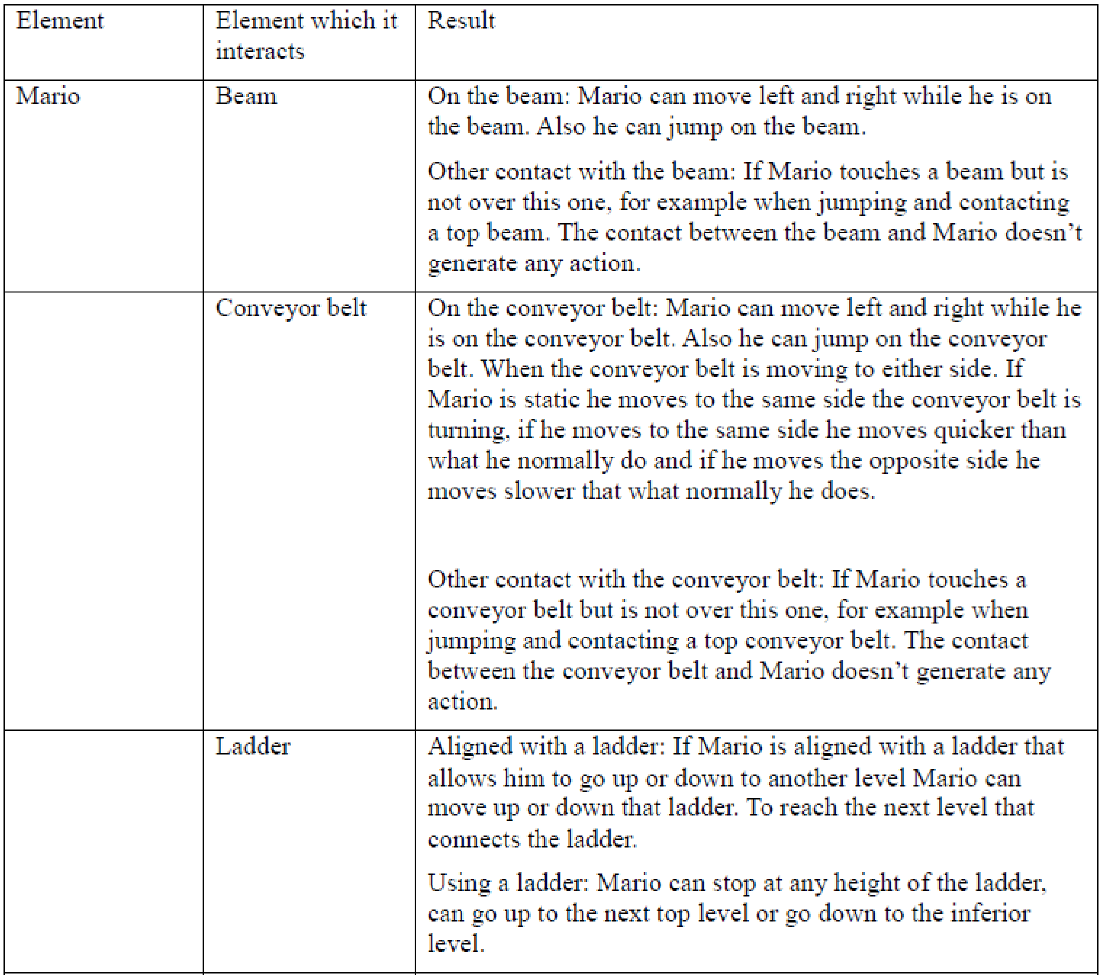
There are other important game elements as well. The game log elements are those, which are used to keep track of important elements that the game has. In this case, we have the game log elements:

- "Score" which registers the score of the current player in the game.
- "High score" which registers the maximum score achieved in the game.
- "Bonus time" which registers the time that the player has to finish a level.
- "Lives" which registers the amount of chances that the player has to pass a level after failing.

These are examples of game log elements in the game. There are other mechanic game elements but this particular game does not have such elements.

To finish the game mechanics we need to incorporate the rules that dictate how the game elements should interact with each other. Table 3-6 shows an example of how the game element Mario interacts with other elements. It shows how Mario interacts with some core game elements that belong to the category platform, which Mario uses to move in a level. A similar table should be created to cover the rest of the elements that may interact with Mario in the game and new tables to describe the interactions of other elements among them, like enemies with platforms.

Table 3-6 Interaction Rules



Element	Element which it interacts	Result
Mario	Beam	On the beam: Mario can move left and right while he is on the beam. Also he can jump on the beam. Other contact with the beam: If Mario touches a beam but is not over this one, for example when jumping and contacting a top beam. The contact between the beam and Mario doesn't generate any action.
	Conveyor belt	On the conveyor belt: Mario can move left and right while he is on the conveyor belt. Also he can jump on the conveyor belt. When the conveyor belt is moving to either side. If Mario is static he moves to the same side the conveyor belt is turning, if he moves to the same side he moves quicker than what he normally do and if he moves the opposite side he moves slower that what normally he does. Other contact with the conveyor belt: If Mario touches a conveyor belt but is not over this one, for example when jumping and contacting a top conveyor belt. The contact between the conveyor belt and Mario doesn't generate any action.
	Ladder	Aligned with a ladder: If Mario is aligned with a ladder that allows him to go up or down to another level Mario can move up or down that ladder. To reach the next level that connects the ladder. Using a ladder: Mario can stop at any height of the ladder, can go up to the next top level or go down to the inferior level.

3.4.3 Example Dynamics

The dynamics define the main interactions with the player. In the game Donkey Kong™ first we describe the game world, the place where the game takes place, the intended flow of the player in this world, and if the game has a story, the description of this story. The following text describes the flow of the game:

"The game has 4 levels - before the first level and between levels there is a transition. Upon completion of the 4 levels, the levels start repeating but with increasing difficulty.

1. Title
2. Initial animation
3. Challenge screen
4. Level 1
5. End of level 1
6. Challenge screen
7. Level 2
8. End of level 2
9. Challenge screen
10. Level 3
11. End of level 3
12. Challenge screen
13. Level 4
14. Final animation

If the player loses a life he returns to the challenge screen above the level in which he lost a life. If the player loses all his lives the “Game Over” screen appears and returns to the title screen. "

The Donkey Kong™ game is divided into levels. The next step is to define the elements that will integrate these levels. The objectives are things that the player seeks to achieve in the game. In a level there can be one primary objective and many secondary objectives. The following text describes the primary objectives in the game:

"At each level of the game except the final level, the main **goal** is to reach a certain platform on the top of the screen because this platform is where Pauline is found or the platform where Donkey Kong is. In the last level, the main goal of the game and the win condition is to remove all the rivets of the level."

Another important element in the levels is the reward, which is what the player earns. These rewards can be explicit or implicit, and the following text describes the explicit rewards in the game:

“All the rewards are given as a function of the score. If the player’s score is high enough, the game records his score on a permanent basis as the best in the game until someone else beats it. The way to obtain score points are:

- Jump an enemy.
- Eliminate an enemy with the hammer.
- Pick an object of Pauline.
- Remove a rivet.

- Finish the level before the time bonus ends."

The next dynamics element is a key element in most games. The challenges are what the player has to overcome in order to progress in the game. In the game Donkey Kong™ there are the following challenges:

"The main challenges of the game will be:

- Evade enemies.
- Finish on time.
- Avoid falling.
- Appropriate use of platforms."

Once the dynamics elements are defined, the next step is the construction of levels. To describe each level, first we describe each scenario where the level takes place, and then we create the objectives, rewards and challenges of the level.

Once we have the game elements with its attributes and actions described, and we have detailed how they can interact, we can create the setting where these elements will interact with the player, and the dynamics of the game. Figure 3-1 shows the scenario creation of level 3 in the game. The figure should be described in detail; the following is an initial description of the first sections in Figure 3-1:

"Section 1 has 4 beams and an elevator of beams. The beams are 3 times the width of Mario. The first beam is at the bottom left and it connects with the bottom engine of the elevator of beams. The second beam is above the first separated by one beam height. The third beam is above the second one up to 4 times the height of a beam; a ladder at the middle of the beams connects them. The fourth beam is above the third 7 times the height of a beam; a ladder on the right corner of the beams links them. The elevator of beams has its lower engine at the base of the screen at one side of the first beam, and its upper engine is where section 1 and 6 joins. The elevator has 3 beams that travel from lower to the top engine. Mario can jump from a beam of the elevator to a beam in section 2."

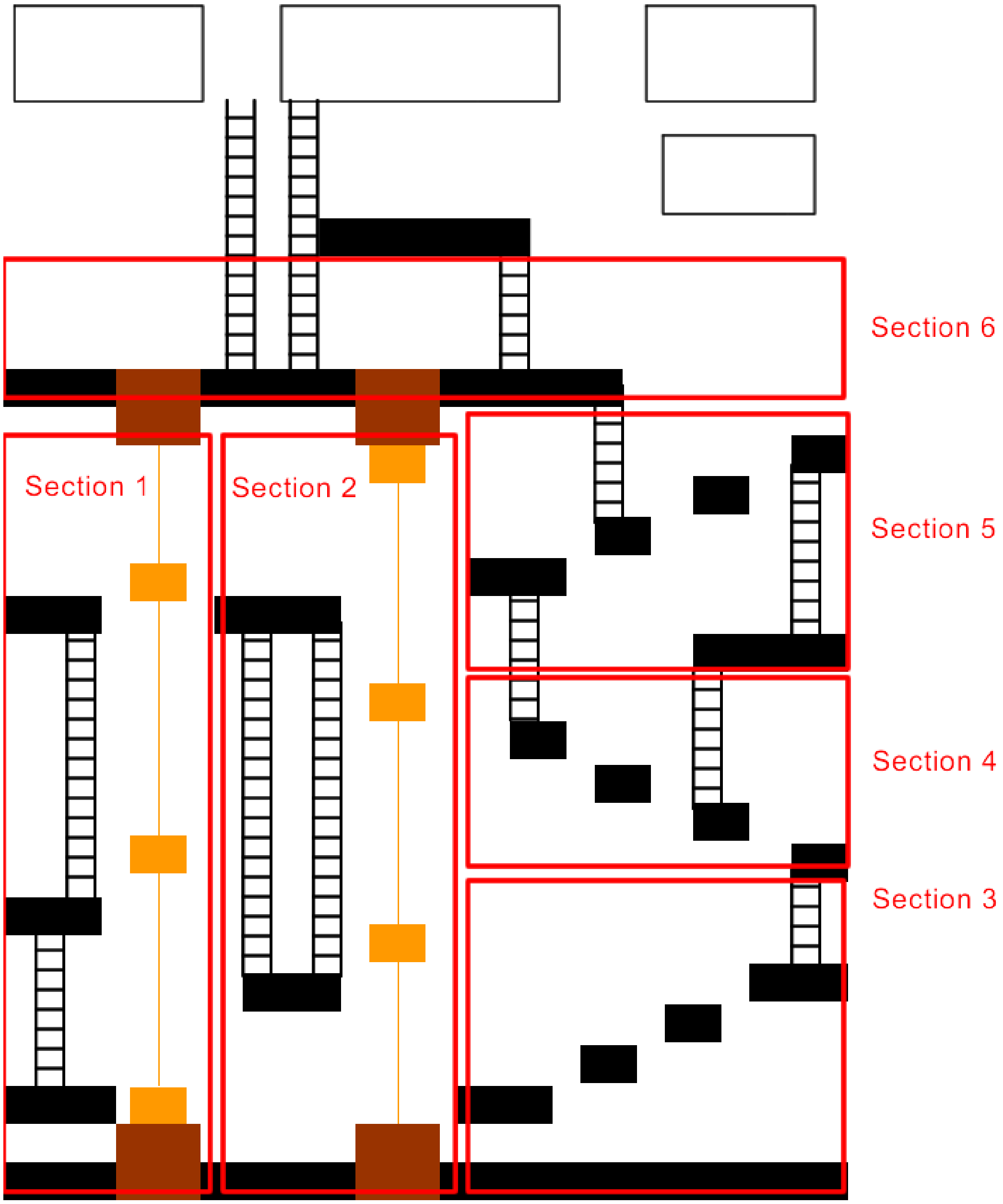


Figure 3-2 Level 3 scenario

The objectives, rewards and challenges of level 3 in the game are shown in Figure 3-2. Just as with the scenario, the objectives, rewards and challenges should be described in detail. The following text shows a description of the element placement in the level:

“Mario: He starts on the second beam from bottom to top, by the ladder that is in that beam of section1.

Pauline: She is on the top beam of the level.

Donkey Kong: Located on the first beam of section 6, at the left of the screen, right next to the first ladder of the level.

Pauline’s bag: Her bag is on the top beam of section 5.

Pauline’s Umbrella: Found on the upper beam of section 1.

Fireball: There are 2 fireballs in the level. The first is found in section 2, starting from the center of the upper beam, and can move across the beam, the ladders, and the other beam connected by the ladders. The second begins in section 5, under Pauline's bag, and can move to the 2 lower beams using the ladders.

Spring: Each spring appears at a certain time to the left of Donkey Kong and begins to bounce the beam of section 6, and at the end it falls to the bottom of the screen."

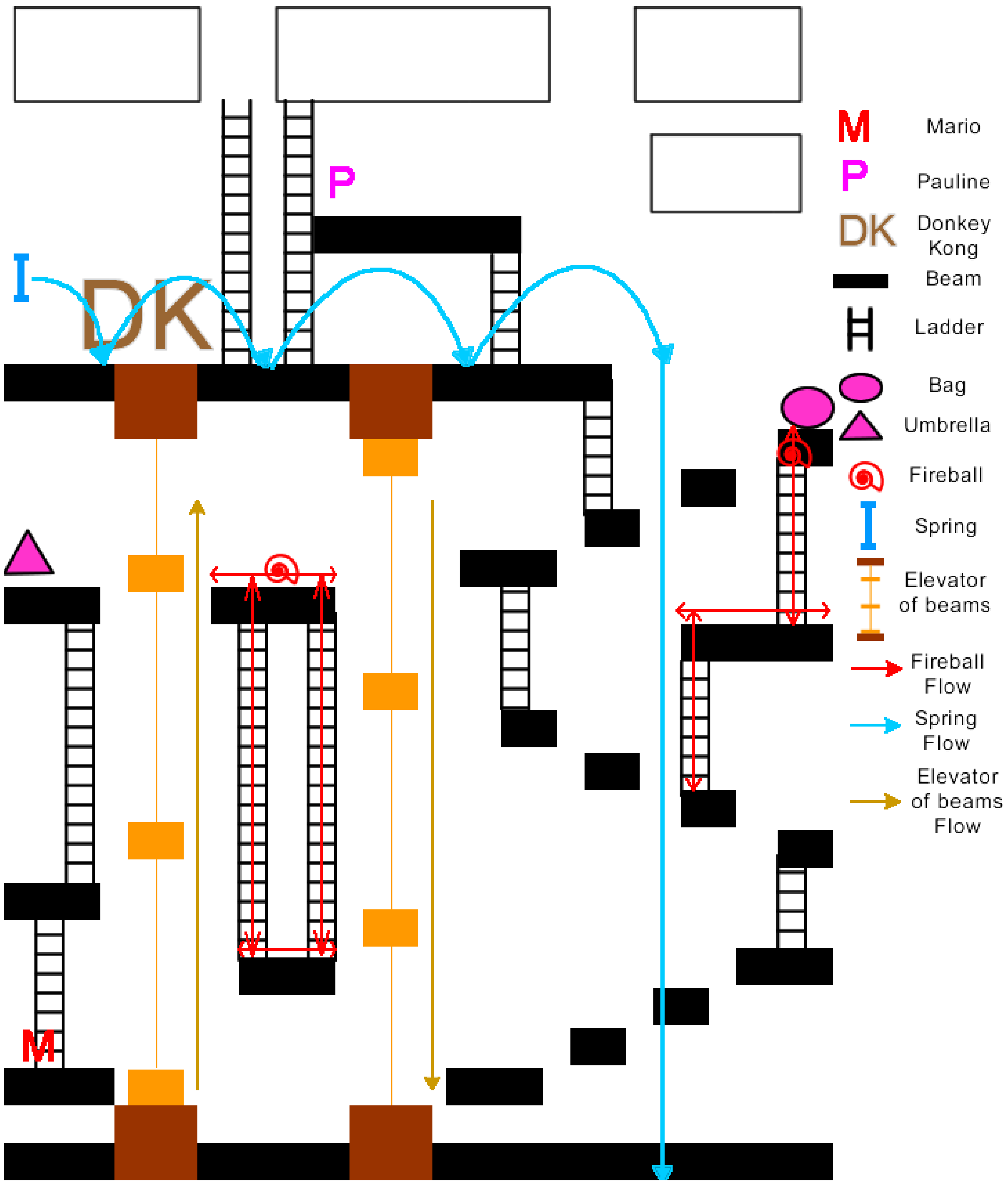


Figure 3-3 Elements placement

Another important concept in dynamics is the interface, the description of how the player will interact with the game in a software and hardware level. Next we describe the home screen in the game and the information that it gives to the player:

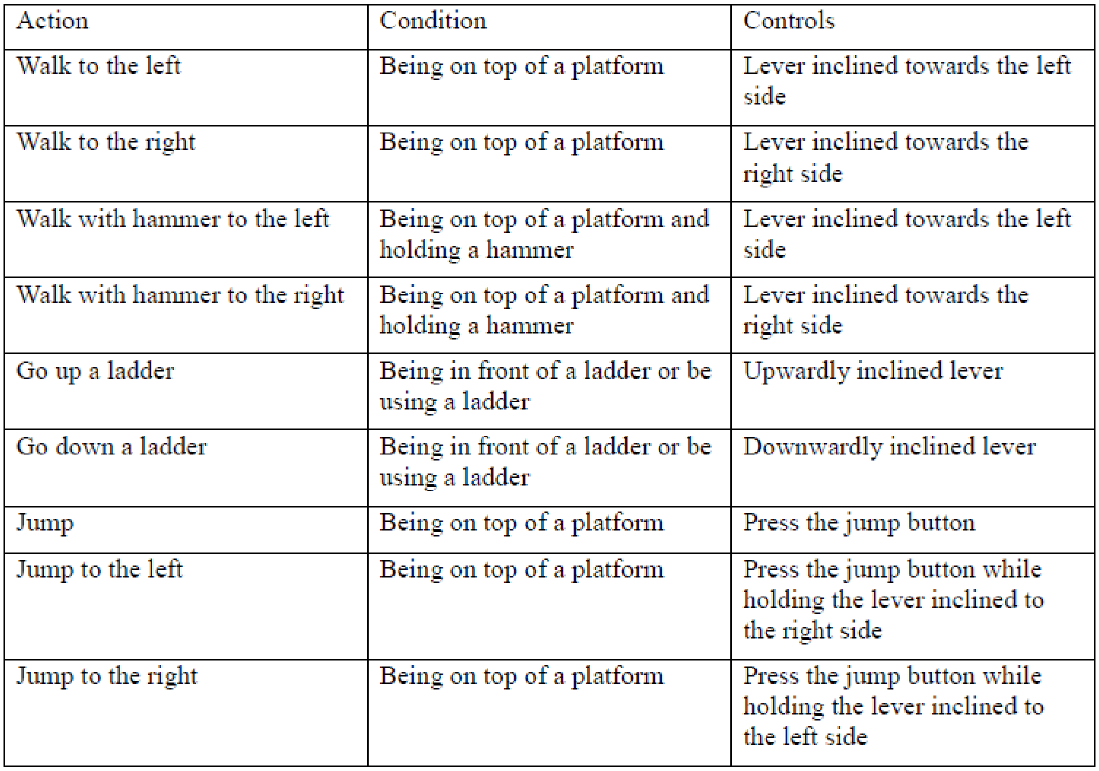
“Home screen (without player interaction): Player Score is on the top left of the screen, the Highest Score is at the center top and the Current level is at the top right. For this screen current level is displayed as zero. In the middle of the screen the words Donkey Kong are displayed. Under the name of the game Donkey Kong character appears."

The “without player interaction" refers to the fact that the player can't take action while this screen is shown. The next description is from the level screen where the player can interact:

“Levels (with interaction): At each level the player can execute Mario’s pre-defined movements. Every time that Mario makes a move that generates points, this score is reflected in the upper left of the screen, adding each point generated by Mario to the score. In case the score generated by Mario is higher that the score found in the top center of the screen, and this score will be updated as the player’s score with the difference that when a new game begins this score will not restart. At the top right of the screen each time the player loses, a figure that represents attempts will disappear. If there are no more figures the player loses and the game ends. Below where the current level and the attempts are shown, for each level there is a box that represents the time that Mario has to finish the level and also represents the extra points that will be added to the player’s score at the end of the level."

To describe how the player interacts with the hardware of the game we have created Table 3-7, which shows the possible actions of the player, the conditions needed to execute the actions and how to physically perform these actions.

Table 3-7 Control interface



Action	Condition	Controls
Walk to the left	Being on top of a platform	Lever inclined towards the left side
Walk to the right	Being on top of a platform	Lever inclined towards the right side
Walk with hammer to the left	Being on top of a platform and holding a hammer	Lever inclined towards the left side
Walk with hammer to the right	Being on top of a platform and holding a hammer	Lever inclined towards the right side
Go up a ladder	Being in front of a ladder or be using a ladder	Upwardly inclined lever
Go down a ladder	Being in front of a ladder or be using a ladder	Downwardly inclined lever
Jump	Being on top of a platform	Press the jump button
Jump to the left	Being on top of a platform	Press the jump button while holding the lever inclined to the right side
Jump to the right	Being on top of a platform	Press the jump button while holding the lever inclined to the left side

To make the game fun it should be balanced. In game design we can identify key aspects that can help to quickly balance the game. These aspects should be pointed out so when the game is in production the programmer can take into consideration which aspects of the game should be easy to modify. The following text shows the key aspects to balance level 1 in the game:

“Level 1:

- Frequency at which the barrels are thrown
- Barrel speed
- Number of blue barrels
- Number of hammers on the level
- Number of ladders
- Number of broken ladders
- Time to finish the level"

Finally, to finish the game dynamics we need to focus on how the player will learn to play our game. In the case of Donkey Kong the game instructions were in the same arcade of the game. The next description shows how we describe the process of the player learning the game:

“The game is intuitive, therefore the lever in the arcade of the game must state what action should be performed depending on the tilt, and in the same way the jump button should indicate its use. The player will discover the rest of the movements by trial and error. Another way to discover the move that Mario can do is to show a demo of the first level, where the player can see Mario executing the main actions."

3.4.4 Example Conclusions

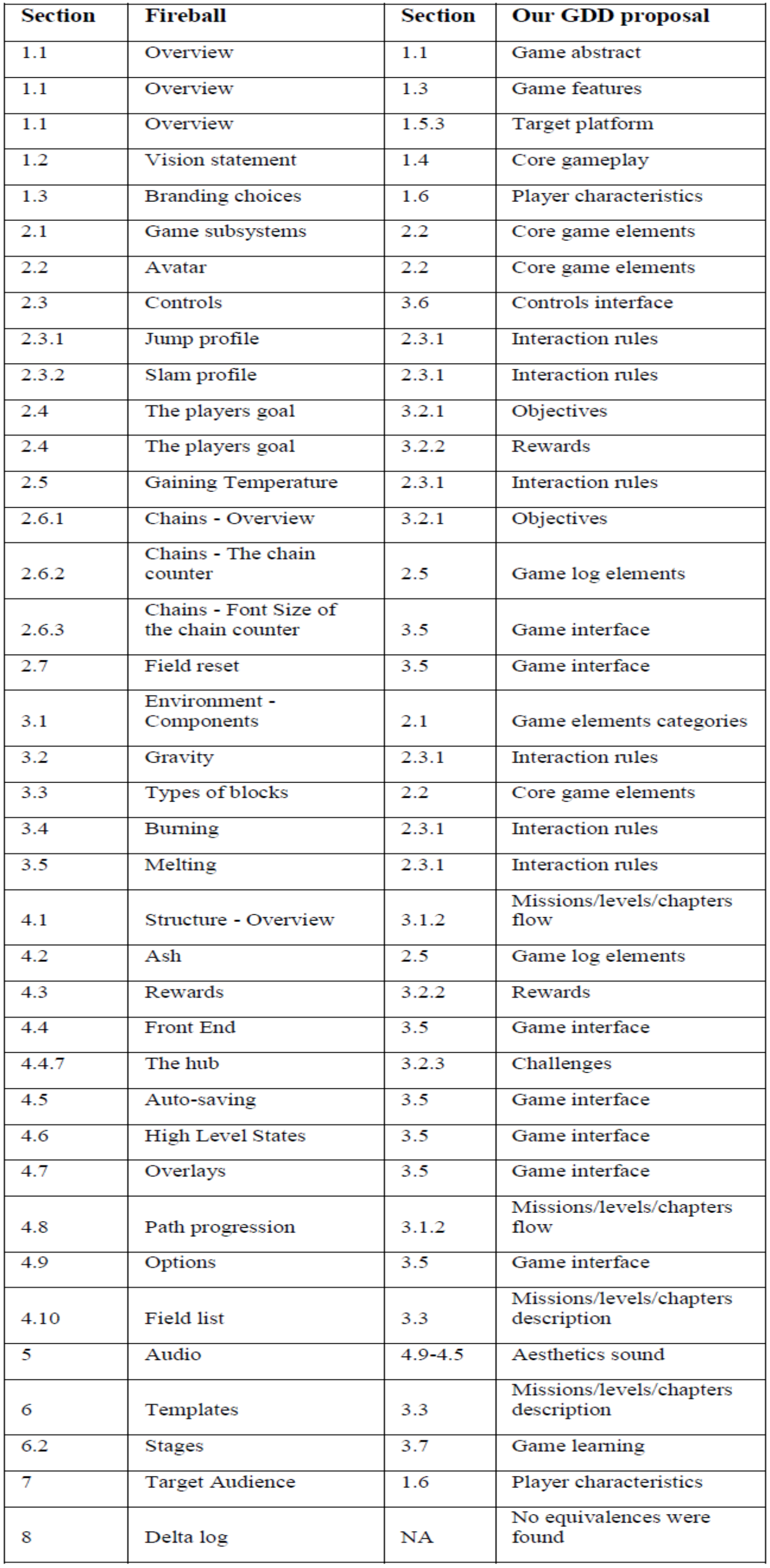
The example illustrates how a game can be described in natural language within the context of application of our improved GDD. Not all the games will be described in the same way, but the example helps to give the main idea on how to use our proposal.

