Universiteit Utrecht

A Thesis submitted in partial satisfaction of the requirements for the degree
Master of Natural Sciences

in

GAME AND MEDIA TECHNOLOGY

Design and Development of a Game Production Control Center

Author:

Stavros Tsikinas

ICA-3938131

Committee in charge:

Dr. Fabiano Dalpiaz
Dr. ir. Frank van der Stappen

July 2015

Table of Contents

Acknowledgements v
Abstract vi
1. Introduction 1
1.1 Motivation 1
1.2 Research Questions 2
1.3 Scope of the Thesis 3
1.4 Contributions 4
1.5 Grendel Games 4
1.6 Thesis Overview 4
2. Literature Study 5
2.1 Video Game Development 5
2.2 Software Project Manager and Game Producer 8
2.3 Game Testing and QA 11
2.4 Game Production Tools 14
2.5 Business and Game Analytics 15
2.6 Conclusions 15
3 Design of the Solution 17
3.1 The Conceptual Model 17
3.2 Metrics of the Conceptual Model 19
3.3 Conclusions 28
4 Prototype 29
4.1 PivotalTracker 29
4.2 Technology 31
4.3 Dashboard 33
5 Case Study 36
5.1 Preparation 36
5.2 Data Sets 38
5.3 Results 40
5.4 Enhancements 43
6 Discussion 47
6.1 Conclusions 47
6.2 Limitations 47

6.3 Future Work 48
7 References 49
8 Appendix 51
The Microsoft Access Diagram 51

Acknowledgements

I would like to thank Dr. Fabiano Dalpiaz and Dr. Frank van der Stappen for their support throughout this thesis as the chair of my committee. They have dedicated many hours to guide me conduct the thesis and their guidance has proven to be more than valuable.

Furthermore, I would like to thank Grendel Games for accepting me as an intern in a very demanding position. Their assistance and guidance was invaluable as field experts. Also, they provided me with some very valuable information that are presented in the thesis.

Also, I would like to thank the people that assisted in the completion of this thesis. Derk de Geus, Remmelt Blessinga and Jan Jaap Severs, who participated in the creation of the game producer's mind-map. In addition, I would like to thank Tristan Lambert, Anne Draaisma, Guido Soetens, Maarten Stevens and Paul Brinkkemper for participating in the case study and helped me evaluate the prototype console.

Also, I would like I would also like to thank my life partner Eva, for being next to me during these stressful months and mentally supporting me in the process of editing the thesis.

Finally, I would like to thank my family, for all the support they have provided me with, during these years that I live in the Netherlands.

Abstract

Game producers face many obstacles, in order to perform effectively their tasks within a game studio. In this project, we present a prototype console that aims to assist game producers in managing the agile game development process. The console is based on a number of game process metrics that make the game development management more effective by providing the game producer with objective, up-to-date information. We evaluate the effectiveness of the prototype console and the metrics through qualitative interviews with experts from the game industry.

1. Introduction

Video games have been introduced in society over 50 years ago [1, 2, 3]. Many games with different genres (adventure, RPG, strategy, sports, etc.) and purposes (entertainment games, educational games, health games, etc.) have been developed by many game studios. Video game development is described as the process of software development, from which a video game is produced [4]. Currently, with the expansion of gaming platforms (PC's, consoles, mobile devices), video game development has become one of the most trending and successful industries [5, 6, 7].

The development team of every game, or game studio, consists of multiple skillful members. Those members belong to many different disciplines, such as programmers, designers, artists, testers and audio engineers. In order those teams to have good understanding and communication, and the development process to be balanced between creativity and productivity, another team member is important; the game producer.

A game producer is the person responsible of scheduling, budgeting, overseeing and managing the development process of a game [8]. Due to all these responsibilities, the role of a game producer is very demanding. That is the reason why big game studios have hierarchies of game producers, namely Executive Producer, Producer and Associate Producer or Junior Producer.

Our ambition on this thesis is that – through the use of dashboards on the most important metrics of game design and development – game producer will be able to accomplish their work more efficiently and effectively.