3.5 Discussion

We obtained the strengths and deficiencies of our GDD by comparing it with a GDD example taken from International Hobo´s company, the GDD of the Fireball video game (Down, 2007). We used the Fireball GDD since it is one of the few well-known commercial GDDs available and published on Gamasutra. The purpose of this comparative analysis is to verify for completeness and to identify the advantages and deficiencies of our proposal.

The contents in the sections of the Fireball GDD were analyzed to see if they fit in any section of our GDD. Table 3-3 shows the sections in the Fireball GDD along with the equivalent section in our GDD proposal.

Table 3-8 Improved GDD - Fireball Comparison



Section	Fireball	Section	Our GDD proposal
1.1	Overview	1.1	Game abstract
1.1	Overview	1.3	Game features
1.1	Overview	1.5.3	Target platform
1.2	Vision statement	1.4	Core gameplay
1.3	Branding choices	1.6	Player characteristics
2.1	Game subsystems	2.2	Core game elements
2.2	Avatar	2.2	Core game elements
2.3	Controls	3.6	Controls interface
2.3.1	Jump profile	2.3.1	Interaction rules
2.3.2	Slam profile	2.3.1	Interaction rules
2.4	The players goal	3.2.1	Objectives
2.4	The players goal	3.2.2	Rewards
2.5	Gaining Temperature	2.3.1	Interaction rules
2.6.1	Chains - Overview	3.2.1	Objectives
2.6.2	Chains - The chain counter	2.5	Game log elements
2.6.3	Chains - Font Size of the chain counter	3.5	Game interface
2.7	Field reset	3.5	Game interface
3.1	Environment - Components	2.1	Game elements categories
3.2	Gravity	2.3.1	Interaction rules
3.3	Types of blocks	2.2	Core game elements
3.4	Burning	2.3.1	Interaction rules
3.5	Melting	2.3.1	Interaction rules
4.1	Structure - Overview	3.1.2	Missions/levels/chapters flow
4.2	Ash	2.5	Game log elements
4.3	Rewards	3.2.2	Rewards
4.4	Front End	3.5	Game interface
4.4.7	The hub	3.2.3	Challenges
4.5	Auto-saving	3.5	Game interface
4.6	High Level States	3.5	Game interface
4.7	Overlays	3.5	Game interface
4.8	Path progression	3.1.2	Missions/levels/chapters flow
4.9	Options	3.5	Game interface
4.10	Field list	3.3	Missions/levels/chapters description
5	Audio	4.9-4.5	Aesthetics sound
6	Templates	3.3	Missions/levels/chapters description
6.2	Stages	3.7	Game learning
7	Target Audience	1.6	Player characteristics
8	Delta log	NA	No equivalences were found

With respect completeness, almost all sections of Fireball and their descriptions were alike with our GDD. The only exception was the version control, titled Delta Log, which was not included in our GDD.

With regard to deficiencies, we identified several areas of improvement for our proposal. One is to include a section to describe an example when the game mechanics are complicated or difficult to understand. Another area of improvement is to include a guiding section on how to create game missions, levels or chapters. And the last improvement would be versions control to manage changes in a video game's stages.

As for the strengths and advantages of our proposal, we found that a part of our proposal had no equivalence in the GDD of Fireball. Although useful parts such as game objectives, game justification, initial scope, game balance, experience, constraints and assumptions, and document information are important for pre-production, production and post-production stages, details such as scope, constraints, assumptions, or goals of the game are not treated in the Fireball GDD even though these have a direct influence on the game design decisions. However, since balance and experience are not mentioned in the Fireball GDD, so in future stages such as implementation, we could not measure these experiences or balance the game easily.

Finally we find that some Fireball GDD parts are scattered and not classified correctly, making it difficult to track parts, such as game interface, rewards or game interaction rules between elements. This may hinder the usefulness of the document especially in the implementation stage.

3.6 Summary

We analyzed various GDD templates to obtain an initial structure of a GDD, and then we presented a SRS standard template identifying its structure, relations and details as a means to improve our GDD. Finally we conducted a comparative analysis of our GDD proposal with a commercial GDD. The purpose of this analysis was to find the degree of completeness, lacks and benefits of our proposal for production and post-production stages.

Designing a video game with our GDD can offer:

- Traditional software development uses SRS to go from requirements specification to software design as standard practice. In a similar fashion our GDD will benefit game developers by supporting formality, detail and stronger relations between pre-production and production stages.

- SRS improvements such as: common understanding, quality assurance, decision-making, relations, boundaries, limitations and knowledge of game parts.
- A better grasp of completeness, game objectives, game justification, initial scope, game balance, experience, constraints and assumptions, and document information.

Our proposal identifies relevant issues that need to be addressed such as:

- GDD elaboration requires certain skills regarding SRS experience and training.
- GDD formalization may hinder creativity.
- A need to adopt Version Control Management techniques and tools
- The possible necessity of including explicit examples of mechanics.
- The possible necessity of a guide to create game levels.

For future work we plan to develop a tool to support the creation and validation of the proposed GDD.

CHAPTER 4
HANDLING PLAYER'S EXPERIENCE IN VIDEO GAME DEVELOPMENT

4.1 Problem Description

A main topic in video games development processes is to guarantee an optimum level of interactive experience (Schell, 2008). The experience is commonly evaluated in the final stage of the development process (*Evaluating User Experience in Games: Concepts and Methods (Human-Computer Interaction Series)*, 2010), but it is advisable to relate the experiences to the game goals, and align the game features to the desired experience early in the game development process. Managing experience in early stages can help to avoid unexpected results such as unfulfilled goals that may lead to wasting of resources on features that may not be meaningful to the player.

Establishing player experience drivers, and creating guidelines to implement them that can be traced to the game elements in a Game Design Document (GDD) to enhance the quality of final experience during the full development process, may help to minimize unexpected results.

4.2 Solution Approach

User experience is a key in game development, which is why understanding the relation between UX (user experience) and other relevant areas of game development may help to do a better job of handling the experience. Figure 4-1 shows the relation between UX and other areas. The following sub-sections briefly address each identified area and relationship.

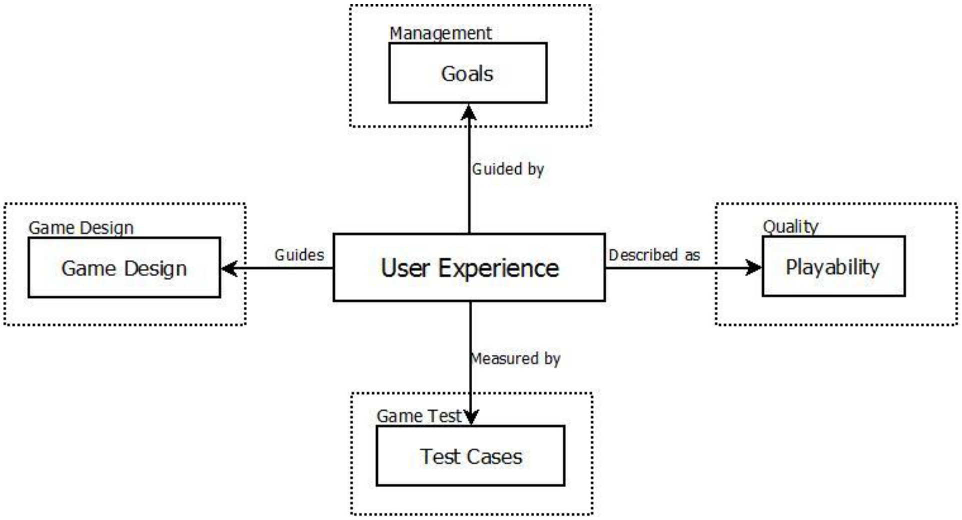


Figure 4-1 User Experience Relations

4.2.1 Management

A video game always has a purpose - to make money, to convey an idea, to train, to educate or to solve a problem. This purpose can be transformed into goals. A goal may be defined as "an objective the system under consideration should achieve. Goal formulations thus refer to intended properties to be ensured..."(Jackson, 1995). There is no explicit proposal to identify goals and their quality attributes for video games. But in traditional software, requirements engineering builds on the goals to derive system requirements, and relevant quality attributes may be obtained from these goals (Jackson, 1995; van Lamsweerde, 2001; Wiegers, 2003).

In games, one of the first things to address is defining the desired experience that the game will bring to the player (Schell, 2008). User experience can be defined as "a person’s perceptions and responses that result from the use and/or anticipated use of a product, system or service"(ISO/TC 159/SC, 2010). In video games UX becomes more complex due its ludic nature. It is important to clarify that all games generate an experience, but that experience may not be steered by goals. We point out, that due to a lack of alignment between UX and quality attributes with goals, a game may not satisfy the expected goals. Therefore the UX as well as quality attributes need to be guided and aligned to the goals in order to achieve these goals.

4.2.2 Game Design

Game design is a key in creating the desired UX. Experience may be given when game elements interact with the player, and the interaction is defined in the game design. Many authors mention that such interaction is an important open issue during game design (Bates, 2004; Rollings & Adams, 2003; Schell, 2008).

Game heuristics have been used to evaluate different aspects of game design (Desurvire et al., 2004; Desurvire & Wiberg, 2009; Korhonen & Koivisto, 2006; Korhonen et al., 2009; Pinelle et al., 2008a). A heuristic is "a design guideline which serves as a useful evaluation tool" (Desurvire et al., 2004). Heuristics may be a useful tool to review general aspects of game design, but they are not tailored to a specific game, which means that they may not be applicable on some games or they can even hinder the design of a game.

Having a way to relate the intended UX to the game elements can be useful in measuring and tracking experience. A GDD that classifies game elements in an organized and structured manner may be used to explicitly relate the intended UX to game elements. Therefore, a mechanism is needed to connect GDD elements with experiences or it will be a difficult task to track the expected experiences to the game design.

4.2.3 Game Testing

In game testing, UX evaluation is a challenging activity. Recently, there are some works on evaluating game UX (*Evaluating User Experience in Games: Concepts and Methods (Human-Computer Interaction Series)*, 2010). Although, these efforts provide some guiding light to measure the experience, most of them require a fully operational video game or a working prototype in late stages to be able to measure the UX. Finding the UX in late stages can prevent a game that brings unwanted experiences to be released, but fixing these unwanted experiences in late stages is more costly than dealing with them in early ones.

A technique used in software engineering to find and fix bugs in games is a test case. A test case is "a set of inputs, execution preconditions, and expected outcomes developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement" (IEEE, 1990). Test cases can be used to verify whether or not a game satisfies a quality attribute.

Having the experiences and game elements connected from game design makes it possible to validate the alignment of the game elements with experiences. Test cases can be extended as a tool to evaluate this alignment in prototypes or the testing stage of the game.

A broad range of metrics can evaluate user experience, and Game Experience Management (GEM) can be integrated with different tools and methods that use these metrics. The main metrics used to evaluate UX are the following:

- In game metrics: these metrics represent actions of the player in the game (how many tries it took to beat a challenge, percentage of the world discovered, hidden items found, etc.). These actions can be interpreted to figure out what was the experience of the player when playing the game.
- Questionnaire: these metrics are obtained directly from players of the game by asking them question or opinions about statements. The questions are directly related with the experience of the player while playing the game.
- Eye tracking: these metrics are obtained by following the eye of the player while playing the game. An electronic devise follows the focus of the eye on the screen where the game is being played; these metrics are often used in conjunction with other UX metrics.
- Biometrics- Face recognition: these metrics are similar to eye tracking, but instead of recording the eye movement they record the face of the player while playing the game. Algorithms that analyze the face of the player and associate it to the part of the game played can interpret experience of the player.
- Biometrics- Voice recognition: these metrics are similar to eye tracking and face recognition, but here the voice of the player is recorded while playing the game. Algorithms that analyze the voice of the player and associate it to the part of the game played can interpret the experience of the player.
- Biometrics- Electroencephalography: these metrics are similar to eye tracking, face and voice recognition, but here the brain waves of the player are recorded while playing the game. Algorithms that analyze the voice of the player and associate it to the part of the game played can interpret the experience of the player.

4.2.4 Quality

The definition of playability has not been definitively settle and is currently under discussion, but for the purpose of this work we will define playability as "a set of properties that describe the player experience using a specific game system whose main objective is to provide enjoyment and entertainment..."(J. González, Zea, & Gutiérrez, 2009). The use of playability as a way of describing the user experience allows the breakdown of an experience in properties that can be measured and validated.

As mentioned on Section 4.2.3, there are several studies dealing with the evaluation of user experience in video games (*Evaluating User Experience in Games: Concepts and Methods (Human-Computer Interaction Series)*, 2010), but they do not establish a desired experience as a starting point. While evaluating user experience is important, knowing the desired experience before evaluating it, may help prioritize relevant properties of the playability. This desired experience will provide a baseline against which to compare the results obtained when evaluating playability. This means not only knowing if the measured playability is positive or negative, but also providing a means of determining whether or not the desired playability satisfies the established objectives.

4.3 Solution Description

In a previous work, we proposed a GDD (Gonzalez et al., 2012) as a way of adding formality to game design. But also we identified the need for a methodology that can handle the UX through the whole game development process. Therefore, we have taken on the challenge to elaborate such methodology for managing, tracking and measuring the UX. The proposal uses the goals and concept of the game to characterize the desired experience and how it is carried out through the whole development process and its game elements.

Figure 4-2 shows the flow and activities of the Game Experience Management (GEM) methodology. Our proposal is based on the Quality Attribute Workshop (QAW) proposal (Lattanze, Stafford, & Weinstock, 2003), it was adapted and extended to accommodate the video game context for managing, tracing and measuring the UX. QAW fulfills our requirements due to its early identification of architectural drivers using business/mission context, high-level functional requirements, constraints, and quality attribute requirements. All these properties may be mapped to the video game domain using game design drivers to guide game design as architectural drivers guide software architecture design. Next, we describe each of the activities within our GEM proposal.

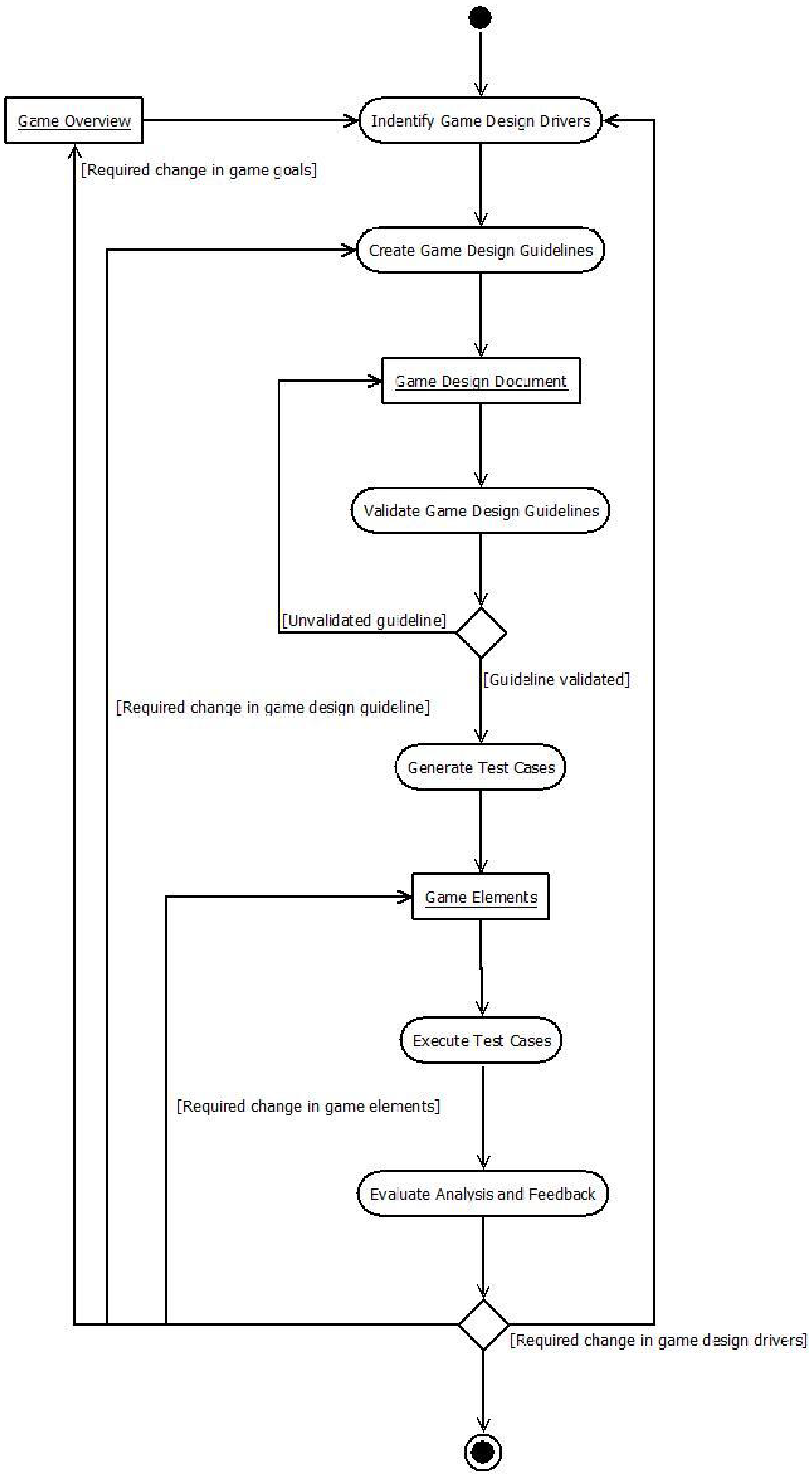

Game Overview
Indentify Game Design Drivers
[Required change in game goals]
Create Game Design Guidelines
Game Design Document
Validate Game Design Guidelines
[Unvalidated guideline]
[Guideline validated]
[Required change in game design guideline]
Generate Test Cases
Game Elements
Execute Test Cases
[Required change in game elements]
Evaluate Analysis and Feedback
[Required change in game design drivers]

Figure 4-2 GEM Activity Diagram

4.3.1 Identify Game Design Drivers

The first activity is focused on finding out drivers that should guide the game design. The activity comprises the following steps:

1) Take care of the GEM methodology presentation and introductions, where the motivation and methodology is explained
2) An overview with wherein topics like game objectives, game concept, target audience, game features (number of players, genre, target platforms, game theme, among others), initial constraints or core gameplay are presented. The gathered information sets the baseline to create an initial list of possible game design drivers. Game design drivers are high-level properties that the game should have in order to generate the intended experience.
3) Identify game design drivers, where the game design drivers' list is consolidated by voting in favor of each one. Prioritization of the guidelines is accomplished by allocating to each stakeholder a number of votes. Voting is done in round-robin fashion, in two passes. By the end of the first pass all stakeholders will have placed half of his or her votes. For the second pass all stakeholder will distribute the rest of his or her votes. The votes are counted, and the guidelines are prioritized accordingly. The game design drivers in the final list can be related to a UX evaluation proposal for the sake of a better evaluation on further stages in game development. We used the proposal of González, Montero and Padilla (J. L. González et al., 2009) as the base UX evaluation proposal in our method, but other similar evaluation approaches may be used.

4.3.2 Create Game Design Guidelines

The second activity focuses on creating guidelines that can be related to specific game design elements. A game design guideline is a description of how game elements need to be created in order to achieve the intended experience established in the game design drivers. A guideline is similar to a heuristic, which is created to have a general application in a context; however the people involved specifically for the game to be developed create a guideline. The use of guidelines allows creativity and innovation; however the use of heuristics to guide the creation of game elements may limit both. We are convinced of the usefulness of heuristics, as a validation instrument on elements already created, but not guiding its development. A Game Design Document (GDD) is definitely required in this activity to relate its elements with the guidelines (Gonzalez et al., 2012). The activity comprises the following steps:

1) Use brainstorming to define guidelines, where an initial guideline list is created and there is at least one guideline for each game design driver.
2) Consolidate guidelines, where similar guidelines could be merged and guidelines in conflict may be rewritten to avoid conflict.
3) Prioritize guidelines, where a guideline voting mechanism is used to determine which guidelines have precedence.
4) Identify relationships between guidelines and game elements, associating guidelines to game elements.

4.3.3 Validate Game Design Guidelines

This activity has only one step. Whenever a game element is finished, the validation step is executed in order to verify the fulfillment of the guidelines associated to such element. If a guideline is not followed, there should be a justification explaining the reason. If the justification is rejected or there is no justification, the element is not approved. Once all elements related to a guideline are approved, the guideline is considered valid.

4.3.4 Generate Test Cases

This activity will generate different sets of test cases. The activity comprises the following steps:

1) Create test cases that are focused on evaluating the guidelines in terms of fulfillment of their goals. There should be at least one test case for each guideline.
2) Create test cases that are focused on evaluating whether or not the properties of the game described in the game design drivers are achieved. There should be at least one test case for each game design driver.
3) Create test cases that are focused on evaluating the overall UX generated by the game.

It should be noted that these test cases are not devised to find game bugs. We are expanding the use of test cases to allow the set of entries be broad enough to include a questionnaire or part of it; biometrics and its relation to parts of the game; a video of facial expressions of the players as they play and their interpretation; or metrics collected by the game while playing. If the game design drivers were related to a user experience evaluation proposal, the evaluation tool of the proposal may be used as a substitute or complement of the test cases proposed.

4.3.5 Execute Test Cases

This activity has only one step that consists of the execution of each test case. As we explained earlier, our methodology extends the use of test cases, which means that the execution and expected result of each test case may be very broad. The execution may be: applying questionnaires to a focus group that plays only some sections of the game; or having the game collect some metrics while testers play some parts of the game. The expected result may be: a specific output of a questionnaire, or a minimum number in a specific metric collected while the testers were playing a specific part of the game. This activity has no specific phase to be executed; it will depend on the test cases' specifications. The test cases can be executed in pre-production, production or post-production.

4.3.6 Evaluation, Analysis and Feedback

This activity consists in analyzing the test cases results for possible adjustments in the game. The activity has the following steps:

1) Analyze if the guidelines have achieved their goals.
2) Analyze if the game design drivers brought the intended properties (experience) to the game.
3) Analyze the overall experience that the game brought to the player.
4) If required, modify the game elements, guidelines, game design drivers or even change the goals.

4.4 GEM and improved GDD

In this section we describe the model of our proposed GEM and how it can interact with our iGDD.

4.4.1 GEM Model

GEM proposal has three main concepts: the game design driver, the game design guideline and the test cases. The game design drivers represent properties that the game should have to obtain the intended UX. The game design guideline is a guide that describes how to design specific game elements. The test cases are used to measure the level of satisfaction of the guideline, drivers and the overall experience that the game brings. Figure 4-4 shows the GEM model that describes the relation between these main concepts. The User Experience Evaluation Tool class refers to external evaluation tools that can be integrated with the test cases to evaluate UX. These evaluation tools can have broad kinds of metrics like the ones mentioned in Section 4.2.3.

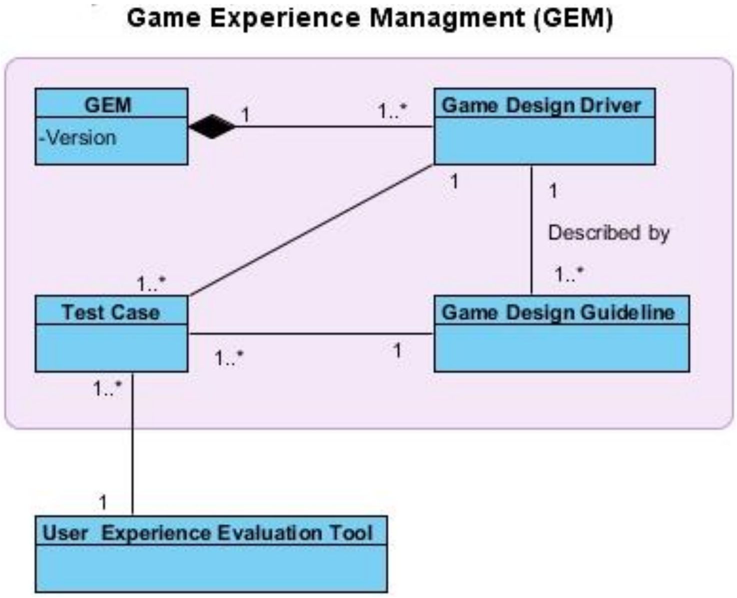


Figure 4-3 GEM Model

4.4.2 GEM relation with iGDD

The GEM proposal needs to be related to a GDD in order to be able to trace the elements that bring the experience to the player and to know if they are achieving their goals. Figure 4-4 shows the relation between the GEM and the iGDD. The game concept and the game goals (which are described in the Overview class in the iGDD) are used to establish the UX that the player should have with the game. This UX is described in high-level properties (Game Design Drivers class in the GEM) that the game should have. For each high level property, one or more guides on how to create specific game elements (Game Design Guideline class in the GEM) are created. These guidelines must be specific as to what game elements (Mechanics Game Elements, Game Modality Elements and Gameplay Interface classes on the iGDD) they affect. This allows tracing of the UX to each specific game element that is intended to produce it. To prove that the guidelines, the drivers and the UX are satisfying we create test cases (Test Case class in the GEM). These test cases can be related with different UX evaluating tool (User Experience Evaluation Tool class) for video games to facilitate the EX evaluation. The test cases have the same relation with the iGDD as the guidelines, the main difference being that the guidelines dictate how the game elements should be created and the test cases evaluate if the guideline, driver and UX in general achieve its goals.

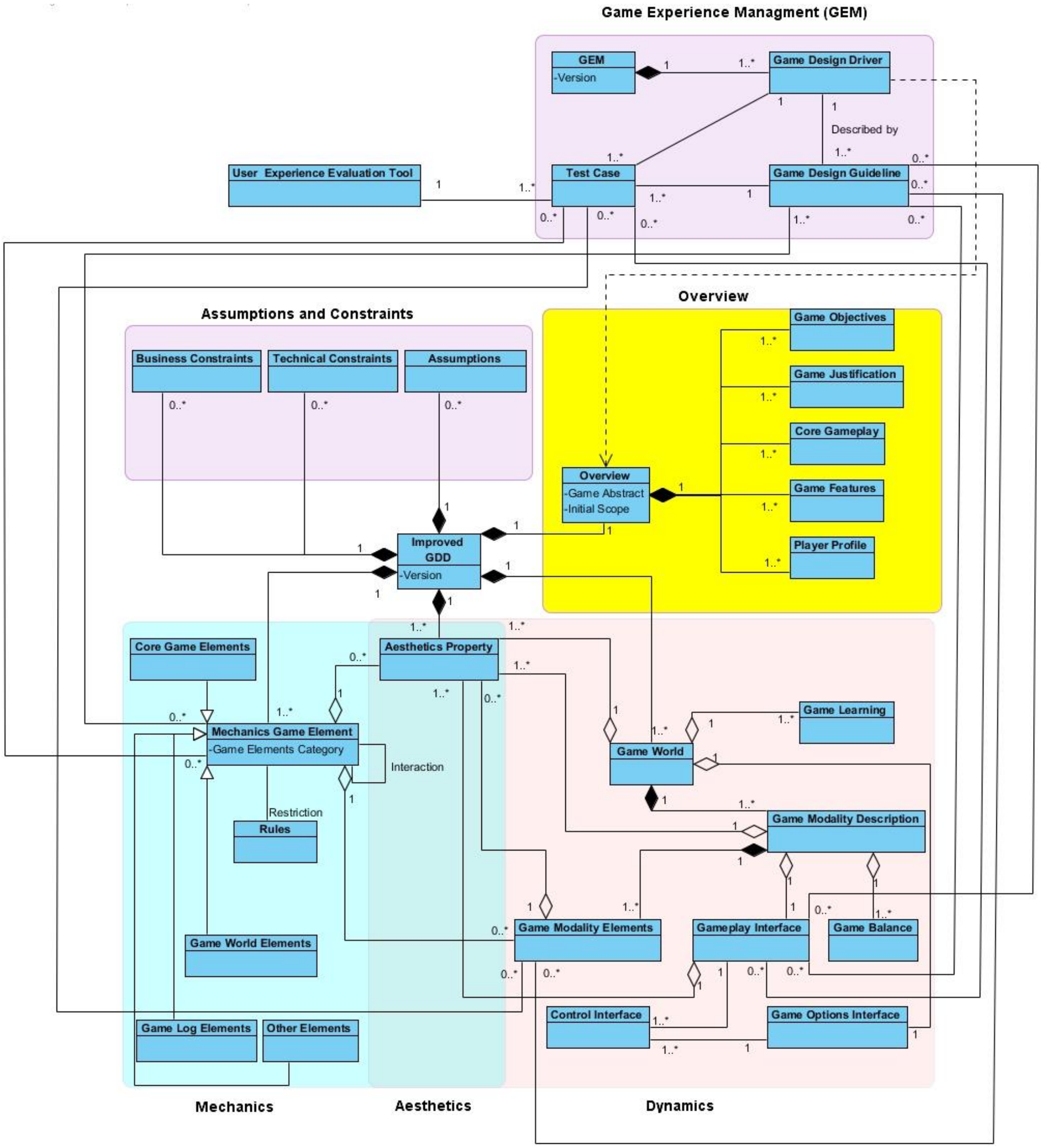


Figure 4-4 GEM and iGDD Models Relation

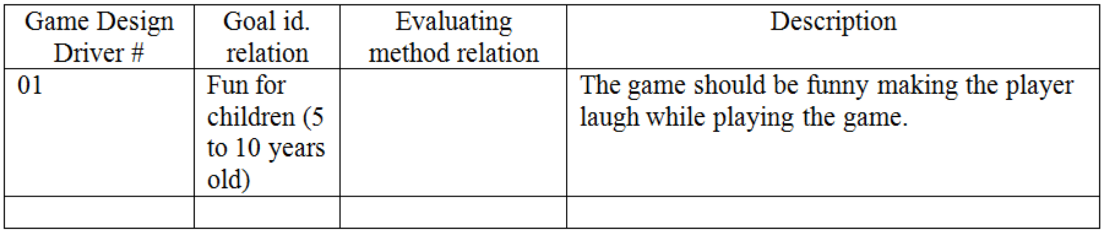
4.5 *Example*

In this section we describe an example of our GEM to illustrate how the player experience can be created, traced and measured. We use a hypothetical example to describe the GEM main concepts. We cover some illustrative samples of each section and give our conclusions.

4.5.1 Example Game Design Drivers and Guidelines

The game design drivers will guide the design to achieve the desired player experience. In this example we are creating a game where the premise of the game is to rescue your beloved pet from a circus that kidnapped it. To obtain the game design drivers for this game we need to review the overview of the game. We mainly focus on creating players' experiences that help us to achieve the game goals, but it is important to review other game characteristics, such as target platform, and player profile. The main goal of our game is "fun for children of ages between five and ten years old". Table 4-1 shows the first driver derived from this goal.

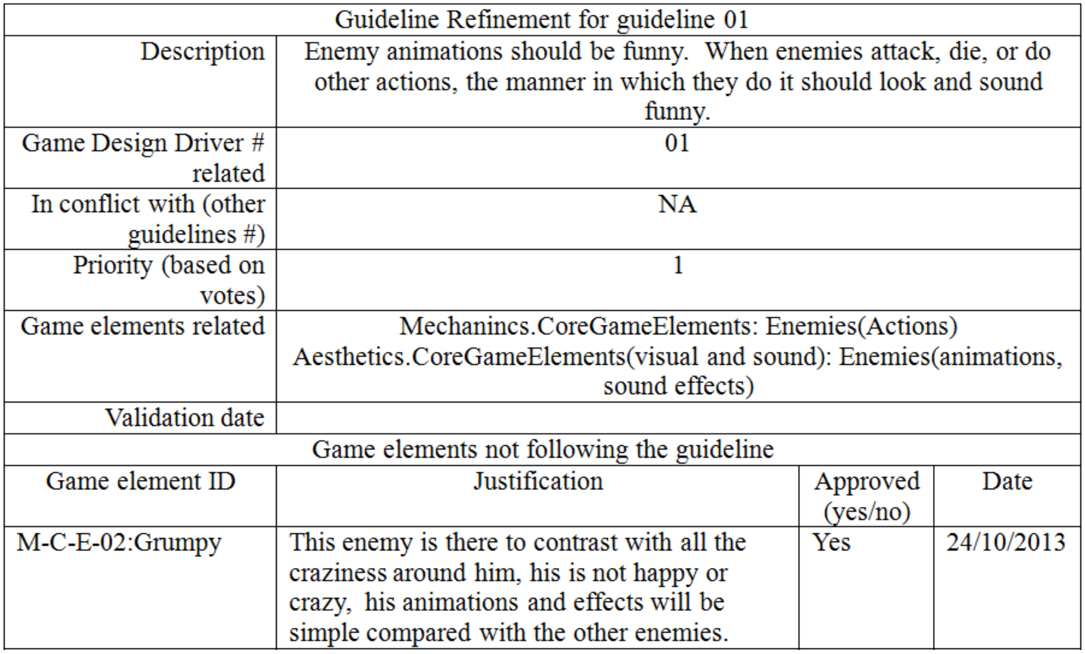
Table 4-1 Game Design Driver Table



Game Design Driver #	Goal id. relation	Evaluating method relation	Description
01	Fun for children (5 to 10 years old)		The game should be funny making the player laugh while playing the game.

In Table 4-1 we can see that the first game design driver of the game is to be funny. This may not be a clear property, but the description that the driver should make the players laugh helps in understanding what the property is about. To give the game this property we will create specific guidelines associated to specific game elements that will bring the desired property. One sample guideline created to bring the first game design driver to the game is shown in Table 4-2. The guideline in Table 4-2 is one of many guidelines that can be created to give a game the property of a certain driver.

Table 4-2 Game Design Guideline



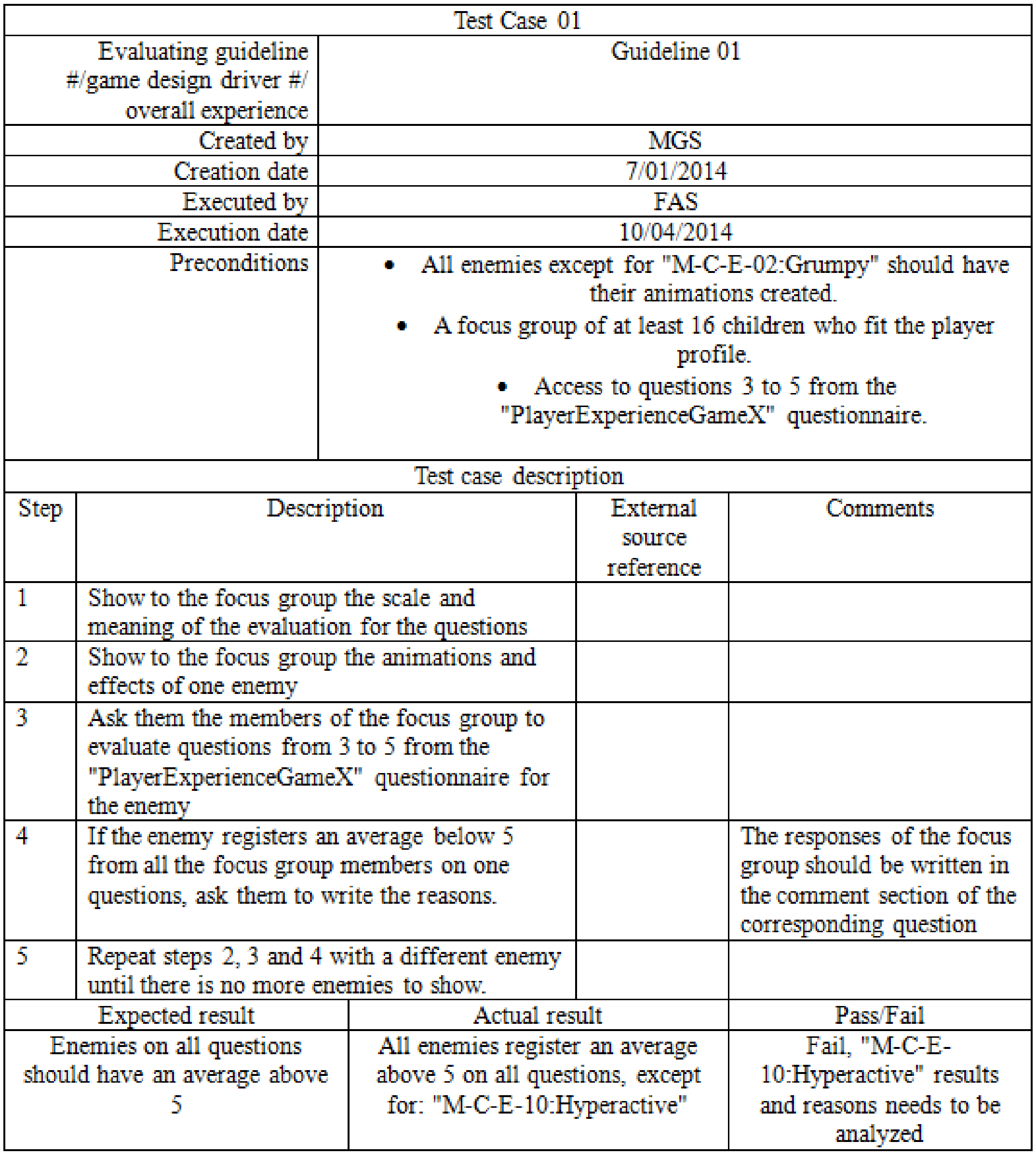
Guideline Refinement for guideline 01			
Description	Enemy animations should be funny. When enemies attack, die, or do other actions, the manner in which they do it should look and sound funny.		
Game Design Driver # related	01		
In conflict with (other guidelines #)	NA		
Priority (based on votes)	1		
Game elements related	Mechanincs.CoreGameElements: Enemies(Actions) Aesthetics.CoreGameElements(visual and sound): Enemies(animations, sound effects)		
Validation date			
Game elements not following the guideline			
Game element ID	Justification	Approved (yes/no)	Date
M-C-E-02:Grumpy	This enemy is there to contrast with all the craziness around him, his is not happy or crazy, his animations and effects will be simple compared with the other enemies.	Yes	24/10/2013

In the example of Table 4-2 we can observe that the guideline describes how the associated game elements should be created, in order to help obtain the desired driver. In the description we see that all enemies in the game are related to this guideline and the details of the GDD relation is described in the "Game elements related" section, for this example the guideline is related to the mechanics and aesthetics (sound and visual) elements of the enemy category. To validate the guideline, all elements related should follow the guideline or should have an approved justification as to why the specific element does not follow the guideline. In Table 4-2 there is an example of a game element that does not follow the guideline. There is a justification on why the enemy "Grumpy" does not follow the guideline, the justification has been reviewed and approved, which means the guideline can be validated even when "Grumpy" does not follow the guideline.

4.5.2 Example Test Cases Creations and Execution

The test cases can be used to validate the player experience at different levels as mentioned previously. The test case can be associated to different evaluation tools, depending on each case. For this example we created a test case to evaluate the guideline that has been shown in the previous section. The test case is related to a part of a questionnaire, which will be used to evaluate different parts of the intended player experience. Table 4-3 the test case description.

Table 4-3 Test Case Description



<table>
<tr><td colspan="6">Test Case 01</td></tr>
<tr><td colspan="2">Evaluating guideline #/game design driver #/ overall experience</td><td colspan="4">Guideline 01</td></tr>
<tr><td colspan="2">Created by</td><td colspan="4">MGS</td></tr>
<tr><td colspan="2">Creation date</td><td colspan="4">7/01/2014</td></tr>
<tr><td colspan="2">Executed by</td><td colspan="4">FAS</td></tr>
<tr><td colspan="2">Execution date</td><td colspan="4">10/04/2014</td></tr>
<tr><td colspan="2">Preconditions</td><td colspan="4">• All enemies except for "M-C-E-02:Grumpy" should have their animations created.
• A focus group of at least 16 children who fit the player profile.
• Access to questions 3 to 5 from the "PlayerExperienceGameX" questionnaire.</td></tr>
<tr><td colspan="6">Test case description</td></tr>
<tr><td>Step</td><td colspan="3">Description</td><td>External source reference</td><td>Comments</td></tr>
<tr><td>1</td><td colspan="3">Show to the focus group the scale and meaning of the evaluation for the questions</td><td></td><td></td></tr>
<tr><td>2</td><td colspan="3">Show to the focus group the animations and effects of one enemy</td><td></td><td></td></tr>
<tr><td>3</td><td colspan="3">Ask them the members of the focus group to evaluate questions from 3 to 5 from the "PlayerExperienceGameX" questionnaire for the enemy</td><td></td><td></td></tr>
<tr><td>4</td><td colspan="3">If the enemy registers an average below 5 from all the focus group members on one questions, ask them to write the reasons.</td><td></td><td>The responses of the focus group should be written in the comment section of the corresponding question</td></tr>
<tr><td>5</td><td colspan="3">Repeat steps 2, 3 and 4 with a different enemy until there is no more enemies to show.</td><td></td><td></td></tr>
<tr><td colspan="3">Expected result</td><td colspan="2">Actual result</td><td>Pass/Fail</td></tr>
<tr><td colspan="3">Enemies on all questions should have an average above 5</td><td colspan="2">All enemies register an average above 5 on all questions, except for: "M-C-E-10:Hyperactive"</td><td>Fail, "M-C-E-10:Hyperactive" results and reasons needs to be analyzed</td></tr>
</table>

The test case in Table 4-3 shows how the test case is related the guideline 01. It describes the preconditions needed so the test case can be executed, which in this case, aside from completing the enemies for the game, a focus group and certain sections of a questionnaire are needed. The steps description on the test cases shows how the test should be executed. For this case there were three questions asked for each enemy. Finally, the document indicates the expected result compared to the actual result. The test case results in a fail because not all enemies were ranked with an average over five. Since the test case failed, corrective actions on the evaluated item needs to be taken.

4.5.3 Example Evaluation, Analysis and Feedback

The test case example in the previews section fail, which means that we need to do an analysis on why the test failed. The test case register the reasons of why the children give a grade below five, so we review the reasons for grading "Hyperactive" enemy low in the three questions asked. We find out that the children thinks that the enemy is scary, because it has big red eyes, and he is shaking all the time, also the screams of the enemy sound like if he was crazy. Once the reasons for the test case fail have been studied, we can take actions to correct the game, it can be replacing the enemy with other less scary, or modify the enemy with animation and sounds less scary.

4.5.4 Example Conclusion

We cover a partial example of how the GEM can help to guide the game design to achieve the desired experience. We analyze a hypothetical case, where humor was one game design driver. We create a game design guideline to help to achieve humor in the game. We test how well this guideline accomplishes its objective, and work out how corrective actions can be taken when the objective is not met. This example uses a questionnaire applied to a focus group as the main evaluation tool, but this is just one of many other tools that can be applied. Other useful evaluation tools can be used in game data (e.g. number of fails per challenge) or the use of facial expression recognition software to track the players' emotions.

4.6 Discussion

Our proposal offers management, measurement and tracking of player's experience. A game development team that wants to put special emphasis on the UX may find our proposal useful. Our proposal may be used in a feedback-loop manner, first using it on an initial prototype and then using the evaluation analysis and feedback information to modify the game elements, guidelines, game design drivers or goals and so on. Our proposal may help to: align goals with UX, by taking the goals into account while defining the intended UX; trackback UX to game elements, by relating the guidelines to the game elements in a GDD; evaluate the UX, by using test cases which may be complemented with a UX evaluation methodology. We identified some shortcomings while creating our methodology: a) Keeping up to date relations may be a time consuming and difficult task, b) Elaboration of documentation may be overwhelming on agile development groups. We believe that a software tool supporting our methodology can help to overcome these limitations.

4.7 Summary

We analyzed the UX and its relevance in the video game context, as well as the concepts and areas with which it is related. We have presented a methodology for managing the playability from early stages of game development, in response to the need of playability contemplation throughout the development process. Finally, we presented the benefits, deficiencies and opportunity areas for our proposal. As future work, we will develop a software tool that helps to mitigate the mentioned inadequacies, integrate our methodology with agile methodologies such as Scrum (current trend used to create video games) and validate the methodology with a case study.

CHAPTER 5
PROPOSAL INTEGRATION WITH GAME DEVELOPMENT MODELS

5.1 Current Game Development Models

In section 2.4 we described game development models; we analyzed different approaches to game development and established two main categories in game development methodologies: the first is the traditional model, and the other the agile development model.

5.1.1 Traditional Models

Traditional models have well defined activities and roles with regard to create software. An adequate documentation and faithful following of the process is a key in the success of these methodologies. They require a significant effort to implement, but once the model is adopted, it offers the benefit of a better understanding of the project and the team. These models are better suited to projects where the uncertainty and changes are low.

Some advantages of traditional models are:

- Easy to understand.
- The software product is well documented.
- The more stable the project is easier to estimate and predict.
- Different indicators can be tracked to improve project and team areas.

Some drawbacks of traditional models are:

- New requirements greatly impact the project.
- Requirements and design errors detected in late stages can be hard and costly to fix.
- The flexibility to make changes in the project is limited even in incremental models.
- Communication between client and development team is not constant and may lead to misunderstandings.

In game development, traditional models are presently used. The waterfall model falls short for use in game development because of its lack of flexibility. But incremental models like the spiral model are better suited for creating video games because they allow for adjustments to the project in each cycle.

5.1.2 Agile Development Models

Agile development models are iterative models that focus on flexibility and give customer prioritized based software on each delivery. They are easy to adopt and instead of needing an exhaustive requirements elicitation and documentation, they work closely with the client and receive constant feedback. These models are better suited to situations where uncertainty and changes are high.

Some advantages of agile models are:

- Changes on the project are welcome and are easier to introduce than in traditional models.
- Focus on delivering business value to the customer.
- Frequent feedback from the costumer keeps the project aligned with expectations.
- Communication is direct and continuous

Some drawbacks of agile models are:

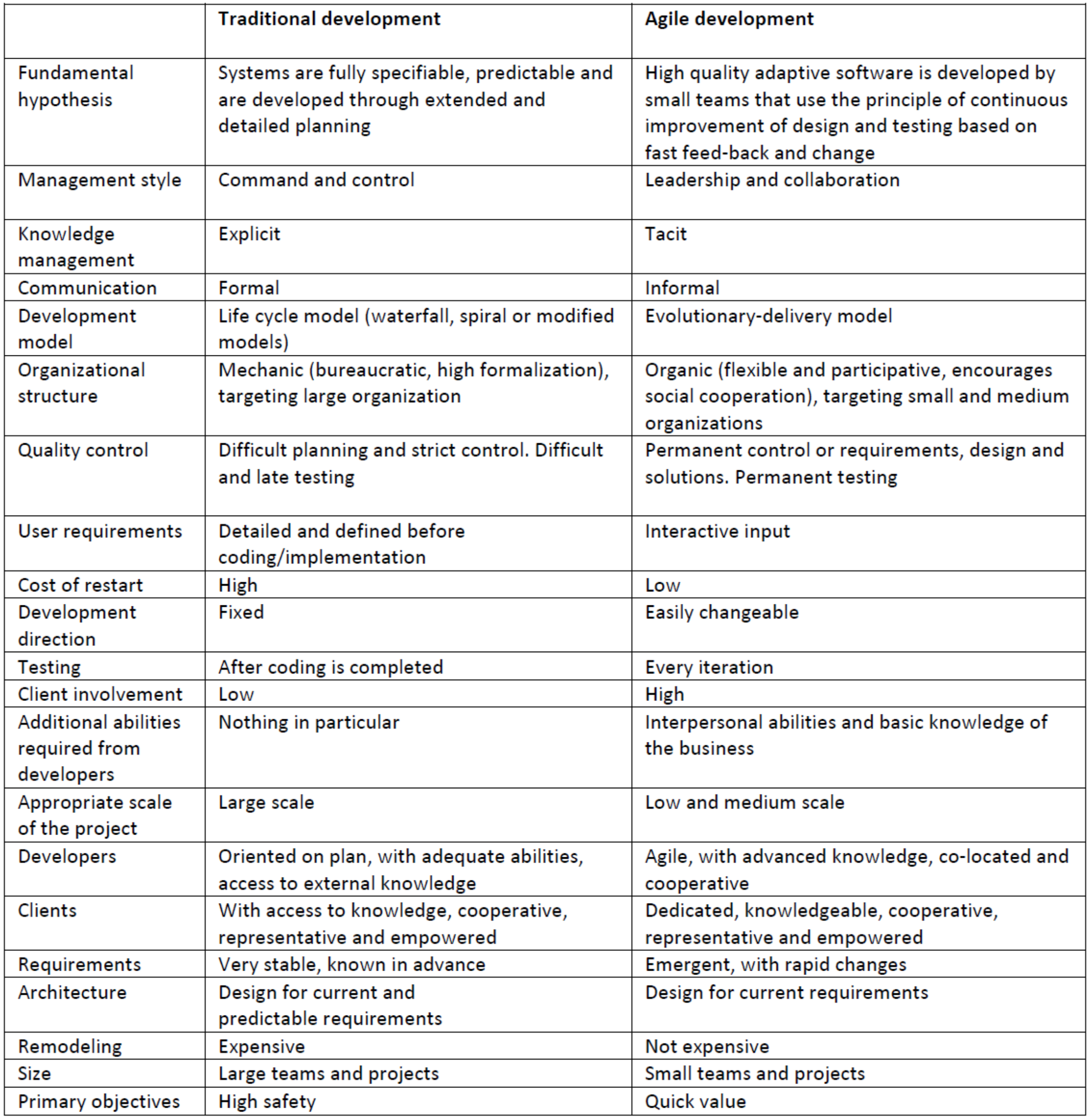
- Projects are hard to predict and estimate.
- Emerging requirements may make the project longer than expected.
- Difficult to negotiate contracts with the customer due the uncertainty of time and cost.
- When new personnel are integrated in the middle of a project, it is hard for them to catch up because of the lack of documentation.

Agile models have growth potential in game development companies. Since mobile and social media triggers the growth of indie game companies, the agile models are present in these companies (as seen on section 2.4). The flexibility of agile models is a good option for emerging companies that have a high degree of uncertainty on each project.

5.1.3 Game Development Methodologies Tendency

Traditional or agile models can do game development; they can even be combined in big projects by doing pre-production with an agile model and production and post-production with a traditional one. Which model is better for game development? It depends on each specific project. The sizes of the project, the expertise of the team, the technology to be used, are some criteria that can help to decide which model to use. Table 5.1 shows a comparison of different aspects of traditional and agile models.