1.1 Motivation

As mentioned previously, there are many games developed and published daily. On the other hand, there are many games that failed to be developed or shipped and, therefore, were cancelled. The reasons that development of games is cancelled can vary; financial issues [9] or loss of target group interest [10] or low quality [11]. Other titles get cancelled because employees are transferred to other game titles [9, 12] or there planning is too optimistic (poor scheduling) [13]. Many examples exist of game projects that failed due to poor management, including:

- NBA Live 2013. Electronic Arts was very optimistic about the specific game title, by adding 'some of the best new technology", according to EA Sports executive vice president [14]. However, due to poor scheduling, he stated that "We knew it was going to be a long journey and we're just not there yet" and the game had to be cancelled. Eventually, the game was cancelled 6 days before its initial release date [15].

- Ashes Cricket 2013. It was stated as "one of the worst games of 2013 [16, 17]. The game studio, Trickstar Games was developing the game title, but the process was rather slow and the quality was low. The game managed to be published, only for PC version, while it was initially targeted for Xbox 360 and PlayStation 3 [16].

- Prey 2. Yet another game title that started with high potential, but was unable to be shipped on time and with the appropriate quality. So, after years of postponing the release and updates regarding the progress of the game, the publisher stated that "It was a game we believed in, but we never felt that it got to where it needed to be – we never saw a path to success if we finished it. It wasn't up to our quality standard, and we decided to cancel it. It's no longer in development" [18].

- Lord of the Rings: The White Council. A role playing video game, aimed for PlayStation 3, Xbox360 and PC. A game destined to depict the trilogy written by J.R.R. Tolkien.

Electronic Arts was the publisher and the game started development in June 2006. Although it was described as "the innovation and quality of the next-generation RPG" [19], in 2007 the game was put on hold and eventually cancelled, due to management and scheduling problems [20, 21].

These examples present problems that exist in game studios regarding the game production and the results that a poor game production can have. Therefore, the motivation of this thesis is important in helping the game producers handle the game development process in a more effective way. This thesis is aimed to propose a software solution that would assist game producers from the beginning of the development until the end of the project.

The terms game development and game production are closely related, however there is an important difference between these two terms. Game production corresponds to the activities that are performed from defining the concept of a game until its release [22], whereas game development focuses solely on the activities made by the development team to complete the art, design and programming tasks of the game [4].

1.2 Research Questions

In order to investigate and explore the field, there are certain questions that need to be addressed. These questions are aimed to evaluate the research that this thesis is dealing with. The research question consist of a main question and four sub-questions:

How to facilitate, with a software solution, a Game Producer in managing the game development process?

The scope of the thesis focuses on exploring how the game development process can be managed with a software solution and, therefore, the options that do not involve the use of software were not considered.

Sub-Question 1: *What are the key responsibilities of a Game Producer?*

Game producers have multiple responsibilities, during the game development. On this question we try to investigate, from the literature study, the responsibilities that are the most important for a game producer.

Sub-Question 2: *What are the problems that come from a non-coordinated game development?*

Many aspects of game production need to be handled by the game producer, otherwise the games may fail or even cancelled. This question identifies, through literature study, the problems that can occur if the game production is not managed properly, or even at all.

Sub-Question 3: *What are the metrics to assess the effectiveness of coordinating the game development?*

Game development teams need to be effective, in order to produce the best possible outcome, with respect to the schedule made and the milestone that have been set from the management team. This question aims to determine the factors that a game producer needs to take into account, when monitoring the process of game development.

Sub-Question 4: *What is an effective software solution to support the game producer's coordination?*

This question is aimed to identify what effective game production is and how a software solution can help game producers evaluate the effectiveness of game development.

1.3 Scope of the Thesis

Game production, depending on the size of the game studio, covers a broad range of activities. Figure 1 presents a visualization of the game production activities. The arrows represent the activities that are directly related.