Table 5-1 Traditional vs. Agile Development (Stoica, Mircea, & Ghilic-Micu, 2013)



	Traditional development	Agile development
Fundamental hypothesis	Systems are fully specifiable, predictable and are developed through extended and detailed planning	High quality adaptive software is developed by small teams that use the principle of continuous improvement of design and testing based on fast feed-back and change
Management style	Command and control	Leadership and collaboration
Knowledge management	Explicit	Tacit
Communication	Formal	Informal
Development model	Life cycle model (waterfall, spiral or modified models)	Evolutionary-delivery model
Organizational structure	Mechanic (bureaucratic, high formalization), targeting large organization	Organic (flexible and participative, encourages social cooperation), targeting small and medium organizations
Quality control	Difficult planning and strict control. Difficult and late testing	Permanent control or requirements, design and solutions. Permanent testing
User requirements	Detailed and defined before coding/implementation	Interactive input
Cost of restart	High	Low
Development direction	Fixed	Easily changeable
Testing	After coding is completed	Every iteration
Client involvement	Low	High
Additional abilities required from developers	Nothing in particular	Interpersonal abilities and basic knowledge of the business
Appropriate scale of the project	Large scale	Low and medium scale
Developers	Oriented on plan, with adequate abilities, access to external knowledge	Agile, with advanced knowledge, co-located and cooperative
Clients	With access to knowledge, cooperative, representative and empowered	Dedicated, knowledgeable, cooperative, representative and empowered
Requirements	Very stable, known in advance	Emergent, with rapid changes
Architecture	Design for current and predictable requirements	Design for current requirements
Remodeling	Expensive	Not expensive
Size	Large teams and projects	Small teams and projects
Primary objectives	High safety	Quick value

As we can see in Table 5.1, traditional models are better for big and stable projects and agile models are better for small and medium projects with some uncertainty.

Among the agile models, Scrum framework is one of the most used and adapted for game development. Section 5.2 will explain Scrum in more detail, why it is popular for game development, and how it is being adapted.

5.2 *SCRUM for Game Development*

5.2.1 Scrum Description

Scrum alliance® define Scrum as "a simple yet incredibly powerful set of principles and practices that help teams deliver products in short cycles, enabling fast feedback, continual improvement, and rapid adaptation to change."(“Scrum Alliance,” 2014). In Figure 5.1 shows the basic process behind the Scrum framework.

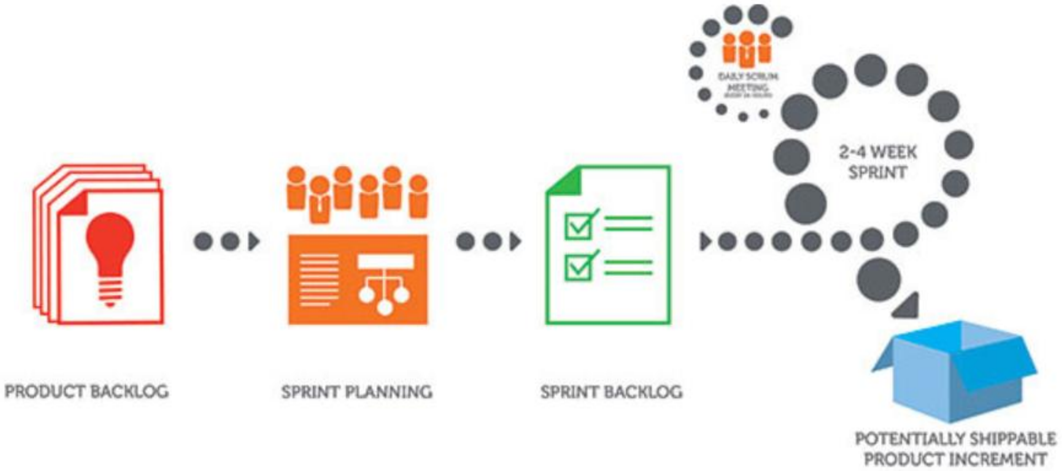


Figure 5-1 Scrum Framework (“Scrum Alliance,” 2014)

The product backlog represents the features the product is required to have. This list is prioritized and tends to evolve as the project advances.

The sprint planning is a meeting where the top features from the product backlog are considered for implementation. These features are planned and estimated, and represented in the sprint backlog.

The sprint backlog represents the items to be completed on a sprint, and is the artifact that results from the sprint planning meeting.

A sprint is a Scrum iteration where the items in a sprint backlog are completed. Each item has to be integrated and tested. A sprint can take from 2 to 4 weeks. Every day there is a daily Scrum meeting to assess progress and adjust if necessary. The result of a sprint is a potentially shippable product, which means that the features selected from the product backlog are functional and ready to deliver. At the end of each sprint there is a sprint end meeting where the product is reviewed and a review of the teamwork is conducted to improve in the next sprint.

Once a sprint ends, there is a new cycle where the team selects new features from the product backlog, plans how to implement the features and conducts a new sprint to do it.

5.2.2 Integration of Scrum with Game Development

Scrum is one of the most adopted widely used agile frameworks for game development (Kasurinen et al., 2013; Keith, 2010). This is because it has a clear framework that can be adapted or combined with other agile or non-agile practices.

There are some reports on how to adopt Scrum framework to do game development (Godoy & Barbosa, 2010; Keith, 2010). Keith offers a detailed work on the adaption of Scrum to game development. The flexibility and fast feedback that Scrum brings helps to clarify the uncertainty that many game projects have.

One important aspect to point out is that scrum helps to find the fun first in games. Finding the fun is the leading directive in pre-production, and in scrum we can prioritize the product backlog to find the fun in the first cycles. The game idea may change with each iteration, and at the end, the game may be different from the concept that we began with. This is fine as long as we achieve the goals even if these goals change. Figure 5.2 helps to illustrate this point by comparing a traditional game development model with an agile one.

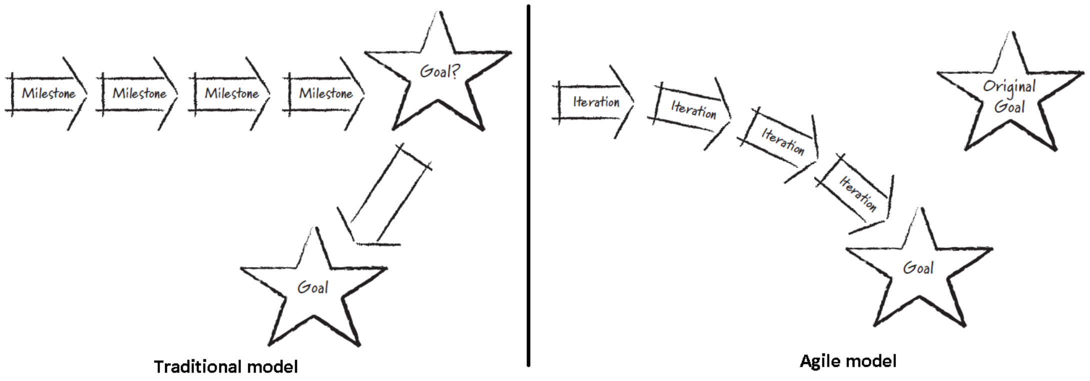


Figure 5-2 Traditional vs. Agile Models Goals Comparison (Keith, 2010)

There are some challenges when adapting Scrum to game development. The multidisciplinary nature of game development is one of them. Scrum has only three roles, but in game development there are many specialized people like programmers, artists or designers. To integrate these specialized roles among the role of team members it is important to understand how team members will communicate. Another challenge is how to integrate the game design with Scrum, how much game design should be done prior to the creation of the product backlog, how much should be created in sprints? This is important since the product backlog is based on features that the game has, but there can be no features if there is no initial game design that indicates these features. Another challenge is how the Scrum framework may be adapted to incorporate the unique aspects of game development. Figure 5.3 shows the adapted agile flow that Keith does for game development. We can see that the flow is a lot of similarity with the Scrum framework, which means the Scrum framework can be adapted in a similar way for game development.

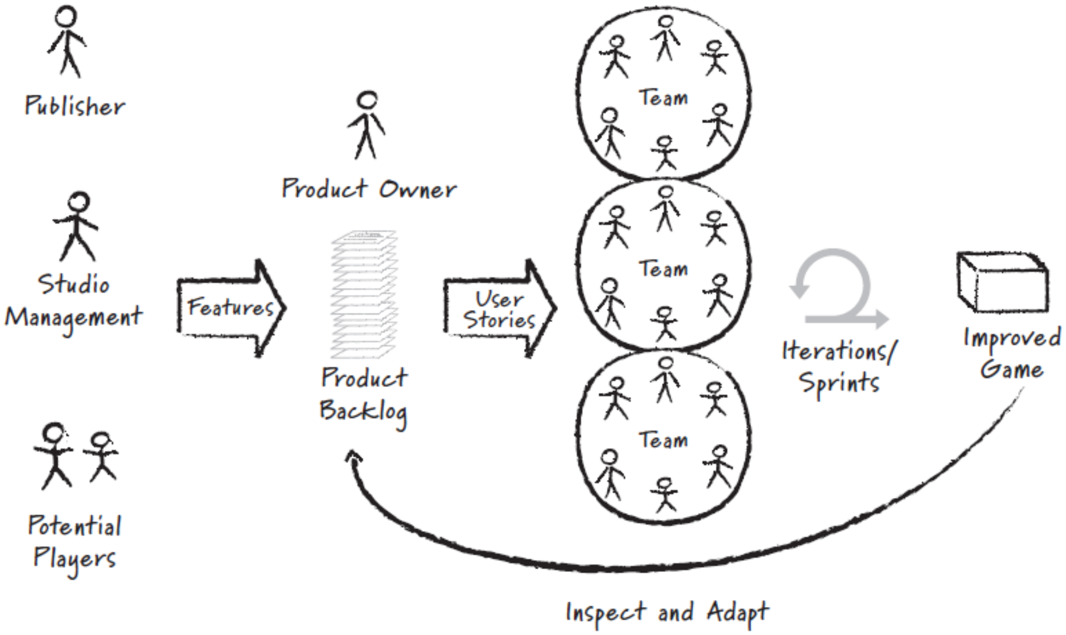


Figure 5-3 Agile Game Development Flow (Keith, 2010)

Is important to clarify that Scrum is not a silver bullet (Keith, 2010) and that game projects may fail when using scrum for many reasons. Some reasons are beyond the scope of Scrum, but some others are related to a bad adoption or miss conception of the Scrum concepts. Finding a way to adopt and understand the Scrum concepts for game development is a challenge by itself.

5.3 Patterns Use in Scrum: Software Development Project Pattern (SDPP)

In this section we discuss patterns use in software engineering and how patterns can be used to help to integrate our proposal with agile development.

5.3.1 Patterns in Software Engineering

We discussed patterns in Section 2.4.3 and described the main classifications of patterns, which are: improvement patterns, collaboration patterns and process patterns. Process patterns are especially useful to integrate the improved Game Design Document (iGDD) and Game Experience Management (GEM) with agile development. A process pattern can provide the framework to merge these methodologies and tools and integrated with an agile development model to execute game development projects project.

We need to integrate in one process pattern the framework of Scrum with the iGDD and the GEM. Therefore there is a need for a process pattern flexible enough to allow this integration of a development model with the iGGD concept and the GEM methodology.

5.3.2 Software Development Project Pattern (sdPP)

The software development project pattern (sdPP) framework (Martín et al., 2012) uses a data model to describe both the problem and the solution to the problem. Figure 5-4 shows the sdPP model, which is composed of objects described by textual information, information based on experience and information based on activities and products. As we can see, the problem is described by information based on experience and the solution adds the "to-do" based on the experience in order to implement a better solution. This information based on experience enriches the context and best practices of application if the adopted solution is by the pattern. The flexibility of the sdPP makes them an appropriate framework for using agile development models (like Scrum) to create an instance of the sdPP. The sdPP for agile development has been proven with positive results as described by Martín et al. "the validation of the sdPP framework indicate that their effective use contributes to the improvement of the results in the quality of the products developed during a software project, the predictability of the effort estimations to develop a project and the provision of a useful way to identify and adapt the most recommended practices for a specific type of a software project" (Martín et al., 2012).

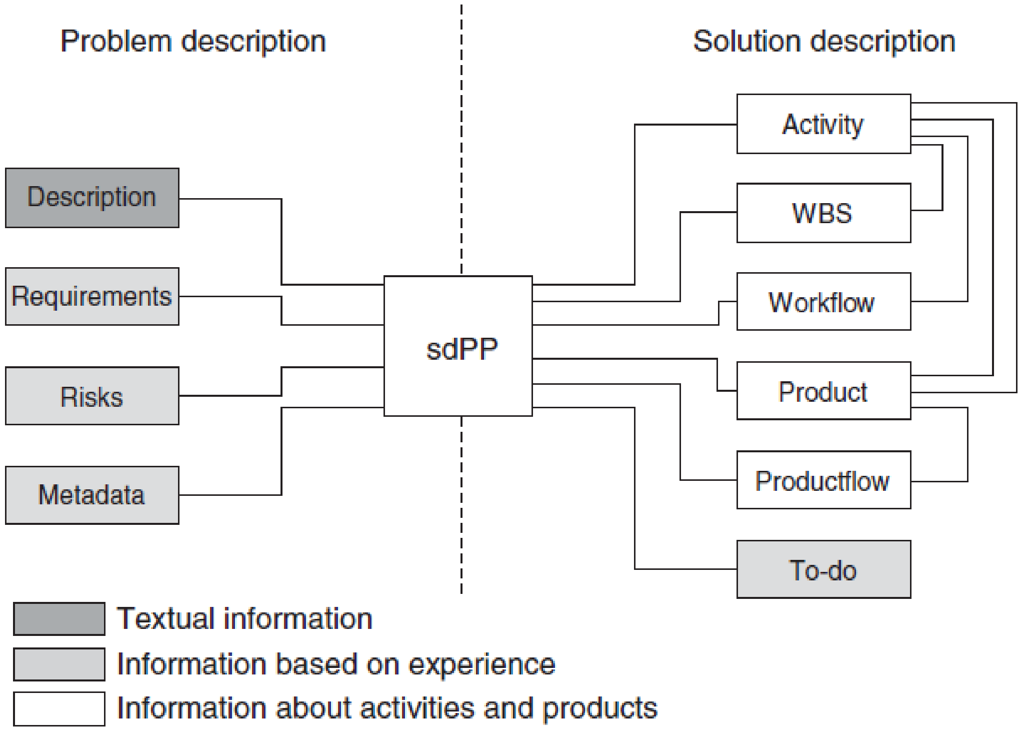


Figure 5-4 sdPP Model (Martín et al., 2012)

5.4 *SDPP for Video Games*

In this section we discuss about the use of sdPP with Scrum and how to integrate the iGDD and GEM to the sdPP with Scrum.

5.4.1 sdPP with Scrum

There is an instance of sdPP working with Scrum, which can be found in [1]. This instance describes the Scrum practices in the sdPP objects. Figure 5-5 shows how the workflow objects of the sdPP with the Scrum instance looks like. We can see a clearly sequence of activities and the products related to each activity. The flexibility of the sdPP allows the use of this abstraction of Scrum framework to incorporate in this instance the concepts of the iGDD

[1] http://www.diego-martin.info/index.php/mi-research/sdpp-project/sdpp-examples/36-scrum-sdpp

and the products and activities of GEM methodology. The resulting instance of the sdPP is described on Section 5.4.2.

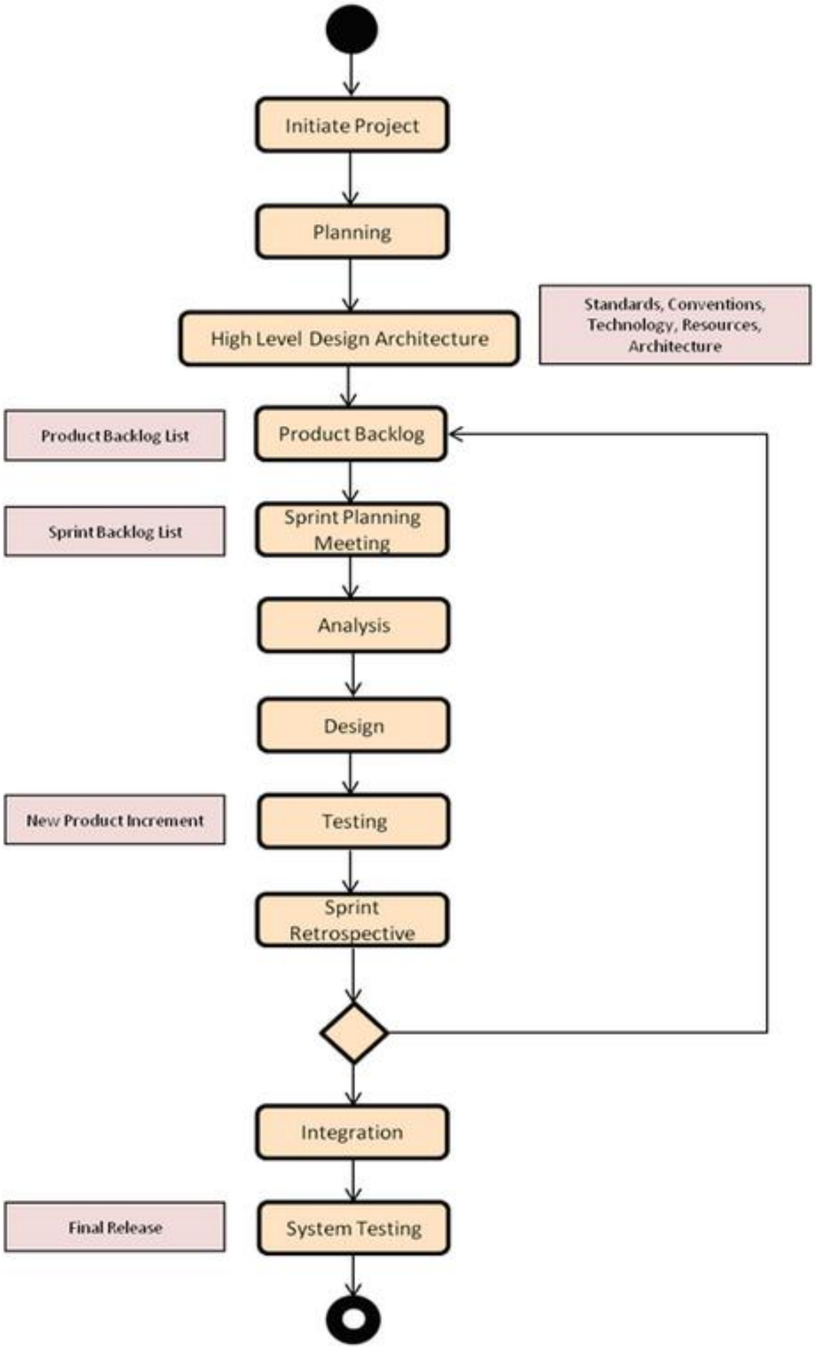


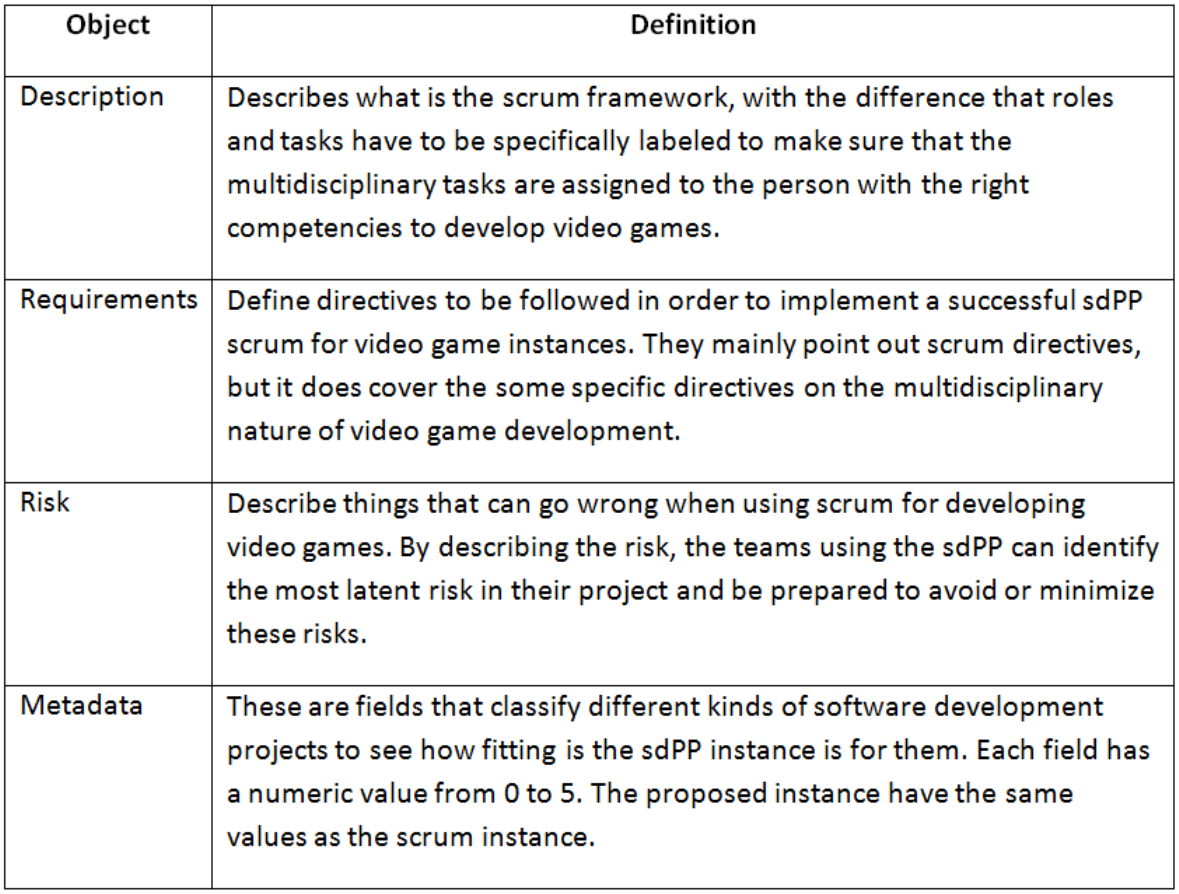
Figure 5-5 sdPP Scrum Instance Productflow

5.4.2 sdPP with Scrum integrating iGDD and GEM for Game Development

The sdPP with Scrum instance is used as a base to create our own instance of Scrum for video games. Each sdPP object was analyzed and adapted to the specific particularities of game development, and as a result we create an instance of the sdPP with Scrum for video games specially tailored to be used with the iGDD and GEM.

On the problem side of sdPP Table 5-2 show the objects and the description of each one, for the created instance:

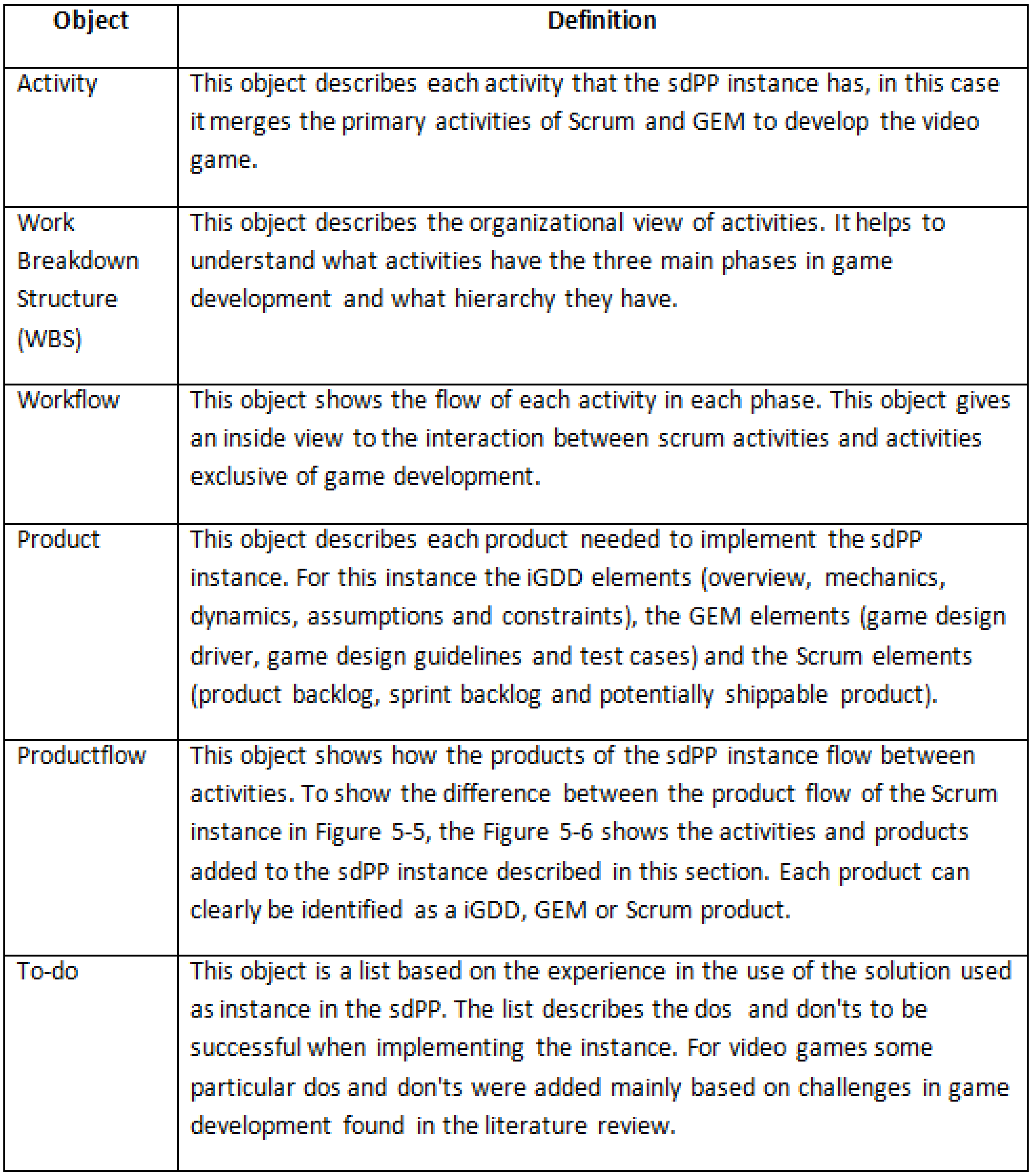
Table 5-2 Objects Definition Problem Side



Object	Definition
Description	Describes what is the scrum framework, with the difference that roles and tasks have to be specifically labeled to make sure that the multidisciplinary tasks are assigned to the person with the right competencies to develop video games.
Requirements	Define directives to be followed in order to implement a successful sdPP scrum for video game instances. They mainly point out scrum directives, but it does cover the some specific directives on the multidisciplinary nature of video game development.
Risk	Describe things that can go wrong when using scrum for developing video games. By describing the risk, the teams using the sdPP can identify the most latent risk in their project and be prepared to avoid or minimize these risks.
Metadata	These are fields that classify different kinds of software development projects to see how fitting is the sdPP instance is for them. Each field has a numeric value from 0 to 5. The proposed instance have the same values as the scrum instance.

On the solution side of sdPP Table 5-3 show the objects and the description of each one, for the created instance:

Table 5-3 Object Definition Solution Side



Object	Definition
Activity	This object describes each activity that the sdPP instance has, in this case it merges the primary activities of Scrum and GEM to develop the video game.
Work Breakdown Structure (WBS)	This object describes the organizational view of activities. It helps to understand what activities have the three main phases in game development and what hierarchy they have.
Workflow	This object shows the flow of each activity in each phase. This object gives an inside view to the interaction between scrum activities and activities exclusive of game development.
Product	This object describes each product needed to implement the sdPP instance. For this instance the iGDD elements (overview, mechanics, dynamics, assumptions and constraints), the GEM elements (game design driver, game design guidelines and test cases) and the Scrum elements (product backlog, sprint backlog and potentially shippable product).
Productflow	This object shows how the products of the sdPP instance flow between activities. To show the difference between the product flow of the Scrum instance in Figure 5-5, the Figure 5-6 shows the activities and products added to the sdPP instance described in this section. Each product can clearly be identified as a iGDD, GEM or Scrum product.
To-do	This object is a list based on the experience in the use of the solution used as instance in the sdPP. The list describes the dos and don'ts to be successful when implementing the instance. For video games some particular dos and don'ts were added mainly based on challenges in game development found in the literature review.

The productflow of the sdPP instance described in this section can be used to identify the main differences between the base Scrum instance and the proposed instance to use the iGDD and GEM. Figure 5-6 makes a clear distinction by using green background on activities added and yellow on activities modified to support game development. On the product side each product shows to which solution it belongs.

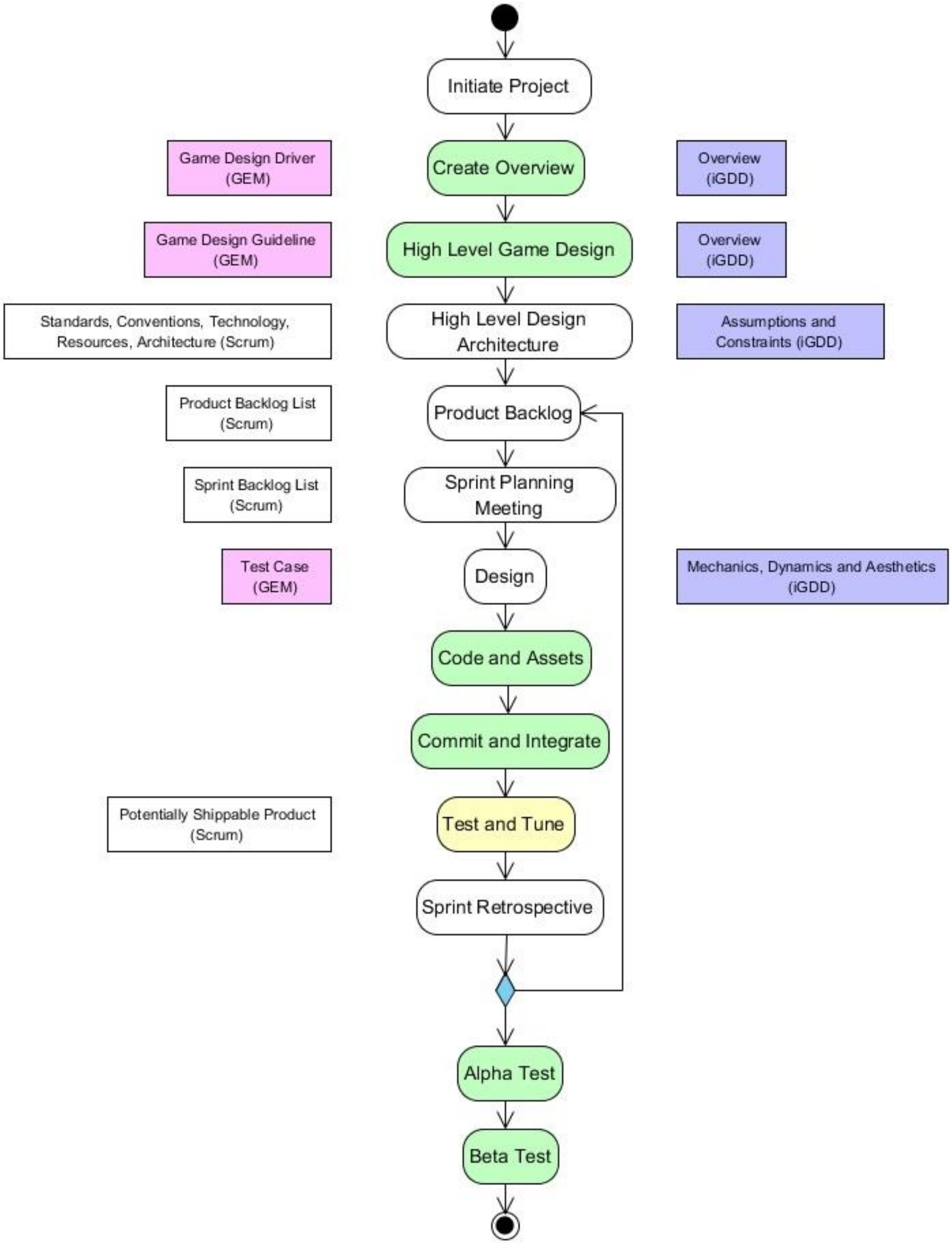

Initiate Project
Game Design Driver (GEM)
Create Overview
Overview (iGDD)
Game Design Guideline (GEM)
High Level Game Design
Overview (iGDD)
Standards, Conventions, Technology, Resources, Architecture (Scrum)
High Level Design Architecture
Assumptions and Constraints (iGDD)
Product Backlog List (Scrum)
Product Backlog
Sprint Backlog List (Scrum)
Sprint Planning Meeting
Test Case (GEM)
Design
Mechanics, Dynamics and Aesthetics (iGDD)
Code and Assets
Commit and Integrate
Potentially Shippable Product (Scrum)
Test and Tune
Sprint Retrospective
Alpha Test
Beta Test

Figure 5-6 sdPP Video Games Productflow

5.5 Game Experience Driven Agile Development (GamExDAD)

In this section the GamExDAD is described as the integrations of the iGDD, GEM and Scrum in an sdPP instance to develop video games. We do a high level description of the activities and the interaction with the different products in the GamExDAD. Then, the GamExDAD model is described showing the detailed relations among concepts in the integrated models.

5.5.1 A walkthrough GamExDAD

To explain the general flow and interaction of the GamExDAD, the productflow diagram shown in Figure 5-6 will be used for a high level explanation of each activity.

Initiate project: a game development project may have different sources: an original idea for a game, an opportunity found in a specific market, or a company that wants to announce its new product or service. Once a game idea from some source initiates a game development project, the first activity is about assigning resources in order for a person or a team to transform a game idea into a game concept.

Create overview: a person or a team uses the *overview (iGDD)* to describe the concept of the game. They describe the game in a brief abstract, identify the main objectives of the game, the genre of the game, ask questions as to why the game is worth doing, define which type of players would like to play the game, and what will be the main activities that the player will be doing while playing the game. Once this information is known the team will use it to establish the game *design drivers (GEM),* which are high level properties that the game will have in order to generate the desired experience in the player. Each driver should be linked the main game objective that is helping to achieve.

High level game design: once the game design drivers are defined, the team will continue using the *overview(iGDD)* to define some main features of the game: the game modalities (single player, multiplayer, online, arcade mode, history mode, among others), the platform or platforms on which the game is intended to run, the game theme (medieval, futuristic, western, among others), the game story and an initial scope of the levels, size and time that the game may require. Once the overview is finished and the team has a better understanding of what the game will be, for each game design driver the team will define one or more *game design guidelines (GEM): these* guidelines provide information on how to create specific game elements (all enemies, music in levels, personality of some non-player character). The guidelines will provide information as to which game elements or game elements category from the *mechanics, dynamics and aesthetics (iGDD)* they should be related in order to establish the relation when these elements or elements category are created.

High-level design architecture: once the game concept, the game design driver and guidelines have been established, the technical setting on which the game will be created will be established. The standards, conventions, technology, resources and architecture is not an explicit Scrum product, but is implied that a self-organized team will arrange these technical settings so the software part of the game can be created. These technical settings are reviewed to determinate any technical constraint (iGDD) that the game may have (e.g. the game will be in 2 dimensions given the capabilities of the game engine chosen). The game overview is reviewed also to find any business constraint (iGDD) that the game may have (e.g. the game needs to conform to the Entertainment Software Rating Board category E for everyone). Among the constraints any necessary assumption is defined (e.g. to implement an online mode in a game we establish the following assumption "the quality of the connection will be stable").

Product backlog: this activity defines the requirements of the game in the product backlog (Scrum). These requirements are represented by product backlog items, which are prioritized and have to be small enough to be completed by a team in one sprint. The product backlog item can be functional or non-functional. An example of a product backlog item may be the development of one level of the game or the animation, sound effects and codification of all the main character actions.

Sprint planning meeting: this activity defines the task to be done in one sprint. The team takes the top priority product backlog item and breaks it in to small tasks, assigns an expected effort and places it on the sprint backlog (Scrum). The team keeps assigning tasks to the sprint backlog until the time available for the sprint is full.

Design: all tasks related to game elements will be required to do an initial design so the artist and programmers can use this design as reference to create the assets and code of the elements. The game design will be related mainly to the mechanics, dynamics and aesthetics (iGDD) of the game. The design of a level of the game will involve all three categories, while the design of the main character and all his action will involve only mechanics and aesthetics. Each game element to be designed has to be checked for relation with the guidelines and how it affects the element creation. Once all the game elements that belong to a guideline have been created and validated, the team will create a test case (GEM) to measure how well the guideline achieves its objectives. Each test case will specify when it should be executed.

Code and asset: this activity refers to the creation of the game elements and all the code and assets (such as music or animations) that belong to the elements. All the technical tasks needed in order to create the game elements are done in this activity as well.

Commit and integrate: this activity integrates the game elements so the product resulting from the sprint can be tested.

Test and tune: the resulting product of the sprint is tested, small adjustments can be made in order to polish the game, but radical changes should be places in the product backlog in order to include them in the next sprint. The result from this activity will be a potentially shippable product (Scrum).

Sprint retrospective: in this activity the game is shown to the product owner, he or she gives feedback, and any change derived from this feedback should be represented in the product backlog.

Alpha test: the game have all the features ready and no more features will be added, the objective of this activity is to play the game and find and remove bugs. This activity requires exhaustively testing the game, trying to cover as many actions as possible in the game.

Beta test: the game has no known bugs; therefore the objective of this activity is to test the experiences of the possible market that will play the game. This activity can be close, which means that only personal of the company or an outsource company will do the test, or it can be open which means the teams grants access to the game to potentially players that will give feedback. The result from this activity is the full game ready to be released.

It is important to note that these are high-level descriptions of the activities and their relations with the products. The details will depend on the team since Scrum is based on self-organized teams, which mean each team can follow these activities in its own way.

5.5.2 GamExDAD Model

The GamExDAD method merges the iGDD and GEM with Scrum framework using the sdPP. Figure 5-7 shows the model, which represents the concepts of the GamExDAD and how they are related. The relations to point out in this model are the ones that connect the sdPP with the iGDD and the GEM. As we can see the sdPP is composed of a version of the iGDD and the GEM. The GEM generates activities to be introduced in the sdPP, which merges with the activities that the scrum instance of the sdPP have. Both the iGDD and the GEM generate products to be introduced in the sdPP instance.

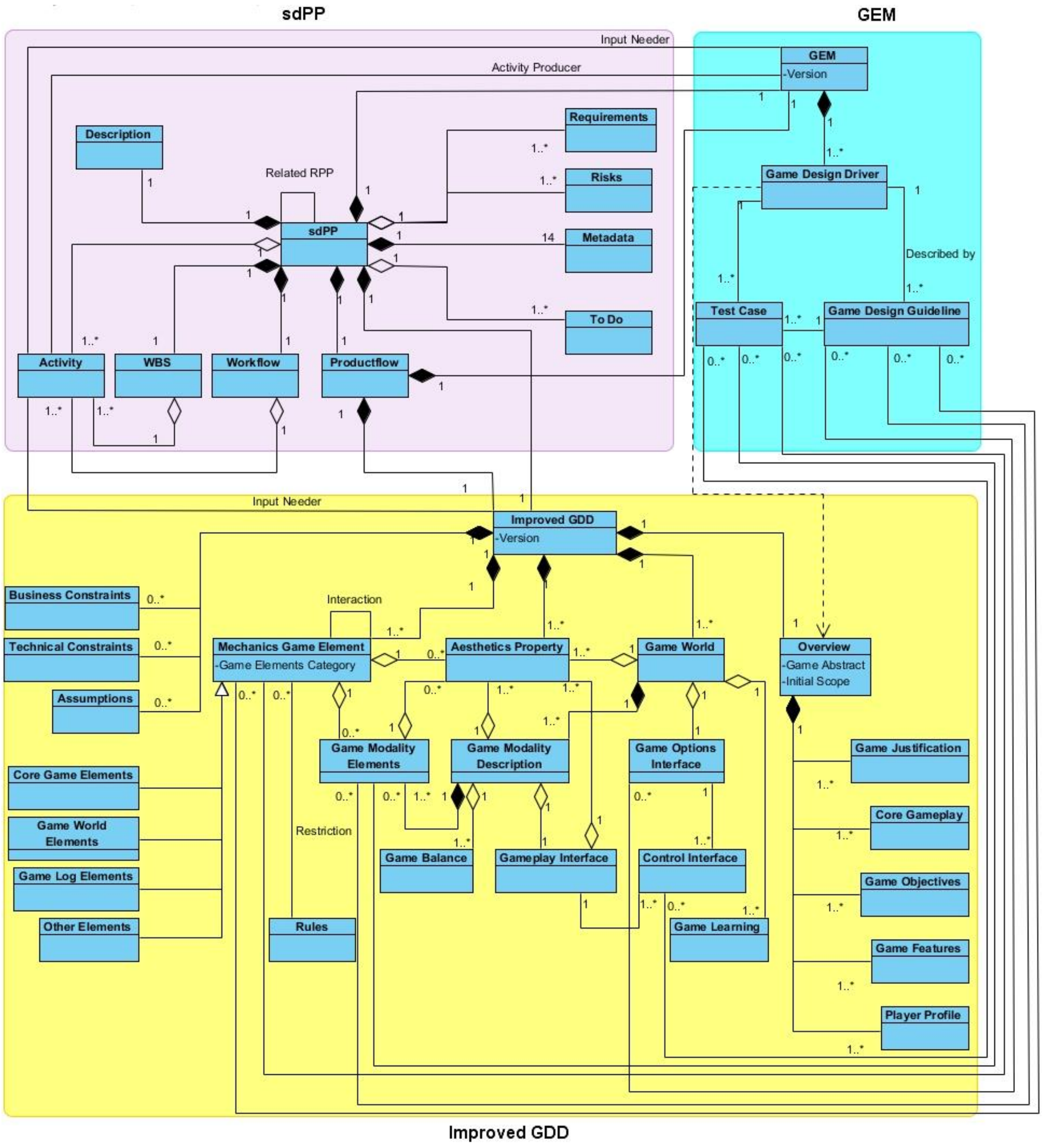


Figure 5-7 GamExDAD model

5.6 *Discussion*

GamExDAD integrates the iGDD and the GEM into Scrum framework. This integration can facilitate the use and adoption of both iGDD and GEM with a proved agile framework in game development.

GamExDAD is aimed for small emerging companies that have trouble finding the right tools and methods to design and develop a game that can bring the desired experience to the player.

A software tool can help to overcome some challenges in the use of the GamExDAD, by establishing the GamExDAD rules of use, and constantly checking for conformance with these rules.

5.7 *Summary*

In this chapter we presented the ExGameDAD as a solution to the rework and undesired user experience in game development, which can lead, to waste of resources, games that do not meet the desired Ux, cancelled project or even the bankruptcy of a company. Our work is aimed to be used by emerging and small game development companies that have problems finding tools and methods to overcome these problems.

CHAPTER 6
EVALUATION AND VALIDATION

This chapter evaluates the Game Experience Driven Agile Development (GamExDAD) to find out if the player experience generated by the game can be improved by establishing the desired experience at the beginning of the project and tracing it to the game elements and reducing the rework while developing the game by creating a Game Design Document (GDD) that can clearly communicate the game requirements to be implemented. First, the approach to validating the GamExDAD is described. Second, the case study to be used in the validation is detailed. Third, the execution of the case study is described. Fourth, the results obtained from the validation are presented. Fifth, the results are interpreted and analyzed. Finally, a small summary of the chapter is given.

6.1 Validation Approach

The strategy proposed to validate the research questions stated in Section 1.2.3, is based on the execution of a case study that provides experimental evidence that GamExDAD reduces rework and shows promising results on improving player experience in game development.

The arguments supporting the claims come from a case study that compares the GamExDAD to the currently used and proposed equivalent. Quantitative and qualitative information resulting from the case study is used as supporting evidence for the claims made in this chapter. In Section 6.2 the case study is described in detail.

6.2 Case of Study

In this section the case study used to validate the research questions mentioned in Section 1.2.3 is described.

The case study has the following goals:

- Measure whether or not our proposal will decrease rework while developing the game.
- Measure whether or not our proposal is capable of improving the player experience that the game generates.

The first research question is:

Is it possible to decrease rework while developing a game, by formalizing the design with a GDD that has a clearly defined structure, relations and details, and incorporates the Software Requirement Specification (SRS) best practices?

The second research question is:

Is it possible to improve the player experience that the game brings by managing, handling and tracking the desired player experiences in early stages of game development?

A third question is added to track how the productivity is affected by using the GamExDAD:

How much productivity is affected by incorporating our GamExDAD in the game development process?

The variables used in the case are defined as follow:

- Rework is defined as the time invested in an artifact after the first test is done on it. To measure the rework in the case study, any artifact put to test for the first time will start to register rework time after the test is done.
- Player experience is defined as the way a person feels from the interaction of playing a video game. The player experience is a complex variable, whose definition is not yet standardized. However as discussed in the related work section, there are several studies that attempt to define and detail the player experience. For this case the model Core Elements of Gaming Experience (CEGE) proposed by Calvillo et al. (Calvillo-Gámez et al., 2010) was used because at the moment it was the best model according to our criteria: it can be applied in the early stages, it has a clear set of variables and provides an evaluation tool. The model is divided into latent variables, and is further divided into observable variables. The latent-observable variables used in the model are the following:
    - Puppetry-Control: Control is formed by the actions and events that the game has available to the player.
    - Puppetry-Facilitators: The facilitators are the amount of time that the player is willing to play, the previous experiences with similar games or other games, and the aesthetic values of the game.
    - Puppetry-Ownership: Ownership is when the player takes responsibility for the actions of the game, he or she feels them as his or hers because they are the result of conscious actions and the game has acknowledged this by rewarding him or her.

    - Video game-Environment: Environment is the way the game is presented to the player, the physical implementation into graphics and sounds.
    - Video game-Gameplay: Game-play defines what the game is about, its rules and scenario.
- Enjoyment and frustration are variables added as a reference to measure the overall enjoyment and frustration of the player and are defined as follows:
    - Enjoyment: Enjoyment is a positive experience for the player as result of playing the game.
    - Frustration: Frustration is a negative experience for the player as result of failing to overcome challenges in the game.

All the variables (latent and observable) are related to the CEGE-Questionnaire, a tool that can be given to the players after they play the game. Table 6-1 shows the variables information. CEGE-Questionnaire used in the case of study can be found in Appendix D.

Table 6-1 CEGE Variables and Items

Category	Type	Scale	Num. Items (n)
Enjoyment	Reference	Positive	3
Frustration	Reference	Negative	2
Puppetry	CEGE	Positive	21
Video game	CEGE	Positive	12

The applied surveys were processed as follows. Each item of the CEGEQ is represented as an ordinal value v ϵ [0,6] where zero is the absence of the factor and six is the maximum value. Let C={C_1(Enjoyment), C_2(Frustration), C_3(Puppetry), C_4(Video game)} be the categories of the CEGEQ. The evaluation cˆ_i for the category C_i ϵC in each survey instance is calculated as

$$\hat{c}_i \; = \frac{t \cdot \sum_{v \in C_i} v}{6 \cdot n_i},$$

where t ϵ {+1,-1}, is assigned based on the type of category (+1 for positive categories and -1 for frustration); and n_i, is the number of items of the category C_i as described in table II. In this way, each survey is represented by four normalized values; i.e., cˆ_1,cˆ_3,cˆ_4 ϵ [0,1] and cˆ_2 ϵ [-1,0] because frustration is a negative category.

Productivity is defined as the amount of requirements that a team can complete in an hour. A requirement is the description of base functionality that the game must have. A requirement may be broken down into user histories. The base requirements were the same for all the teams, but all the teams have the liberty to break down the requirements in their own way.

6.2.1 Case Study Design

The case study was created with a strategy of empirical research comparing two methods based on the work defined by Wohlin (Wohlin et al., 2012). To structure our experiment we used the suggestions offered by Kitchenham (Kitchenham et al., 2007).

6.2.2 Case Study Subjects

For the case study, two groups were created. The first is the video game developers: formed by students in a software engineering master degree program. Testers consisting of children that are learning multiplication form the second group. The main function of the first group is to create the game. The main function of the second group is to test the game.

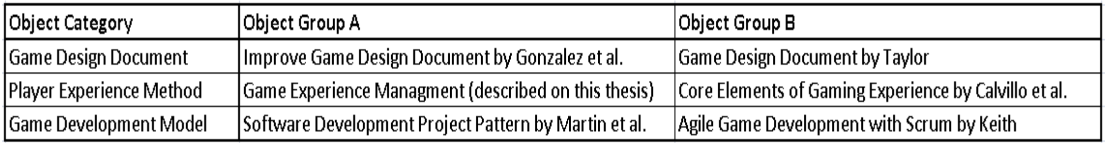
6.2.3 Case Study Objects

There are three experimental objects to analyze in two groups. These three key experimental objects are shown in Table 6-2. The first category is documentation of the game design in the GDD, the second is about models for improving the player experience, and the latter is about methodologies for game development. We divide the objects in two groups as follow:

Group A contains our proposal, which we will be using to meet the goal of this case, which is to significantly improve the player experience. The objects in this group are: the GDD proposed in (Gonzalez et al., 2012); the software design Project Pattern (sdPP) (García Guzmán, Martín, Urbano, & Amescua, 2012) adapted for video games with Scrum; and the Game Experience Proposal presented in this thesis.

Group B contains the counterproposal with which we contrast our proposal to see if we meet the goal. The objects in this group are: the GDD from Taylors (Taylor, 1999); Agile Game Development with Scrum from Keith (Keith, 2010); and the CEGE´s from Calvillo (Calvillo-Gámez et al., 2010).

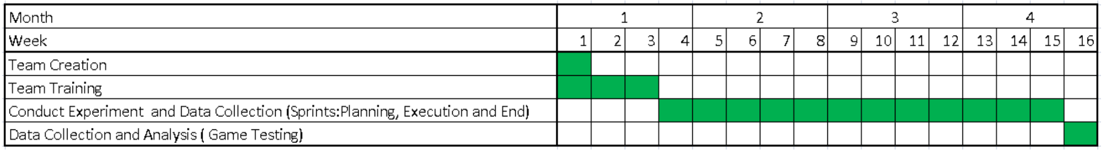
Table 6-2 Objects Distribution



Object Category	Object Group A	Object Group B
Game Design Document	Improve Game Design Document by Gonzalez et al.	Game Design Document by Taylor
Player Experience Method	Game Experience Managment (described on this thesis)	Core Elements of Gaming Experience by Calvillo et al.
Game Development Model	Software Development Project Pattern by Martin et al.	Agile Game Development with Scrum by Keith

The case study activities and duration is shown in Table 6-3 and the dataflow of objects and subjects is shown in Figure 6-1.