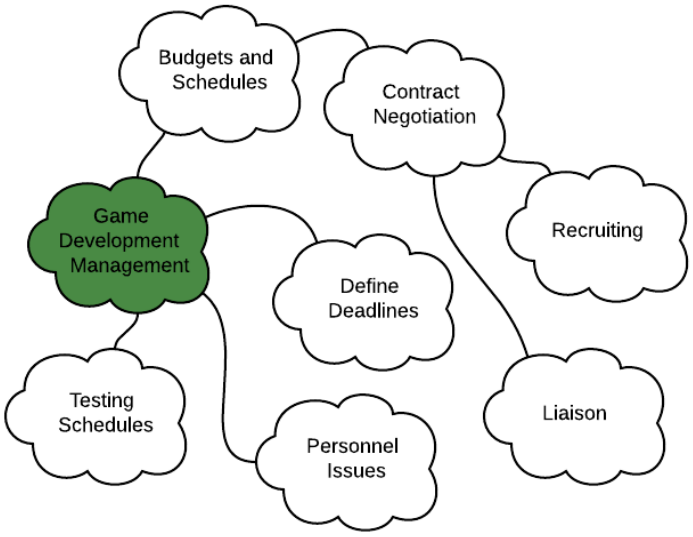

Figure 1: The game production tasks.

The most common activities consist of [8, 22, 23]:

- Game development management. This is an activity performed by the game producer that deals with tracking the progress of a game development and overseeing the development team on daily basis.
- Creation of budget and schedules. This activity is performed by the game producer and the development team, aiming to develop the schedules in which the team will follow for the completion of the game project.
- Scheduling of Testing. This activity is performed by the game producers, in collaboration with the testing leads, and concerns the testing phases and the testing schedules for a game development.
- Liaison between development team and stakeholders. This is an activity performed by the game producer, in order to assist the development team and the stakeholders walk on the same path for success.
- Recruiting personnel. This activity is performed, in order to hire staff that fit with the company needs.
- Negotiating contracts. This activity covers all the negotiations held during the production of a game. The negotiations can be about licensing, outsourcing and localization.
- Cope with personnel issues. This is an activity that the game producer performs, in order to keep the development team positive and productive. It aims to enforce discussion between the team members to solve impediments.

- Define deadlines. This activity is performed by the game producer and all the stakeholders, in collaboration with the development team. In association with the budget and the schedule activity, deadlines on phases and milestones are created.

For this reason, this thesis will focus on a specific part of game production, namely tracking the game development process. The rest of the activities of game producers will be excluded from the scope of the thesis.

1.4 Contributions

There are certain contributions that this thesis is trying to achieve in the game production field. First of all, a main contribution on this thesis is to propose a solution to game producers through a console, which will assist them in managing the process of game development and evaluating the progress that the development is making.

Another significant contribution of this thesis is the creation of an amount of metrics, by which the game producer can obtain important information regarding the progress of the game production. These metrics are proposed, by exploring the field of game production, from the scope of game development, and then evaluated by game producers together with the console, in order to assess the contributions of the thesis.

1.5 Grendel Games

During the completion of the thesis, I was occupied by Grendel Games as an intern for 8 month. During the internship, I was performing the tasks of a game producer and I was able to have an insight on what is actually happening in the industry. The company's role was to provide us with data that we could obtain to determine the metrics and the console that we developed. That is the reason that Grendel Games for the rest of the thesis will be mentioned as **case company**.

1.6 Thesis Overview

Chapter 2 consists of the literature study, where the theoretical approach of the research, made for the thesis, is being presented. On that chapter, all the fields that have been explored on the literature study are reviewed, such as the game development, game testing and quality assurance (QA), the role of the project manager and the game producer and business/game analytics.

In Chapter 3 the way that the findings of the literature study are converted into a model is presented. Also, new metrics are being introduced, regarding the game production process from the game development scope.

Chapter 4 presents the prototype of a console that aims to assist game producers in managing game development, by using the metrics described on the previous chapter. In this chapter the technical elements of the console are introduced.

Chapter 5 consists of the case study, where scenarios of the console execution are described, executed and evaluated by field experts. This evaluation is closely related to the effectiveness of game production.

The last chapter, Chapter 6, is the discussion, in which the results of the solution are being discussed. Also, the limitations of the research are being presented and the possible future work on the field is mentioned.

2. Literature Study

This chapter of the thesis aims to present the fields that have been explored, related to the research made. The fields that this chapter introduces are game development, game testing and quality assurance, project manager and game producer, game production tools and finally, business and game analytics.