Table 6-3 Activity Planning



Month	1				2				3				4			
Week	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Team Creation	🟩															
Team Training	🟩	🟩	🟩													
Conduct Experiment and Data Collection (Sprints:Planning, Execution and End)				🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	🟩	
Data Collection and Analysis (Game Testing)																🟩

The activities were planned as follows:

Team Creation: The professor creates teams from the master’s program students and divides them into two groups, group A and group B. The time planned for this activity was one hour. The professor conducts an interview with the students about their experience in similar projects and creates the two groups to be as homogeneous as possible based on the information obtained from this interview.

Team Training: The researcher trains the teams of both groups A and B. Each group was trained in different game design documents, game experience models and game development methods as shown in Table 6-1. The students receive the training and prepare the game development settings for the case. The time planned for this activity is three weeks to train groups A and B.

Conduct Experiment and Data Collection: The researcher conducts a sprint-planning meeting with each team. In this meeting the base requirements and its priorities are presented, the team divides the base requirements into user stories to plan the sprint. Then, teams work on the user stories of the sprint for two weeks recording the time spent on each user story. Then, the researcher conducts a sprint end meeting where the user stories are classified as completed, unfinished or rejected, and the review of the correct use of objects (GDD, player experience method and game development model) used by teams during the sprint. Finally, the team and researcher plan the date for the next sprint-planning meeting, ensuring that the time difference between sprint end and sprint planning is less than five days. This process continues until the base requirements are finished or the time available ends. The time for these steps is three months, where the teams should have from four to five sprints and finish from ten to fifteen base requirements.

Data Collection and Analysis: The teams test their learning multiplication game with children. Afterward, they give the participants the CEGE-

Questionnaire and ask for the best and worst features of the game. Then, teams deliver the data (times and User Stories associated with base requirements) obtained from the sprints and their opinion of the objects used during the game development. Finally, the researchers gather the data. Figure 6-1 shows the activities and main roles associated in the data collection process.

To validate the case, we use the Wilcoxon test (Sawilowsky, 2007) in order to find a statistically significant improvement in the player experience and a non-relevant diminishment of productivity. Linear regression analysis was also performed to find correlations between the variables of productivity and experience.

The data of times and User Stories associated with base requirements collected during the case study were recorded in the log of Kanban web. These data were validated for each sprint planning and end. Using the CEGE-Questionnaire, it is possible to identify specific elements that produce the difference in both experiences.

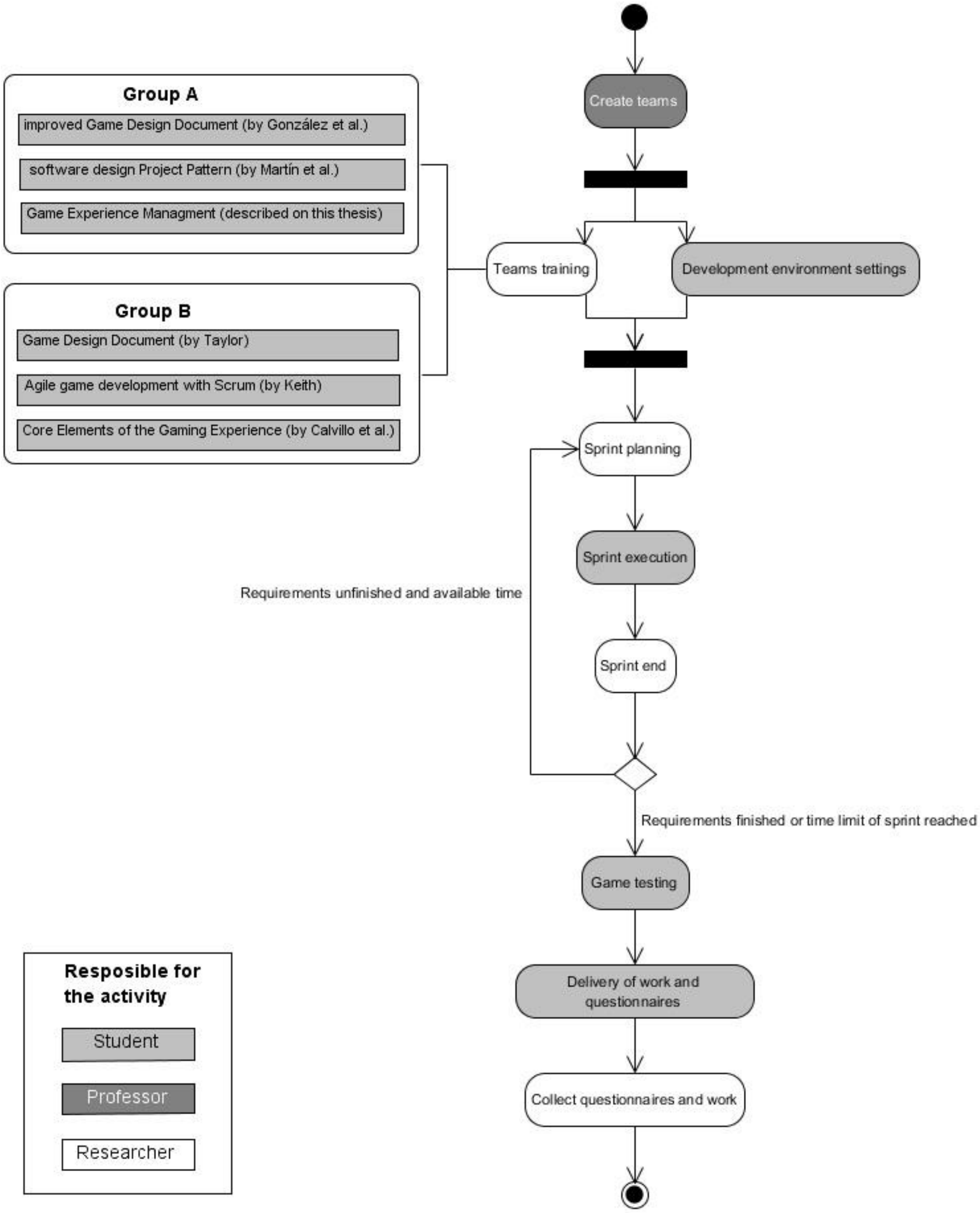

Group A
improved Game Design Document (by González et al.)
software design Project Pattern (by Martín et al.)
Game Experience Managment (described on this thesis)
Group B
Game Design Document (by Taylor)
Agile game development with Scrum (by Keith)
Core Elements of the Gaming Experience (by Calvillo et al.)
Create teams
Teams training
Development environment settings
Sprint planning
Sprint execution
Requirements unfinished and available time
Sprint end
Requirements finished or time limit of sprint reached
Game testing
Delivery of work and questionnaires
Collect questionnaires and work
Resposible for the activity
Student
Professor
Researcher

Figure 6-1 Data Collection

6.3 *Evaluating the Proposal*

This section explains how the case study plan was implemented. First we describe the sample, second we describe how the groups for the case were created and trained, and finally we present the data collection process followed.

6.3.1 Sample Description

For the development of the case, teams of students from the master degree in software engineering program were created. These teams were divided into two groups. Group A: teams create the game with our proposed Game Design Document (GDD), Game Experience Proposal (GEP) and the adaptation of the SDPP for video games. Group B: teams create the game with a Taylor's GDD (Taylor, 1999), a model for the evaluation and management of the player experience called Core Elements of Gaming experience (Calvillo-Gámez et al., 2010) and suggestions from the book "Agile Game Development with Scrum"(Keith, 2010). The last group is composed of children from ages 7 to 11 who are learning multiplications.

An interview with the master degree in software engineering students was conducted to learn their experience in software development and agile methodologies. Based on this information, the teams were formed and homogeneously distributed between the teams in group A and B. The first three weeks of the case study, each team received training on the tools and methodologies used in both Groups A and B. Our assumption for the children is that they were receiving the same teaching techniques to learn multiplication.

6.3.2 Groups Preparation

Team creation and team training were conducted as planned and there was no deviation from the plan. On the conduct experiment and data collection activity there were some deviations from the plan because most teams had problems scheduling meetings, therefore, on several occasions the time difference between sprint end and sprint planning was more than five days. On the other hand, none of the teams was able to finish the fifteen base requirements, but all teams achieved the minimum of ten base requirements. The data collection and analysis was conducted as planned and there was no deviation from the plan.

6.3.3 Data Collection

The data to measure productivity and rework resulting from the game development were collected as follows: At the beginning of each sprint, requirements were divided up on user stories for each team criteria, and these stories were estimated in hours. During each sprint the team members were recording the time spent for each story on a web based Kanban. At each sprint end, every story was reviewed to add any of these statements to the story: "completed", "unfinished" or "rejected". The unfinished and rejected stories were retaken at the next sprint start. After the last sprint, the team members were asked their opinion as to the best and worst of the objects used in the game development. Finally, data from each of the sprints was integrated into spreadsheets for analysis.

Data to measure player experience were collected as follows: Each team tested the game with children from 7 to 11 years of age. Children played the game for 30 minutes and at the end they were given the CEGE-Questionnaire. The results of the questionnaire were integrated into spreadsheets. Finally, the children were asked their opinion about the best and worst of the game.

To validate the correct use of objects in groups A and B, the researchers reviewed the artifacts and guides of the objects at each sprint end meeting. The researchers guided each team in the correct use of each object in order to improve in the use of the objects in each sprint. The researchers took qualitative notes on the use of objects of each team each sprint.

6.4 Results

This section presents the data and results obtained. The first subsection describes data related to the rework evaluation on Groups A and B. The second subsection describes data related to how the productivity is affected by the methods used in Group A and Group B. The third subsection presents data on player experience and the attributes that conform to it in group A and Group B. The last subsection evaluates the data to validate the hypothesis.

6.4.1 Rework results

Rework is defined as the time invested in an artifact after the first test is done on it, as mentioned in Section 6.2. To obtain the rework on each project, we add the rework time reported on each sprint, which gives us the rework time of each project in Groups A and B. The following information concerning rework is presented: the mean of rework in group A was 1.3883, and the mean in Group B was 4.63; the mean of the total time invested in the project in Group A was 50.8811, and the mean in Group B was 40.2686; this gives a mean on percentage of rework in Group A of 2.73%, and in Group B of 11.50%. Figure 6-2 shows the boxplot data related to the rework in group A and B.

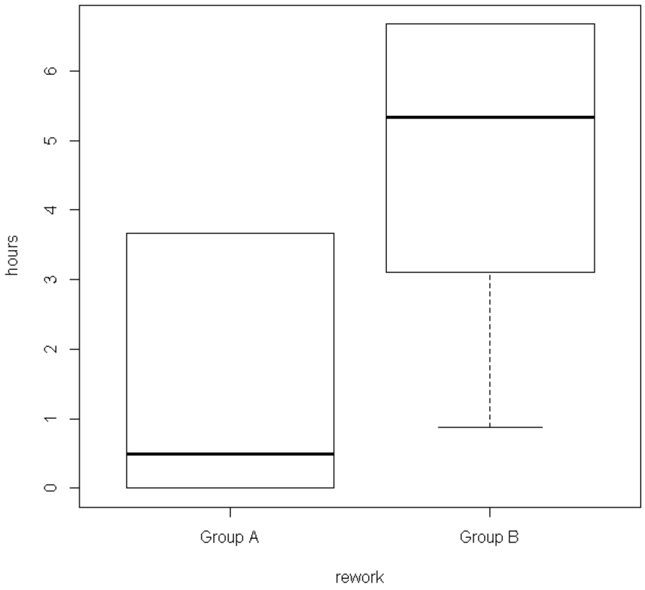


Figure 6-2 Rework time

The data shows a clear difference between rework in Groups A and B, Group B being the one with the most rework.

To test the difference between the rework presented in both groups, a Wilcoxon test was conducted. The results show a statistical improvement in reducing rework in Group A as compared with Group B. The results give a P-value of 0.008497, which is less than 0.05 needed to establish a significant improvement**. This proves that GamExDAD generates less rework than the counter proposal under the case study context;** these results were published in the 4th International Conference on Software Process (Mitre-Hernández et al., 2015).

6.4.2 Productivity Results

To observe how much the productivity is affected by using GamExDAD against the counterproposal, we kept track of the requirements and the time it took to finish each of them. All teams in Groups A and B finish the minimum of ten requirements but none of them finish the maximum of fifteen. Concerning productivity, the following information is presented: the mean of requirements finished in Group A was 11, the mean in Group B was 11.6666; the productivity mean in Group A was 0.21657 requirements/hour and in Group B was 0.28886, which means that productivity was better in Group B. Figure 6-3 shows a boxplot of productivity distribution in both groups. In Figure 6-3 we can clearly observe better values in Group B than in Group A.

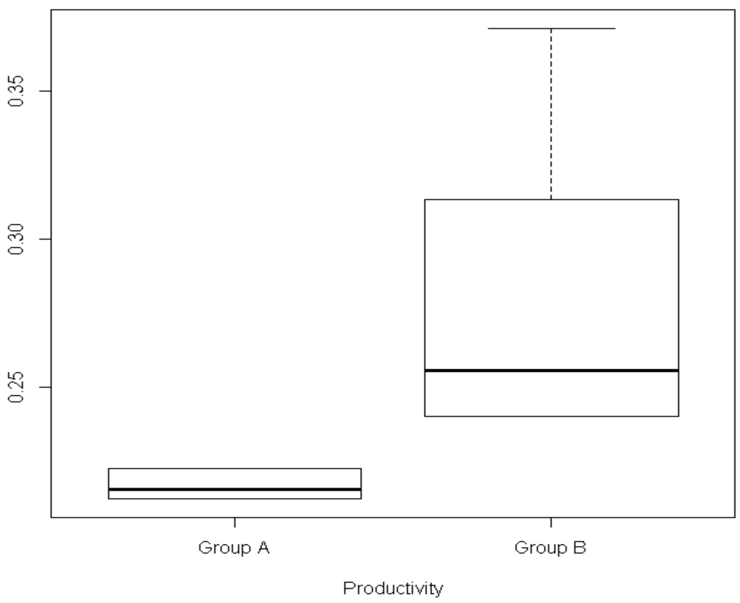


Figure 6-3 Productivity Boxplot Distribution

We applied the Wilcoxon test to see if the productivity was affected by using our proposal comparing it with the counterproposal. The results of the test (as previous data were showing) are that productivity in Group A is statically worse than that of Group B with a p-value of 0.001511. **This means that GamExDAD has an impact on productivity when comparing it with the counterproposal under the case study context**.

6.4.3 Player Experience Results

As stated earlier, a player's experience is defined as the way a person feels from the interaction of playing a video game. In this case we obtained the player experience by adding several variables that the player experiences while playing the game. A statistical analysis was performed where several Wilcoxon tests were used in order to compare the UX between the games developed by groups A and B. A p-value is considered significant when it is less than 0.05.

To measure the variables, we use the CEGE-Questionnaire which when given to a player produces a numeric value for each variable. We gave the questionnaire to the testers in Groups A and B, and after normalizing the data we obtained the results shown in Table 6-4.

Comparison between groups is shown in Table 6-4 and Figure 6-4; the Wilcoxon signed-rank test shows that the enjoyment (W=81, p = 0:29) and video game (W=73.5, p=0.48) variables did not elicit a statistically significant change between games developed by groups A and B. But the frustration in players is reduced significantly (W=104.5, p=0.030), and the puppetry is also increased significantly (W=101, p= 0.049) in those games developed with GEM.

Table 6-4 Player Experience Observable Variables Values

		Group A (GEM)		Group B (CEGES)	
Category		Mean	Std. dev.	Mean	Std. dev.
C_1	Enjoyment	0.940	0.080	0.889	0.155
C_2	Frustration*	-0.194	0.120	-0.368	0.257
C_3	Puppetry*	0.741	0.097	0.675	0.093
C_4	Video game	0.516	0.129	0.494	0.041

The distribution of the player experience and its variables is shown on Figure 6-4. Each boxplot shows the results of Group A and Group B, which results are based on the data collected from the CEGE-Questionnaire given to the children. The distribution shows a higher value in Group A than in Group B.

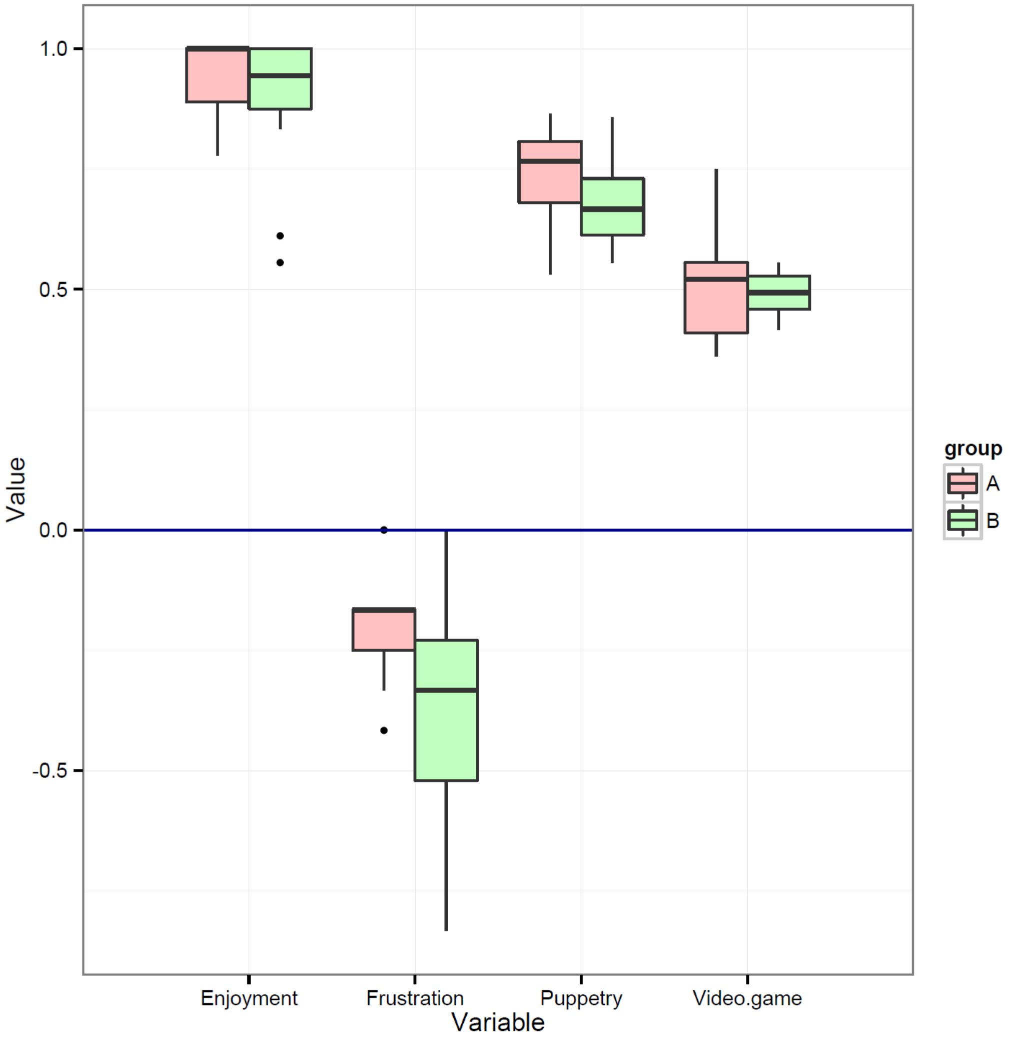


Figure 6-4 Player Experience Boxplot

As shown in Table 6-4 the mean value of enjoyment and game-play categories were slightly better for group A than for group B; however, they did not show a significantly difference between games developed by group A and B. We consider that this was mainly driven by two factors: a) the game genre, the topic, and the general rules were previously defined, and b) the participants were not specialized in improving the game scenario. The puppetry variable, on the other hand, was significantly better for games developed by teams that use our proposal (group A). It impacts directly in reducing the frustration in players. The CEGEQ includes reference variables to see relationships between CEGE and Enjoyment and Frustration. If the CEGE are present, then frustration should be low. We have seen that puppetry could be improved in games developed by teams of group B; as expected by the theory the experience was slightly negative; i.e. the players were more frustrated. It means that CEGE diminished for games developed by group B.

6.5 *Interpretation/Discussion*

In this section we discuss the results presented in the previous section. First, an evaluation of the results and inferences are made. Second, the case study limitations are explained. Finally, the lessons learned from this case are delineated.

6.5.1 Evaluation and Inferences

The quantitative data shows that GamExDAD reduces rework when compared with the counterproposal, this in response to the first research question established in this dissertation. In addition the qualitative data collected from the developers with statements like the following "*It is a very quick way to design a game without getting lost in the documentation*" shows that the structure of the GDD offered by the GamExDAD is key to the game design understanding.

Under the case study context:

- GamExDAD increases the Px by reducing the frustration in players (increasing the puppetry) and shows promising results in improving video game to increase the enjoyment.
- GamExDAD hinders productivity, which means there is a need to find an easier way to adapt and use it. Creating a software tool that facilitates the proper use of GamExDAD can improve the resulting player experience and greatly impact improving the productivity of the team. In the case study the teams followed the proposal using text templates without a software tool that enforced the proper use of the

structure in the iGDD, neither they have a tool that automatically check for associations between the game elements in the iGDD and the game design guidelines in the GEM, doing these activities without the assistance of a software tool was time consuming. We believe this is one of the main causes of the low productivity. This inference is supported by qualitative data obtained from the members of the teams that used our proposal. The following are some opinions expressed by the team members that used GamExDAD when asked the downside of using our proposal:

- "Missing a way to link the game elements with drivers to be more explicit"
- "We need a more visual way to link to the game elements."
- "If you do not have someone to help and guide you on the proper use of it, it can get a bit complicated"

6.5.2 Limitations

The main limitation in this case is that the teams didn't have the time and capacity to create all the game elements, mainly the aesthetic sound part of the game. They used external sources to complete the game.

6.5.3 Lessons Learned

It is hard to coordinate the scrum meeting with different teams when the team members are not fully dedicated to the project. The meetings should be firmly established at the onset of the project and team members should be aware that the meeting date and time will only change by force majeure.

Having fixed requirements previously balanced so the team members can break them down into smaller tasks, proves to be a useful way of keeping track of the productivity on each team.

6.6 Summary

This chapter described how the GamExDAD was validated. First, the evaluation approach was presented, where a case study was selected to compare GamExDAD against a counterproposal composed of current methods and models used to develop games. Second, the case study was described. The subjects, objects, activities and time planned for the case execution was detailed. Third, the execution of the case was presented, where the data collection is explained on the activities as the groups training, game development and testing. Finally, the interpretation of the case study results was presented, where we use the data resulting from the case to answer the research questions of this dissertation and analyze these results. From these results we have concluded that under the case of study context:

- **GamExDAD reduces rework**
- **GamExDAD reduce frustration by improving puppetry**
- **GamExDAD shows promising results in improving Enjoyment by increasing the video game category**
- **GamExDAD hinders productivity due the lack of software tools that can enforce the iGDD structure and automatically check for the relations between the iGDD and GEM.**

CHAPTER 7
CONCLUSION AND FUTURE WORK

This chapter provides a summary of the contributions described in this dissertation. Furthermore, the direction of future research based on this work is also presented.

7.1 Contributions

Game Experience Driven Agile Development (GamExDAD) addresses the problem of ineffective communication of game design due to an inadequate Game Design Document (GDD), and the undesired player experience that games can produce due to poor or non-existent handling of intended player experience during the entire game development process.

Specifically the main contributions of this dissertation include:

- A taxonomy to describe the game design, which can be used as a base to create an ontology. This taxonomy was used as a base to create the iGDD. The taxonomy brings a structured and interconnected way to capture the game design. The game design is divided into three main categories: mechanics, dynamics and aesthetics; these categories are interconnected. Dynamics describes the interactions that the player may have with the game, mechanics describes the elements from which the dynamics are constructed and aesthetics describes the aesthetic properties (mainly visual and sound) that the mechanics and dynamics elements may have. Created to serve as the basis of iGDD which was published in the Proceedings of CGAMES'2012 USA 17th International Conference on Computer Games, and presented in the conference that held place in Louisville, Kentucky, USA from July 30 to August 1, 2012(Gonzalez et al., 2012).

- An iGDD that describes the game design in a formal and detailed way so it can be used as an SRS to create the software of the game. The template uses requirements engineering best practices (Callele, Neufeld, & Schneider, 2011; Jackson, 1995; Jacobs & Ip, 2005; van Lamsweerde, 2001; Wiegers, 2003) and a proven SRS template (Engineering & Committee, 1998) to improve the formality and level of detail in the iGDD. Published in the Proceedings of CGAMES'2012 USA 17th International Conference on Computer Games, and presented in the conference that held place in Louisville, Kentucky, USA from July 30 to August 1, 2012(Gonzalez et al., 2012) and latter validated in the 4th International Conference on Software Process (CIMPS) in 2015 (Mitre-Hernández et al., 2015) where results on how the iGDD help to reduce rework en game development were presented.

- A method to relate the game objectives and the possible experiences that the game brings which can assure that player experiences help to achieve the game objectives. This method produces the desired experience based not only on the objectives but in on the player profile and game concept. Created as result of an academic stay at the Granada University with Dr. Francisco Luis Gutiérrez Vela as director of the Software Specification, Development and Evolution Research Group (GEDES for its acronym in Spanish) from September 10 to December 14, 2012.

- A method to track and validate the desired experiences to the game elements. This can be done by breaking down the game design drivers (which are based on the game objectives) into guidelines that can be associated with specific game elements in the game design and validated when these elements are created.

- A method to evaluate the Ux that the game brings in different stages of the game development process. The measure is done by using test cases to evaluate game design guidelines drivers and the overall Ux. The method can use different evaluations tools, such as questionnaires, measurements based on biometrics or on brain computer interaction.

- An sdPP instance to do game development with Scrum and the use of the iGDD and GEM. Associating the Scrum activities to the GEM and iGDD activities and artifacts creates this instance. These activities and artifacts are associated with the sdPP objects to create the instance. This and the other listed contributions were proposed as part of an accepted project within the call of FOMIX (Fondo Mixto Zacatecas), which will implement the software tool of the GamExDAD proposal. The project is currently in development and will end with the creation and testing of the GamExDAD software and the results will be send to an international journal. The user manual of the GamExDAD prototype can be seen in Appendix C.

- An international publication as early mentioned at the 17th International Conference on Computer Games (CGAMES) in 2012 entitled “Proposal of Game Design Document from software engineering requirements perspective” (Gonzalez et al., 2012).

- An international publication as early mentioned at the 4th International Conference on Software Process (CIMPS) in 2015 entitled “Decreasing Rework in Video Games Development from a Software Engineering Perspective” (Mitre-Hernández et al., 2015).

- An international publication on the International Journal of Software Engineering and Knowledge (IJSEKE) in 2016 entitled "User Experience Management from Early Stages of Computer Game Development" (Mitre-Hernandez, Lara-Alvarez, Gonzalez-Salazar, Mejia-Miranda, & Martín, 2016).

Contributions that were presented have the potential to enrich and grow in various research fields as described in the Section 7.2.

7.2 Direction for Future Research

This section will discuss the direction for future research derived from this dissertation. One project is already exploring some directions suggested by this thesis. Future work resulting from this thesis:

- Software Engineering Field: Create a software tool to support the use of GamExDAD, that is, the creation of a software tool that enforces and facilitates the correct use of GamExDAD to increase productivity in the development team.

- Software Architecture Field: Determine how game design described in the iGDD affects software architecture. The Game design affects the software design, and identifying how the game mechanics and dynamics of the game can be used to select the best architectural style for the game or even to automatically create an initial structure of the software architecture can help to improve the time invested and quality of the game.

- Human-Computer Interaction Field: Extend the GEM to involve potential player in the player profile definition involving player-centric practices. Creating a more accurate understanding of the potential players by involving them in the definition of desired experiences in the GEM, will originate more useful game design drivers to create better an experience for these players.

- Serious Games Field: Evaluate how educational models can be integrated in iGDD mechanics and dynamics to create addictive and effective educational games. In México there is a need for new approaches to improving education, mainly in elementary schools for mathematics. Analyzing the best practices around the world to teach mathematics in elementary school and adapting these practices to the dynamics of a game's GamExDAD can help to improve the learning results in the children.

- Human-Computer Interaction Field: Investigate the different data sources (actions of the player on the game, players opinion, biometrics, voice and face recognition among other) that can be used to evaluate the experience in games, and integrate in the GEM the advantages and disadvantages of using them. Evaluating the user experience in game is not an easy task, but by giving an insight to the advantages and disadvantages that each type of data sources has, the game developers can take better decisions about which type of evaluation tools are better suited to create the test cases for each project.

- Human-Computer Interaction Field: Adapt the GamExDAD so it can create adaptive games that use experiences of the player to change not only the difficulty but other meaningful aspects of the game as well (e. g. history of the game, character responses, weather of the game). Technology that can sense the player while playing by different means and give fast feedback has become more accessible, this gives us the possibility of designing a game that can use the feedback given by this technology to adapt the game in real time by changing meaningful aspects to improve the player experience.

There is an ongoing FOMIX (Fondo Mixto Zacatecas) project as mention on Section 7.1, to strengthen the Software Engineering Master with the creation of a Human-Computer Interaction (HCI) laboratory that is currently working on the following areas:

- Create the software tool of GamExDAD

- Integrate educational models in the iGDD

- Evaluate player experience using data sources such as: actions of the player on the game, player's opinion, biometrics, voice and face recognition.

This FOMIX project will provide the underpinnings to continue the work presented on this dissertation and to explore the serious game fields, towards addressing educational problems.

7.3 Conclusions

This dissertation has offered the GamExDAD as a solution to the problems presented in section 1.2.3. The iGDD was created to create a more formal and structured approach to describe game design as presented in (Gonzalez et al., 2012). The GEM was created to integrate the desired player experience into the game design and to validate and evaluate it in the entire game development process. The GamExDAD was created to validate the viability of this solution in the game development process by integrating both solutions in an agile game development process based on the SCRUM framework.

An experiment was created to evaluate GamExDAD, which gave us the following results:

- **GamExDAD reduces the rework time in game development.**
- **GamExDAD improves player experiences in games.**

GamExDAD is a relevant contribution to the video game field and with the ongoing FOMIX project that will extend the reach of the GamExDAD, it will become a stronger tool for executing game design based on desired player experiences.

BIBLIOGRAPHY

Abrahamsson, P., Oza, N., & Siponen, M. T. (2010). *Agile Software Development*. (T. Dingsøyr, T. Dybå, & N. B. Moe, Eds.). Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-642-12575-1

Adams, E. (2003). *Break Into The Game Industry: How to Get A Job Making Video Games*. McGraw-Hill Osborne Media.

Alexander, C. (1979). *The Timeless Way of Building :* Oxford University Press.

Alexander, C., Ishikawa, S., Silverstein, M., Jacobson, M., Fiksdahl-King, I., & Angel, S. (1977). *A Pattern Language: Towns, Buildings, Construction (Center for Environmental Structure)*. Oxford University Press.

Ampatzoglou, A., & Stamelos, I. (2010). Software engineering research for computer games: A systematic review. *Information and Software Technology*, *52*(9), 888–901. http://doi.org/10.1016/j.infsof.2010.05.004

Apperley, T. H. (2006). Genre and game studies: Toward a critical approach to video game genres. *Simulation & Gaming*, *37*(1), 6–23. http://doi.org/10.1177/1046878105282278

Baldwin, M. (2005). Game Design Document Outline. Retrieved from http://gamesed.com/

Bates, B. (2004). *Game Design [Paperback]*. Premier Press; 2nd Revised edition edition.

Beck, K. (1999). *Extreme Programming Explained: Embrace Change.* Addison-Wesley Longman Publishing Co.

Benington, H. D. (1956). Production of Large Computer Programs. In *Proceedings of the ONR Symposium on Advanced Program Methods for Digital Computers*.

Bethke, E. (2002). *Game Development and Production*. Wordware Publishing Inc.; Pap/Cdr edition.

Boehm, B. W. (1988). A spiral model of software development and enhancement. *Computer*, *21*(5), 61–72. http://doi.org/10.1109/2.59

Brinkkemper, S., Weerd, I., & Weerd, S. (2007). Developing a Reference Method for Game Production by Method Comparison. *IFIP Advances in Information and Communication Technology (AICT)*, *244*(244), 313–327.

Brown, E. (2010). The Life and Tools of a Games Designer. In *Evaluating User Experience in Games* (pp. 73–87). Springer-Verlag.

Brown, M., Kehoe, A., Kirakowski, J., & Pitt, I. (2010). Beyond the Gamepad: HCI and Game Controller Design and Evaluation. In *Evaluating User Experience in Games* (pp. 209–231). Springer-Verlag.

Cairns, P., Cox, A., & Nordin, I. (2014). Immersion in Digital Games: Review of Gaming Experience Research. In I. John Wiley & Sons (Ed.), *Handbook of Digital Games*.

Callele, D., Neufeld, E., & Schneider, K. (2005). Requirements engineering and the creative process in the video game industry. In *Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on* (pp. 240–250). IEEE.

Callele, D., Neufeld, E., & Schneider, K. (2006). Emotional Requirements in Video Games. *14th IEEE International Requirements Engineering Conference (RE'06)*, 299–302. http://doi.org/10.1109/RE.2006.19

Callele, D., Neufeld, E., & Schneider, K. (2011). A report on select research opportunities in requirements engineering for videogame development. *2011 Fourth International Workshop on Multimedia and Enjoyable Requirements Engineering (MERE'11)*, 26–33. http://doi.org/10.1109/MERE.2011.6043942

Calvillo-Gámez, E. H., Cairns, P., & Cox, A. L. (2010). Assessing the Core Elements of the Gaming Experience. In *Evaluating User Experience in Games* (pp. 47–71). Springer-Verlag.

Carless, S. (2009). GDC: Indie Game Sales 101 - The Slides. Retrieved from http://www.gamasutra.com/blogs/SimonCarless/20090323/974/GDC_Indie_Game_Sales_101__The_Slides.php

Caroux, L., Isbister, K., Le Bigot, L., & Vibert, N. (2015). Player–video game interaction: A systematic review of current concepts. *Computers in Human Behavior*, *48*, 366–381.

Chen, J. (2007). Flow in games (and everything else). *Communications of the ACM*, *50*(4), 31. http://doi.org/10.1145/1232743.1232769

Crawford, C. (2003). *Chris Crawford on Game Design [Paperback]*. New Riders; 1 edition.

Desurvire, H., Clapman, M., & Toth, J. A. (2004). Using Heuristics to Evaluate the Playability of Games. In *CHI '04 Conference on Human Factors in Computing Systems* (pp. 1509–1512). Viena, Australia: ACM.

Desurvire, H., & Wiberg, C. (2009). Game Usability Heuristics (PLAY) For Evaluating and Designing Better Games : The Next Iteration. In *3d International Conference on Online Communities and Social Computing: Held as Part of HCI International 2009* (pp. 557–566).

Desurvire, H., & Wiberg, C. (2010). User Experience Design for Inexperienced Gamers: GAP – Game Approachability Principles. In *Evaluating User Experience in Games* (pp. 131–147). Springer-Verlag.

Down, B. I. (2007). Design Document: Play With Fire. Retrieved from http://www.gamasutra.com

Elson, M., Breuer, J., & Quandt, T. (2014). Know Thy Player: An Integrated Model of Player Experience for Digital Games Research. In *Handbook of Digital Games*. John Wiley & Sons.

Engineering, S., & Committee, S. (1998). IEEE Recommended Practice for Software Requirements Specifications. *Electronics*.

Erickson J., Lyytinen, K., & Siau, K. (2005). Agile modeling, agile software development, and extreme programming: The state of research. *Journal of Database Management*, *16*.

Essential Facts about the Computer and Video Game Indsutry. (2015). Retrieved from www.theESA.com

Evaluating User Experience in Games: Concepts and Methods (Human-Computer Interaction Series). (2010). Springer.

Ferrier, A. T. (2008). Game Career Guide Fall 2008. *United Business Media*, 47–49.

Fowler, M., & Highsmith, J. (2001). The Agile Manifesto. *Software Development Magazine*.

Godoy, A., & Barbosa, E. (2010). Game-Scrum: An Approach to Agile Game Development. *IX SBGames (November)*.

González, J. L., Montero, F., Padilla, N., & Guitiérez, F. (2009). Playability as extension of quality in use in video games. *2nd International Workshop …*.

González, J., & Padilla, N. (2008). De la Usabilidad a la Jugabilidad: Diseño de Videojuegos Centrado en el Jugador. *Proceedings of IX Congreso Internacional Interacción*.

González, J., Padilla, N., & Gutiérrez, F. (2009). *Human Centered Design*. (M. Kurosu, Ed.) (Vol. 5619). Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-642-02806-9

González, J., Zea, N., & Gutiérrez, F. (2009). From Usability to Playability: Introduction to Player-Centred Video Game Development Process. In *First International Conference, HCD 2009* (pp. 65–74). San Diego, CA: Springer.

Gonzalez, M., Mitre, H. A., Lemus, C., & Gonzalez, J. L. (2012). Proposal of Game Design Document from software engineering requirements perspective. In *2012 17th International Conference on Computer Games (CGAMES)* (pp. 81–85). IEEE. http://doi.org/10.1109/CGames.2012.6314556

Guo, H., Gao, S., Krogstie, J., & Trætteberg, H. (2015). An Evaluation of an Enhanced Model Driven Approach for Computer Game Creation. In *16th International Conference, BPMDS 2015, 20th International Conference, EMMSAD 2015, Held at CAiSE 2015* (pp. 499–508). Springer

International Publishing.

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* (pp. 04–04). Retrieved from http://www.aaai.org/Papers/Workshops/2004/WS-04-04/WS04-04-001.pdf

IEEE. (1990). *IEEE Standard Glossary of Software Engineering Terminology/IEEE Std 610.12-1990*. Inst of Elect & Electronic. Retrieved from http://www.amazon.com/Standard-Glossary-Engineering-Terminology-610-12-1990/dp/155937067X

Infographic: Mobile Gaming Statistics 2011. (2011). Retrieved from http://www.digitalbuzzblog.com/infographic-mobile-gaming-statistics-stats-2011/

Isbister, K. (2010). Enabling Social Play: A Framework for Design and Evaluation. In *Evaluating User Experience in Games* (pp. 11–22). Springer-Verlag.

ISO/IEC 25010-3: Systems and software engineering: Software product quality and system quality in use models. (2009), 34.

ISO/TC 159/SC. (2010). *ISO 9241-210:2010, Ergonomics of human-system interaction - Part 210: Human-centred design for interactive systems*. Multiple. Distributed through American National Standards Institute (ANSI). Retrieved from http://www.amazon.com/ISO-9241-210-human-system-interaction-Human-centred/dp/B009CFQER2

Jackson, M. (1995). *Software Requirements and Specifications: A Lexicon of Practice, Principles and Prejudices*.

Jason. (2013). Gaming is Good for You (Infographic). Retrieved from http://www.affordableschoolsonline.com/gaming-is-good-for-you-infographic/

Kanode, C. M., & Haddad, H. M. (2009). Software Engineering Challenges in Game Development. *2009 Sixth International Conference on Information Technology: New Generations*, 260–265. http://doi.org/10.1109/ITNG.2009.74

Kasurinen, J., Laine, R., & Smolander, K. (2013). How Applicable Is ISO / IEC 29110 in Game Software Development ? In *14th International Conference, PROFES 2013* (pp. 5–19). Springer Berlin Heidelberg.

Kasurinen, J., Maglyas, A., & Smolander, K. (2014). Is Requirements Engineering Useless in Game Development? In *20th International Working Conference, REFSQ 2014, Essen, Germany, April 7-10, 2014.* (pp. 1–16). Springer International Publishing.

Keith, C. (2010). *Agile Game Development with SCRUM*. Addison Wesley; 1 edition. Retrieved from http://www.amazon.co.uk/Agile-Development-

SCRUM-Addison-Wesley-Signature/dp/0321618521/ref=sr_1_1?s=books&ie=UTF8&qid=1327444787&sr=1-1

Kitchenham, B., Al-Khilidar, H., Babar, M. A., Berry, M., Cox, K., Keung, J., … Zhu, L. (2007). Evaluating guidelines for reporting empirical software engineering studies. *Empirical Software Engineering*, *13*(1), 97–121. http://doi.org/10.1007/s10664-007-9053-5

Koeffel, C., Hochleitner, W., Leitner, J., Haller, M., Geven, A., & Tscheligi, M. (2010). Using Heuristics to Evaluate the Overall User Experience of Video Games and Advanced Interaction Games. In *Evaluating User Experience in Games* (pp. 233–256).

Koleva, B., Tolmie, P., Brundell, P., Benford, S., & Rennick-Egglestone, S. (2015). From Front-End to Back-End and Everything In-Between: Work Practice in Game Development. In *the 2015 Annual Symposium on Computer-Human Interaction in Play (CHI PLAY '15)* (pp. 141–150). ACM New York.

Korhonen, H., & Koivisto, E. (2006). Playability heuristics for mobile games. *… on Human-Computer Interaction with Mobile …*. Retrieved from http://dl.acm.org/citation.cfm?id=1152218

Korhonen, H., & Koivisto, E. (2007). Playability heuristics for mobile multi-player games. *… of the 2nd International Conference on …*, 28–35. Retrieved from http://dl.acm.org/citation.cfm?id=1306828

Korhonen, H., Paavilainen, J., & Saarenpää, H. (2009). Expert review method in game evaluations: comparison of two playability heuristic sets. *Proceedings of the 13th …*. Retrieved from http://dl.acm.org/citation.cfm?id=1621856

Kortmann, R., & Harteveld, C. (2009). Agile game development: lessons learned from software engineering. In *Learn to Game, Game to Learn; the 40th Conference ISAGA 2009*. Society of Simulation and Gaming of Singapore.

Kroll, P., & Kruchten, P. (2003). The rational unified process made easy: a practitioner's guide to the RUP. Retrieved from http://dl.acm.org/citation.cfm?id=937979

Landaeta, J. F., García, J., & Amescua, A. (2008). Practical SPI Planning. In R. V. O'Connor, N. Baddoo, K. Smolander, & R. Messnarz (Eds.), *Software Process Improvement* (Vol. 16, pp. 82–93). Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-540-85936-9

Lankes, M., Bernhaupt, R., & Tscheligi, M. (2010). Evaluating User Experience Factors Using Experiments: Expressive Artificial Faces Embedded in Contexts. In *Evaluating User Experience in Games* (pp. 165–183). Springer-Verlag.

Larman, C. (2003). Agile and Iterative Development: A Manager’s Guide. Retrieved from http://dl.acm.org/citation.cfm?id=861501

Lattanze, A. J., Stafford, J. A., & Weinstock, C. B. (2003). Quality Attribute Workshops (QAWs), Third Edition, (August).

Lemay, P., & Maheux-Lessard, M. (2010). Investigating Experiences and Attitudes Toward Videogames Using a Semantic Differential Methodology. In *Evaluating User Experience in Games* (pp. 89–105). Springer-Verlag.

Martín, D., Guzmán, J. G., Urbano, J., & Llorens, J. (2012). Patterns as objects to manage knowledge in software development organizations. *Knowledge Management Research & Practice*, *10*(3), 252–274. http://doi.org/10.1057/kmrp.2012.15

May, D., & Taylor, P. (2003). Knowledge management with patterns. *Communications of the ACM*, *46*(7), 94–99. http://doi.org/10.1145/792704.792705

McAllister, G., & White, G. R. (2010). Video Game Development and User Experience. In *Evaluating User Experience in Games* (pp. 107–128). Springer-Verlag.

Mirza-Babaei, P., Nacke, L. E., Gregory, J., Collins, N., & Fitzpatrick, G. (2013). How does it play better?: exploring user testing and biometric storyboards in games user research. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems - CHI ’13 (2013)*, (May), 1499–1508. Retrieved from http://dl.acm.org/citation.cfm?id=2466200

Mitre-Hernández, H. A., Lara-Alvarez, C., González-Salazar, M., & Martín, D. (2015). Decreasing Rework in Video Games Development from a Software Engineering Perspective. In *4th International Conference on Software Process Improvement* (pp. 295–304). Mazatlan, Mexico.: Springer International Publishing. http://doi.org/10.1007/978-3-319-26285-7_25

Mitre-Hernandez, H., Lara-Alvarez, C., Gonzalez-Salazar, M., Mejia-Miranda, J., & Martín, D. (2016). User Experience Management from Early Stages of Computer Game Development. *International Journal of Software Engineering and Knowledge (IJSEKE)*.

Mueller, F. “Floyd,” & Bianchi-Berthouze, N. (2010). Evaluating Exertion Games. In *Evaluating User Experience in Games* (pp. 187–207). Springer-Verlag.

Nacke, L. E., Drachen, A., Kuikkaniemi, K., & Kort, Y. A. W. De. (2009). Playability and Player Experience Research. *Proceedings of the IEEE*.

Orita-Almeida, M., & Correa-da-Silva, F. (2013). A Systematic Review of Game Design Methods and Tools. In *12th International Conference,* *ICEC 2013, São Paulo, Brazil, October 16-18, 2013.* (pp. 17–29). Springer Berlin Heidelberg.

Oxland, M. K. (2004). *Gameplay and Design [Paperback]*. Addison Wesley; 1 edition. Retrieved from http://www.amazon.co.uk/Gameplay-Design-Mr-Kevin-Oxland/dp/0321204670/ref=sr_1_1?s=books&ie=UTF8&qid=1327446496&sr=1-1

Pedersen, R. E. (2009). *Game Design Foundations, Second Edition*. Jones & Bartlett Learning. Retrieved from http://www.amazon.com/Game-Design-Foundations-Second-Edition/dp/1598220349

Petrillo, F., & Pimenta, M. (2008). Houston, we have a problem...: a survey of actual problems in computer games development. *Proceedings of the 2008 …*, 707–711. Retrieved from http://dl.acm.org/citation.cfm?id=1363854

Petrillo, F., & Pimenta, M. (2009). What went wrong? A survey of problems in game development. *… in Entertainment (CIE)*, *7*(1), 1–22. Retrieved from http://dl.acm.org/citation.cfm?id=1486521

Pinelle, D., Wong, N., & Stach, T. (2008a). Heuristic evaluation for games: usability principles for video game design. *Proceedings of the SIGCHI Conference on …*, 1453–1462. Retrieved from http://dl.acm.org/citation.cfm?id=1357282

Pinelle, D., Wong, N., & Stach, T. (2008b). Using genres to customize usability evaluations of video games. *Proceedings of the 2008 Conference on …*, 129–136. Retrieved from http://dl.acm.org/citation.cfm?id=1497006

Poels, K., IJsselsteijn, W., Kort, Y. de, & Iersel, B. Van. (2010). Digital Games, the Aftermath: Qualitative Insights into Postgame Experiences. In *Evaluating User Experience in Games* (pp. 149–163). Springer-Verlag.

Rogers, S. (2010). *Level Up!: The Guide to Great Video Game Design [Paperback]*. John Wiley &amp; Sons. Retrieved from http://www.amazon.co.uk/Level-Up-Guide-Great-Design/dp/047068867X/ref=sr_1_2?s=books&ie=UTF8&qid=1327443412&sr=1-2

Rollings, A., & Adams, E. (2003). *Andrew Rollings and Ernest Adams on Game Design*. New Riders; 1 edition. Retrieved from http://www.amazon.co.uk/Andrew-Rollings-Ernest-Adams-Design/dp/1592730019/ref=sr_1_1?s=books&ie=UTF8&qid=1327444866&sr=1-1

Rollings, A., & Morris, D. (2003). *Game Architecture and Design*. New Riders; 1 edition. Retrieved from http://www.amazon.co.uk/Game-Architecture-Design-NRG-Programming/dp/0735713634/ref=sr_1_1?s=books&ie=UTF8&qid=1327

445844&sr=1-1

Rouse, R. (2004). *Game Design, Theory and Practice*. Wordware Publishing Inc.; 2nd Revised edition edition. Retrieved from http://www.amazon.co.uk/Design-Practice-Wordware-Developers-Library/dp/1556229127/ref=sr_1_1?s=books&ie=UTF8&qid=1327446193&sr=1-1

Royce, W. W. (1987). Managing the development of large software systems: concepts and techniques. In *Proceeding ICSE '87 Proceedings of the 9th international conference on Software Engineering* (pp. 328–338). IEEE Computer Society Press Los Alamitos, CA, USA ©1987. http://doi.org/0-89791-216-0

RPG Maker. (n.d.). Retrieved January 1, 2013, from http://www.rpgmakerweb.com/

Sanchez, J. L. G. (2010). *JUGABILIDAD: CARACTERIZACIÓN DE LA EXPERIENCIA DEL JUGADOR EN VIDEOJUEGOS*. University of Granada.

Sawilowsky, S. (2007). Mann-Whitney U test (Wilcoxon Rank-Sum test). In *Encyclopedia of measurement and statistics* (Neil J. Sa, pp. 566–68). SAGE Publications, Inc. http://doi.org/http://dx.doi.org/10.4135/9781412952644.n265

Scacchi, W., & Cooper, K. M. (2015). Research Challenges at the Intersection of Computer Games and Software Engineering. In *Conference on Foundations of Digital Games (FDG 2015)*. ACM.

Schell, J. (2008). *The Art of Game Design: A book of lenses [Paperback]*. Morgan Kaufmann. Retrieved from http://www.amazon.co.uk/Art-Game-Design-book-lenses/dp/0123694965/ref=pd_bxgy_b_img_c

Schummer, J. (2004). Multidisciplinarity, interdisciplinarity, and patterns of research collaboration in nanoscience and nanotechnology. *Scientometrics*, *59*(3), 425–465. http://doi.org/10.1023/B:SCIE.0000018542.71314.38

Schwaber, K., & Beedle, M. (2002). *Agile Software Development with Scrum*. Pearson Education International. Retrieved from http://books.google.com.mx/books/about/Agile_Software_Development_with_Scrum.html?id=lO3XLgAACAAJ&pgis=1

Scrum Alliance. (2014). Retrieved January 1, 2014, from https://www.scrumalliance.org/

Stoica, M., Mircea, M., & Ghilic-Micu, B. (2013). Software Development: Agile vs. Traditional. *Informatica Economica*, *17*(4/2013), 64–76. http://doi.org/10.12948/issn14531305/17.4.2013.06

Sweetser, P., & Wyeth, P. (2005). GameFlow : A Model for Evaluating Player Enjoyment in Games, *3*(3), 1–24.

Takatalo, J., Häkkinen, J., Kaistinen, J., & Nyman, G. (2010). Presence, Involvement, and Flow in Digital Games. In *Evaluating User Experience in Games* (pp. 23–46). Springer-Verlag.

Tavinor, G. (2008). Definition of Videogames. *Contemporary Aesthetics*, *7*.

Taylor, C. (1999). Design Template.

The User Experience Professionals' Association (UXPA). (2012). Usability Body of Knowledge.