The purpose of exploring the game development, game testing and game producer field, relies on the fact that they fall under the game development management activity. The reason the game production tools are researched is to identify the current state-of-the-art software solutions for a game producer. Lastly, the purpose of exploring the business and game analytics is to present what is the current trends of analytics that companies use.

2.1 Video Game Development

Video game development is the process of creating a video game. As mentioned in the introduction, the development team consists of members belonging in different disciplines, namely programmers, designers, artists, audio engineers and testers. Although every game development team or game studio has its own scheduling and time tables, the most common game development process starts with planning and ends with the maintenance.

A game title can stay in development for a few days [24] till many years [22], depending on the quality and the size of the game and the target console. For example, GTA V, by Rockstar Games, developed for PC, PlayStation3 and 4, Xbox360 and Xbox One, was developed from 2009 until 2015. In contrast, Flappy Bird was developed by a single developer in a couple of days [24].

Another important aspect of developing a game is the budget that the game studio has set, during the planning process. The budget to develop a game varies due to multiple reasons. The first reason is the development team. For some game titles one person is enough to develop a successful game [24]. On the other hand, AAA games (games with high quality in graphics, music, promotion, etc.) can occupy a development team of more than 600 members [25, 26].

Another reason is licensing. As mentioned before, developing a game for next-gen game consoles, such as Xbox One and PlayStation 4 takes more time than developing a game for a mobile device. The cost in development varies as well and not just because of the time period in development. The games developed for consoles require to have their software development kit (SDK) [27, 28]. Although to develop a game in mobile devices is a bit more straightforward, there are also licensing expenses that the developers or game studios need to take into account [29].

Most of game studios follow the same stages in the development of a game. The first stage is *pre-production*. This is the first stage of the development, where the team develops an idea of the game. In this stage the necessary documentation regarding the game concept is formed. Important documents on this phase is the game design document (GDD), which is a document consisting of all the aspects and requirements of the game that need to be developed, such as story, characters, gameplay, art, controls and audio [30, 31]. Also, in *pre-production*, the game plan is constructed. Finally, the development team constructs multiple game prototypes, aiming to identify what is more suitable and more ‘fun’ for the end user [31].

As soon as the development team and the stakeholders (publishers and executive staff of the game studio) have come to an agreement of what are the requirements of the game, the next stage begins, the *production*. This is the most important development stage, since all the requirements and features agreed on the GDD are developed [31]. Every team member works according to the GDD, in order to develop an attractive and successful game title.

When the development team has created all the assets and necessary features, there is a stage of polishing and perfecting the project. This stage is called *post production*. The game studios use these stages to plan the development of a game.

Throughout the game development there are specific events that are of great importance, called milestones. Most of the occasions, the milestones are at the end of each stage. The most common milestones are [22]:

- *Alpha*. This is the milestone where the gameplay is functional, there are some assets completed and the game is able to run on the target platform.
- *Beta*. In this milestone all the features of the game are present and there are only bug fixing activities. Also, on this milestone, there cannot be any changes in the concept of the game.
- *Code Release*. This is the final milestone of the game development, in which the development team can ship the game to the publishers for approval.

There are many methodologies of developing software. Waterfall methodology is the oldest in software engineering [32]. It is a methodology that allows the development to occur in phases that follow one another. After the last stage is being processed and executed, the whole development loops back at the beginning. The waterfall process has been introduced to the video game development as well. Figure 2 depicts an example of the phases that are executed during the development of a video game or a piece of it.