Top 10 Best Kinect Hacks. (2011). Retrieved from http://www.kinecthacks.com/top-10-best-kinect-hacks/

van Lamsweerde, A. (2001). Goal-oriented requirements engineering: a guided tour. In *Proceedings Fifth IEEE International Symposium on Requirements Engineering* (pp. 249–262). IEEE Comput. Soc. http://doi.org/10.1109/ISRE.2001.948567

Video Game Industry Statics. (2010). Retrieved from http://www.esrb.org/index-js.jsp

Washburn Jr., M., Sathiyanarayanan, P., Nagappan, M., Zimmermann, T., & Bird, C. (2016). "What Went Right and What Went Wrong": An Analysis of 155 Postmortems from Game Development. In *38th IEEE International Conference on Software Engineering Companion*. IEEE/ACM.

Wiegers, K. (2003). *Software Requirements 2*. Microsoft Press. Retrieved from http://www.amazon.com/Software-Requirements-2-Karl-Wiegers/dp/0735618798

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B., & Wesslén, A. (2012). *Experimentation in Software Engineering*. Berlin, Heidelberg: Springer Berlin Heidelberg. http://doi.org/10.1007/978-3-642-29044-2

Zook, A., & Riedl, M. (2014). Automatic game design via mechanic generation. In *AAAI'14 Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence* (pp. 530–536). AAAI Press.

APPENDIXES

A. Improved Game Design Document (iGDD) template

[Name of the game]

Game Design Document

Created by [Name of the team]:

[Name of each team member]

[Company logo]

[Company name]

[Date]

Table of content

1 Overview

[This section is for summarizing the game and for answering important initial questions: What are the game objectives? What makes it unique? Who is the targeted audience? What is the platform for the game? What genre is the game? What is the overall gameplay? These are samples of what an overview section should have.

The goal of this section is to have a quick way to look for the main highlights of the game. A new member on a game development team can read this section to catch up on the idea of the game, or in an advanced game design stage a designer can use it to verify if the ideas he has are in harmony with the general idea of the game. The high concept document can evolve to become this section].

1.1 Game abstract

[Summarize the game in a few words].

1.2 Game objectives

[Describe the benefits to be achieved by making the game. Objectives should guide the design decisions of the game. Any constraints should be linked to objectives].

1.3 Game justification

[Describe why the game is worth doing. What makes it unique? How will it accomplish the defined goals?].

1.4 Core gameplay

[Describe the main activity the player will be doing in the game. Focus on why it will be fun?].

1.5 Game features

[This section describes the principal characteristics the game will have].

1.5.1 Genre

[Describe the game genre by defining elements or a common basic rule set that describes the nature of the game].

1.5.2 Number of players

[Establish the number of players in the game. If the game is multiplayer then describe the number of players it is intended to handle and indicate if the multiplayer game is competitive, cooperative or collaborative. Describe any special mode the game has for multiplayer].

1.5.3 Target platforms

[Describe on which platforms the game should run. This section is just to give an idea of the capabilities and limitations the game may have. Platform limitations should be detailed and described in the technical constraints section].

1.5.4 Game theme

[Describe the guidelines to the aesthetics of the game. Some examples of game themes are: post nuclear earth, Greek mythology or medieval].

1.5.5 Story summary

[Write a brief summary of the story of the game].

1.6 Player profile

[Describe the characteristics of the target player(s)].

1.7 Initial scope

[Describe the intended time, money or other resources to be spent in the game and how long the game would be].

1.8 Experiences

[Describe what you want the players' experiences to be while playing the game].

2 Mechanics

[This section describes the game elements, its attributes, and its interaction rules. All elements that create the game must be detailed and described in this section. A game character, its visual aspect, its sound effects, its personality may be described in this section].

2.1 Game elements categories

[Create game elements categories. This may help to better organize the design and to establish a solid base for reuse. Some examples of game elements categories are: enemy, boss, weapon or world].

2.2 Core game elements

[Describe the core game elements by describing their attributes and their behavior. Point out which elements can be directly manipulated by the player. Some examples of core game elements are: Mario who belongs to the game element category “player character” or Green Turtle who belongs to the game element category “enemy”].

2.3 Rules

[Describe the valid actions that the player can do and how the game should respond to these actions].

2.3.1 Interaction rules

[Describe the valid interaction between game elements and the result of the interaction].

2.3.2 Artificial Intelligence

[Describe here how the game elements should react under different circumstances in the game].

2.4 Game world elements

[Describe elements that are outside the core gameplay. Some examples of game world elements are: world map or transportation (horse, boat or car)].

2.5 Game log elements

[Describe elements that register the player progression. Some examples of game log elements can be: score, save or achievement].

2.6 Other elements

[Describe any other element that cannot be classified as any other element classification in the mechanics].

3 Dynamics

[This section describes the flow of the game: story, levels, chapters, and puzzles, interfaces (hardware and software). This section is directly related with the mechanics section since the dynamics are constructed from the elements in the mechanics].

3.1 Game World

[This section describes the world where the game is played].

3.1.1 Game theme details

[Describe the world environment, its atmosphere. Detail how the game world should look, sound and feel].

3.1.2 Game Modalities Flow

[Describe how the player can navigate through the world in the game, such as if navigation is linear or he/she can choose where to go, if he/she can skip levels or if there are restrictions on entering some areas].

3.1.3 Game detailed story

[Detail the story and point out how it is related to the game progression. Describe if the story is linear or can change, if it can change, how it changes. If the game has dialogs classify and list the script here, specify which are spoken and which are written].

3.2 Game modalities elements

[This section describes the elements that will form the core gameplay].

3.2.1 Objectives

[Describe the objectives to achieve in the dynamics of the game].

3.2.1.1 Primary

[Describe the primary objectives, which are related to the game progress. They can be goals needed to advance in the game or any victory condition which is the final objective needed to beat the game].

3.2.1.2 Secondary

[Describe the secondary goal, complementary goals, but not a requirement to progress in the game].

3.2.2 Rewards

[Rewards to the player for his actions in the game, for achieving a goal or beating a challenge].

3.2.2.1 Implicit

[Describe the implicit rewards. Implicit rewards don't impact directly on the player capabilities, but instead they are related to the player experience. An example of implicit reward is to rescue a princess in a castle and see how she thanks the player for his effort].

3.2.2.2 Explicit

[Describe the explicit reward. These have a direct impact on the player's capabilities. Some examples of explicit rewards are: a new weapon, an extra life or gold coins].

3.2.3 Challenges

[Challenges to put to the players throughout the game. Some examples of challenges are: a fight, a puzzle or a boss fight].

3.2.4 Other game modalities elements

[Add in this section any other missions/levels/chapter elements not included in the previous sections].

3.3 Game modalities description

[This section describes the flow of every modality of the game. Use the game modalities elements to construct it. Illustrate how big the area is, the background, and the events relevant to history. Try to cover every detail that will occur in every mission, level or chapter].

3.3.1 Single player modalities

[Describe the modalities that can be played by a single player in your game, details every mission level or chapter that the modality has. Use only the game and modalities elements to describe each modality].

3.3.2 Multiplayer modalities

[Describe the modalities that can be played by two or more players. If there are modalities that can be played in single or multiplayer mode, describe the differences that the modality has when playing it in multiplayer mode.

Describe how the players interact: can they communicate? how?, the kind of interaction, is it cooperative, competitive or other type?].

3.4 Game interface

[Describe every element of every screen that the player can manipulate. Some screen examples might be: title, options, main, inventory or save].

3.5 Control interface

[Describe how the player can manipulate every screen in the game].

3.6 Game learning

[Describe the way in which the player will learn to play the game. Is there a tutorial included in the game? When the player learns a new skill, will there be guidance provided on how to use the new skill?].

3.7 Game Balance

[Describe the elements that are easy to change and can be used to increase or decrease the challenges difficulty. Examples of elements that can easily balance the challenges are enemy speed, life or number of enemies in a fight].

4 Aesthetics

[This section details what the player sees and hears. This section can be extended in case of augmented reality games like the inclusion of smells].

4.1 Core game elements visual

[Describe all the visual aspects of the core game elements].

4.2 Game world elements visual

[Describe all the visual aspects of the game world elements].

4.3 Game log elements visual

[Describe all the visual aspects of the game log elements].

4.4 Other elements visual

[Describe all the visual aspects included in the elements section].

4.5 Game world visual

[Describe all the game world visual aspects].

4.6 Missions/levels/chapters visual

[Describe all the visual aspects of the missions, levels and chapters].

4.7 Special areas visual

[Describe all the visual aspects of the special areas].

4.8 Game interface visual

[Describe all the visual aspects of the game interface].

4.9 Core game elements sound

[Describe all the sound aspects of the core game elements].

4.10 Game world elements sound

[Describe all the sounds aspects of the game world elements].

4.11 Other elements sound

[Describe all the sound aspects included in the other elements section].

4.12 Game world sound

[Describe all the sound aspects of the game world].

4.13 Missions/levels/chapters sound

[Describe all the sound aspects of the missions, levels and chapters].

4.14 Special areas sound

[Describe all the sound aspects of the special areas].

4.15 Game interface sound

[Describe all the sound aspects of the game interface].

5 Constraints and assumptions

[This section describes the restrictions that the game will have which should be taken into consideration when making the game design decisions].

5.1 Technical constraints

[Describe the technical constraints. These constraints are usually derived from the platform or the game engine choices. Describe what technical limitations the game has. Some examples of technical constraints might be: no permanent memory or the platform only has multi-touch screen].

5.2 Detailed technical constraints

[Describe specific limitations derived from the technical constraints. Some examples of the detailed technical constraints are: the game cannot be saved or game cannot use colors - only black and white].

5.3 Business constraints

[Describe the business constraints. These constraints are usually derived from business decisions such as targeting kids' audiences, or trying to release before a holiday. Describe what business limitations the game has. Some examples of business constraints might be: Game classification should be for the whole family or the game should be published before December 20].

5.4 Detailed business constraints

[Describe specific limitations derived from the business constraints. Some examples of detailed business constraints are: the game cannot have blood or the game cannot have more than seven missions].

5.5 Assumptions

[Describe any design decision based on an assumption. Specify how and when the assumption should be validated].

6 Document information

[This section describes the concepts needed to understand the document such as the references to other documents used or mentioned in this document].

6.1 Definition, acronyms and abbreviations.

[Define all the concepts, acronyms and abbreviations needed for the understanding of this document].

6.2 Document references.

[List all the documents referenced by this GDD and specify where they can be found].

7 Appendix

[Add any other information or relevant document to the design of the game].

B. Game Experience Management (GEM)Proposal Description

Game Experience Management

January 2014

V.0.5

Table of Contents

List of Tables

1. Introduction

A main topic on video game development processes is how to guarantee a satisfying level of interactive experience. The experience that a game brings is commonly evaluated in the final stages of the development process, but it is advisable to avoid unexpected results such as: unfulfilled goals, by not relating the experiences to the game goals, and unaligned features with the desired experience, which may lead to wasting resources in features that may not be meaningful to the player experience. This can be avoided by specifying and designing experience in early stages of video game development process.

Establishing quality drivers and measurement factors that can be traced to the game elements in a Game Design Document (GDD) to enhance the quality of the final experience during the full development process may help to minimize unexpected results. We propose a methodology to handle user experience in games through a game development process.

2. Game Experience Management

2.1. Identify Game Design Drivers

This activity introduces the methodology and identifies the game design driver. Game design drivers are high level properties that the game should have in order to generate the intended experience.

Step 1: Proposal Presentation and Introductions

In this step, the facilitators describe the motivation for the methodology and explain each step of the method. We recommend using a standard slide presentation that can be customized to represent the needs of the sponsor.

Next, the facilitators introduce themselves and the stakeholders do likewise, briefly stating their background, their role in the organization, and their relationship to the game being built.

Step 2: Game Overview Presentation

After Step 1, a representative of the stakeholder community presents the business and/or mission drivers for the game. The term “business and/or mission drivers” is used carefully here. Some organizations are clearly motivated by business concerns such as profitability, while others, such as governmental organizations, are motivated by mission concerns and find profitability meaningless. The stakeholder representing the business and/or mission concerns (typically a manager or management representative) spends about one hour presenting

- the game’s business/mission context
- high-level functional requirements, constraints, and desired experience

During the presentation, the facilitators listen carefully and record any relevant information that may help to clarify the game design drivers. The game design drivers that will be refined in later steps will be derived largely from the business/mission needs presented in this step.

EG: The game needs to be finished before Halloween.
EG: The target platforms are tablets.

While a detailed game design might not exist, it is possible that high-level game descriptions, context drawings, storyboards or other artifacts have been created that describe some of the game details. At this point, a game designer will present the game features and game design plans as they stand with respect to these early documents. Information in this presentation may include:

- plans and strategies for how key business/mission requirements will be satisfied
- key technical requirements and constraints—such as mandated operating systems, hardware, middleware, and standards—that will drive game design decisions
- presentation of existing context diagrams, high-level game diagrams, storyboards and other written descriptions.

During this time, facilitators continue to make note of key aspects of the presentation for later reference.

EG: The game genre is survival horror.

Step 3: Identification of Game Design Drivers
During steps 2, the facilitators take notes on information regarding game design drivers that are key to realizing the intended experience in the system. These drivers may come from high-level requirements, business/mission concerns, goals and objectives.

Before undertaking this step, the facilitators should excuse the group for a 15-minute break, during which they will caucus to compare and consolidate notes taken during steps 2. When the stakeholders reconvene, the facilitators will share their list of key game design drivers and ask the stakeholders for clarifications, additions, deletions, and corrections. The idea is to reach a consensus on a distilled list of game design drivers. The final list of game design drivers will help focus the stakeholders during strategy brainstorming to ensure that these concerns are represented by the strategy collected.

EG. The game should produce a feeling of being alone.
EG. The game should produce the fear of dying all the time.

2.2. Create Game Design Guidelines

This activity generates, prioritizes and details the game design guidelines. A game design guideline is a description of how game elements need to be created in order to achieve the intended experience established in the game design drivers.

Step 1: Guidelines Brainstorming
After the game design drivers have been identified, the facilitators initiate the brainstorming process in which stakeholders generate guidelines.
A guideline may be related to game design constraints, game features or anything that the game can include to ensure the intended experience.

EG. for the game design drivers " The game should produce a feeling of being alone" there might be the following guidelines: "A level can't have more than 3 Non Player Characters (NPC)", "The level's visual environment should look abandoned".

Each stakeholder expresses a guideline representing his or her concerns with respect to the game experience in round-robin fashion. At least two round-robin passes are made so that each stakeholder can contribute at least two guidelines. The facilitators ensure that at least one representative guideline exists for each game design driver listed in Activity 1.

Guidelines generation is a key step in the method and must be carried out with care. We suggest the following guidance to help facilitators during this step:

1. Facilitators should help stakeholders create well-formed guidelines. It is tempting for stakeholders to recite guidelines, which are too general to be useful, such as “The game should prevent the player from becoming frustrated.” While this is an important requirement, facilitators need to ensure that the experience attribute aspects of this requirement are explored further. For example, the following guideline sheds more light on the efficiency-learnability aspect of this requirement: “If a player fails in the same challenge more than two times, a hint of how to overcome the challenge should be presented to the player.” Note that the initial requirement hasn’t been lost, but the guideline further explores the learnability aspect of this requirement. Facilitators should note that experience attribute names by themselves are not enough. Rather than saying “the game shall be immersive,” the guidelines should describe what it means to be immersive by providing a specific guidance on what or how create specific game elements.

2. The vocabulary used to describe players’ experiences varies widely. It doesn’t matter what a particular attribute is called, as long as there is a guideline that describes what it means.

3. Facilitators should refer to the list of game design drivers generated in Activity 1 from time to time during guideline brainstorming to ensure that representative guidelines exist for each one.

Step 2: Guidelines Consolidation

After the guideline brainstorming, similar guidelines are consolidated wherever possible.

To do that, facilitators ask stakeholders to identify those guidelines that are very similar in content. Guidelines that are similar are merged, as long as the people who proposed them agree and feel that their guidelines will not be diluted in the process. Consolidation is an important step because it helps to prevent a “dilution” of votes during the prioritization of guidelines (Step 3). Such a dilution occurs when stakeholders split their votes between two very similar guidelines. However, if the two guidelines are similar enough to be merged into one, the votes might be concentrated, and the merged guidelines may then rise to the appropriate level of importance.

Facilitators should make every attempt to reach a majority consensus with the stakeholders before merging guidelines. Though stakeholders may be tempted to merge guidelines with abandon, they should not do so. In actuality, very few guidelines are merged.

After the guidelines are merged the facilitators and the stakeholders analyze the guidelines for conflicts. This means that following one guideline may result in going against another. Any guideline in conflict can be redacted if it still satisfies the guideline objective and avoids the conflicts. All guidelines should list if they are in conflict, and with which other guidelines.

Step 3: Guidelines Prioritization

Prioritization of the guidelines is accomplished by allocating to each stakeholder a number of votes equal to 30% of the total number of guidelines generated after consolidation. The actual number of votes allocated to stakeholders is rounded to an even number of votes at the discretion of the facilitators. For example, if 30 guidelines were generated, each stakeholder gets 30 x 0.3, or 9, votes rounded up to 10. Voting is done in round-robin fashion, in two passes. During a pass, stakeholders allocate half of their votes. Stakeholders can allocate any number of their votes to any guideline or combination of guidelines. Following the example, on the first pass, each stakeholder will distribute 5 votes in the guidelines he or she consider more important and can place more than one vote in a guideline. By the end of the first pass all stakeholders will have placed half of his or her votes. For the second pass all stakeholder will distribute the rest of his or her votes. The votes are counted, and the guidelines are prioritized accordingly.

Step 4: Guidelines and Game Elements Relation

Each guideline description should make clear which game elements it affects. However, it is important to explicitly document the relation. A Game Design Document (GDD) or other similar documents may be used to establish these relations. For example, for the guideline "A level can't have more than 3 Non Player Characters (NPC)", the guideline should be related to the section on the GDD where the levels of the game are described. In the guideline "The level's visual environment should look abandoned" the guideline should be related to the section in the GDD where the visual environment of the level is described. All guidelines should be related in a similar way.

2.3. Guideline Validation

Step 1: Guideline Validation

This step validates that the guidelines are followed. Whenever a game element is finished, a game designer or a QA member checks that the elements of the game follow those guidelines with which they are associated. Where a guideline is not followed, the reason why it was not followed must be justified, or the related game elements must be adapted to follow the guideline. A guideline is considered validated when all game elements related to it are approved. As an example, in the guideline "The level's visual environment should look abandoned" will be validated when all the environments in a level are approved. A validated guideline does not mean that all elements related to it follow the guideline, it means that all elements follow the guide or have a valid justification. For the guideline mentioned, all the environments in the level, except the last, looks abandoned, but the last environment in the level has the following justification "The last level in the game does not take place in the ghost town, it takes place in hell. Therefore the last environment on this level must be different." and the justification is approved, so the guideline is validated even if not all related elements follows it.

2.4. Generate Test Cases

In this step the test cases for the guidelines are created. Three sets of test cases need to be created.

Step 1: Guidelines Goal Test Cases

The first set focuses on validate that the game elements related to the guideline are achieving the guideline goal. For each guideline at least one test case needs to be created. As an example in the guideline "The level's visual environment should look abandoned", there should be at least one test case that focus on evaluating if the player considers the visual environment abandoned. The test case may consist of a questionnaire or part of it; biometrics and its relation to parts of the game; a video of facial expressions of the players as they play and their interpretation; or metrics collected by the game while playing.

This set of test cases can be applied once all elements referring to a guideline are finished, on a prototype or the complete game.

Step 2: Game Design Drivers Test Cases

The second set of test cases focuses on evaluating if the properties of the game described in the game design drivers are achieved. For each game design driver there should be at least one test case. As an example in the game design driver "The game should produce a feeling of being alone" there should be at least one test case that focuses on evaluating if the player feels alone when playing the game.

It is advisable to evaluate the guidelines related to the game design driver to see if the guideline is helping to achieve the game property. For example for the aforementioned game design driver " The game should produce a feeling of being alone" there may be a test case that evaluates if achieving the guideline "The level's visual environment should look abandoned" goal, contributes to making the player feel lonely. And so on for each guideline related to the game design driver.

This set of test cases can be applied to a prototype or the complete game, given that they evaluate properties of the game, not game elements.

Step 3: Game Experience Test Cases

The third set focuses on measuring the overall experience that the game generates. This set should focus on finding the negative and positive experiences so they can be compared with the intended experiences defined in the game design drivers.

This set of test cases can be apply to a prototype or the complete game, given that they evaluate properties of the game, not game elements.

2.5. Execute Test Cases

Step 1: Execute Test Cases

This step executes the test cases. The execution and expected result of each test case may be very broad. The execution may be: to apply questionnaires to a focus group that plays only some sections of the game; or letting the game collect various metrics while testers play some parts of the game. The expected result may be: a specific output of a questionnaire, or a minimum number in a specific metric collected while the testers were playing a specific part of the game.

When using questionnaires or something similar as evaluation tools on the three sets of test cases and the group responding to the questionnaires is the same, more accurate results will be achieved by starting with the overall experience test cases, then continuing with the guideline test cases and finally finishing with the game design driver test cases. The results of the evaluation of the overall experience will not be influenced by the prior application of the intended experience evaluation described
in the game design drivers test cases.

2.6. Evaluate Analysis and Feedback

Step 1: Evaluate Guideline Test Cases

This step, using the results of the test cases, analyzes the degree of goal achievement based on the guidelines.

Step 2: Evaluate Game Design Rivers Test Cases

This step analyzes if the game design drivers brought the intended properties (experience) to the game depending on the test cases results.

Step 3: Evaluate Overall Experience Test Cases

This step analyzes the overall experience that the game brought to the player based on the test cases' results.

Step 4: Define Required Changes

This step, if required, defines the changes that the game needs to get closer to the intended experience. The defined modifications might be in: the game elements, guidelines, game design drivers or even changing the goals due to the analysis of the results of the test cases.

3. Conclusion

User experiences in video games are key on meeting the game goals. These experiences must be recognized and considered early in the game development life cycle, and the video game must be designed so that their intended experiences are met.

To do that, the game design drivers must be understood. This proposal, describes a method for eliciting and explicitly documenting players' experiences through drivers early in the development process, provides a forum for stakeholders to come to consensus about these drivers.

User experience is related to goal in the game design drivers as a property of the game. The game design guideline deconstructs these game design drivers into criteria that the game elements need in order to obtain the intended experience. The user experience is evaluated using test cases as initial point, even if there are other tools used for evaluation, test cases are a good starting point to have standard documentation in order to evaluate user experience.

Appendix A. Roles and Templates

Facilitator: The facilitator ensures that the activities and steps of the methodology are carried out. He or she facilitates discussions, and ensures that the methodology is carried out in a timely fashion and that the required artifacts are produced. This role is most effective when assigned to a experienced game designer

Stakeholders: People involved on the game development who participate in the activities of the methodology. A stakeholder may have many roles: documenting, validating, creating or executing, among others.

Table 0-1: Game Design Driver Table

Game Design Driver #	Goal id. relation	Experience attribute related	Description

Table 0-2: Raw Guideline Table

Guideline #	Game Design Driver # related	Description	In conflict with (other guidelines)	Votes

Table 0-3: Guideline Refinement Table

<table>
<tr><td colspan="2">Guideline Refinement for guideline #</td></tr>
<tr><td>Description</td><td></td></tr>
<tr><td>Game Design Driver # related</td><td></td></tr>
<tr><td>In conflict with</td><td></td></tr>
</table>

(other guidelines #)		
Priority (based on votes)		
Game elements related		
Validation date		
Game elements not following the guideline		
Game element ID	Justification	Approved (yes/no)

Table 0-4: Test Case Table

Test Case #						
Evaluating guideline #/game design driver #/ overall experience						
Created by						
Creation date						
Executed by						
Execution date						
Preconditions						
Test case description						
Step	Description	Expected result	Actual result	Pass/Fail	External source reference	Comments

C. Core Elements of the Gaming Experience (CEGE-Questionnaire)

Overview: This questionnaire is used to assess the core elements of the gaming experience. Each item is rated with a 7-point Likert scale. The questionnaire is to be administered after the participant has finished playing with the game.

Scales: There are eight scales in the questionnaire: CEGE, Video-game, Puppetry, Game-play, Environment, Control, Ownership and Facilitators.

Reliability: The Cronbach alpha for the whole questionnaire is 0.794 and for the CEGE scale is 0.803.

Instructions: Please read the following statements and answer by marking one of the numbers that best describes your experience.

No.	Question
1	I enjoyed playing the game
2	I was frustrated at the end of the game
3	I was frustrated whilst playing the game
4	I liked the game
5	I would play this game again
6	I was in control of the game
7	The controllers responded as I expected
8	I remember the actions the controllers performed
9	I was able to see on the screen everything I needed during the game
10	∗The point of view of the game that I had spoiled my gaming
11	I knew what I was supposed to do to win the game
12	∗There was time when I was doing nothing in the game
13	I liked the way the game looked
14	The graphics of the game were plain
15	∗I do not like this type of game
16	I like to spend a lot of time playing this game
17	I got bored playing this time
18	∗I usually do not choose this type of game
19	∗I did not have a strategy to win the game
20	The game kept constantly motivating me to keep playing
21	I felt what was happening in the game was my own doing
22	I challenged myself even if the game did not require it
23	I played with my own rules
24	∗I felt guilty for the actions in the game

25	I knew how to manipulate the game to move forward
26	The graphics were appropriate for the type of game
27	The sound effects of the game were appropriate
28	∗I did not like the music of the game
29	The graphics of the game were related to the scenario
30	The graphics and sound effects of the game were related
31	The sound of the game affected the way I was playing
32	∗The game was unfair
33	I understood the rules of the game
34	The game was challenging
35	The game was difficult
36	The scenario of the game was interesting
37	∗I did not like the scenario of the game
38	I knew all the actions that could be performed in the game

∗Denotes items that are negatively worded.