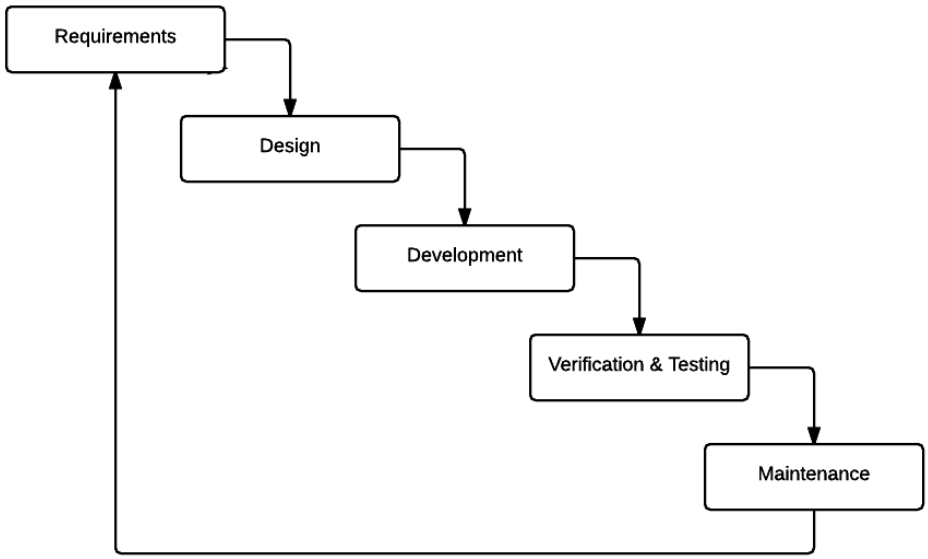

Figure 2: The stages of a waterfall game development.

There are game studios that follow this development model, because waterfall method is "mandating milestones such as game design documents and deliverables such as assets or functionality on a timetable" [33]. The main reason, as stated, is that "waterfall tends to give the illusion of certainty and is the easiest method to explain at the executive level" [33]. On the other hand, there are many game development teams and studios opposed to the waterfall methodology. The main reason is that because of its structure, there is no room for adjustability and changes [33, 34].

Because of this drawback, which limits the exploration of finding the "fun factor", game development studios and teams started using more flexible processes. One very common and effective process of developing games is agile development [31, 35]. This is a process that consists of iterations (most commonly two or four weeks), which allows the development teams produce on each iteration a game, or a piece of it, that is able to be executed, demonstrated, evaluated and tested. In Figure 3 the overview of agile game development is presented. Each iteration consists of different activities that need to be made, in order to achieve the best outcome.

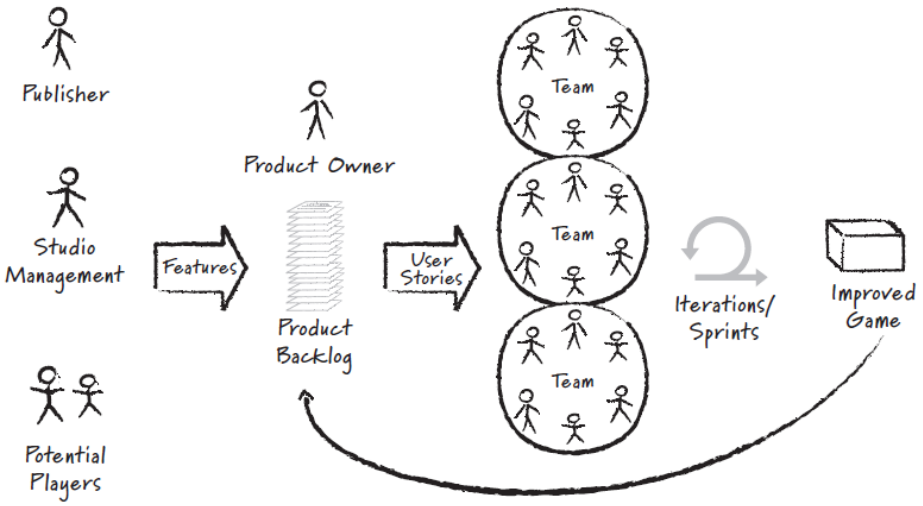

Figure 3: The flow of agile game development.
Agile Game Development with Scrum by Clinton Keith, p. 31 (2010).

As it can be extracted from Figure 2, an advantage of agile development is that the product is being developed in two- or four-week increments and the stakeholders can visualize the progress that the product is making. By this, the development team can evaluate whether the game is heading towards the right direction that the studio and the publisher have agreed upon. Also, the agile development is much more flexible than the waterfall development, because with agile, the development goals can change more often and can meet the stakeholders' needs. An iteration of developing a game feature that needs to be improved in early stage is feasible and the cost that is required to do so is limited.

One very popular, in game development, agile methodology is Scrum [31, 35]. This methodology uses iterations of two or four weeks, known as ***sprints***. An overview of scrum is presented in Figure 4.

In order to implement scrum, a game development team needs to identify the scrum parts [31]. The first and most important part of scrum are the ***sprints***. Each ***sprint*** has a specific goal, named as ***sprint goal***, which all the team members are trying to achieve by the end of an iteration. When this is over, the development team presents the outcome of the ***sprint goal*** to the stakeholders. Another important part of scrum development is the ***product backlog*** that is a prioritized list, consisting of all the requirements and features that are needed to develop the game. The ***product backlog*** is not a static list that stays uninterrupted throughout the development. In every *Sprint* the needs, requirements might change and the same can occur to the ***product backlog***.

A subset of the ***product backlog*** is the ***sprint backlog***, which consists of detailed tasks for the development team throughout the ***sprint***. In agile development these tasks are known as ***stories*** and will be called as such for the rest of the thesis. Apart from the parts of scrum, there are also certain

events that take place during a ***sprint***. *Sprint Planning* is the first event in an iteration. In this meeting the ***sprint goal*** is determined by the whole team. Also, the ***sprint backlog*** is constructed.

In order for all the development team members to keep track of the daily progress, scrum uses another event; the *Daily Scrum*. It is a 10 to 15 minutes meeting, where all the team members explain what have done the previous day, what would do that day and if they have encountered any problems. This meeting is encouraging collaboration and problem solving activities for the members.

At the end of the ***sprint*** there is the last event of ***scrum***. This event is the *Sprint Retrospective*, where the development team evaluates the progress of the development and investigate how the progress can be improved. Because ***scrum*** should use the "inspect and adapt" policy, *Sprint Retrospective* is claimed as the "most important event in agile development" [31, 36].

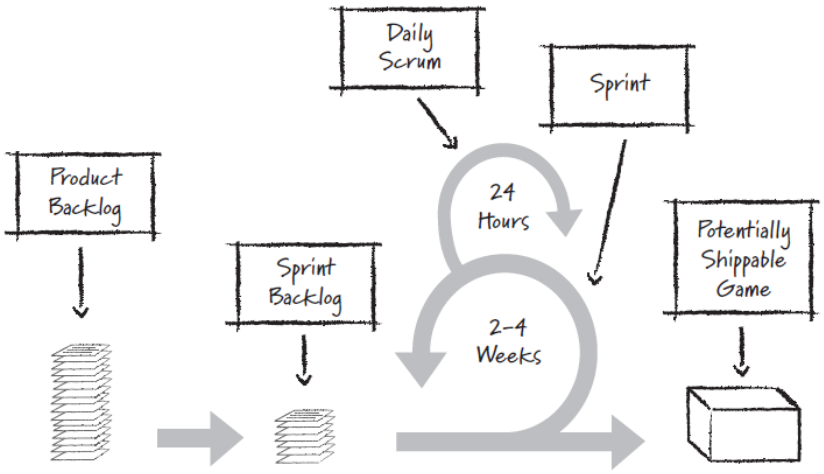

Figure 4: The scrum process.
Agile Game Development with Scrum by Clinton Keith, p. 39 (2010).

2.2 Software Project Manager and Game Producer

As mentioned in the introduction, video game development is a software development process. In order to develop successful software solutions, the development team members need to have good communication and coordination. Because of that the software engineering field has introduced a specific role; the software project manager. This person is responsible for managing the progress of the software development and takes part in multiple activities [8, 22, 23, 31].

In more traditional software development methods, such as the waterfall methodology, the role of the project manager is to lead the development team, aiming to produce a software that is accepted by the stakeholders. However, in more modern development methods like agile and scrum, the role of the project manager changes. In that occasion, the project manager is the person who oversees the development process and collaborates with the development team to make decisions.

In video game development, the role of project management is typically assigned to the game producer. A game producer has certain activities that needs to fulfill, in order to assist the rest of the development team focus more on the development of the game.

A primary activity of a game producer is scheduling. The game producer, in association with the development team, creates the long-term plan of the game [22, 37]. On the game plan there are the estimates from all the development members and the milestones that the team has set for the production of the game. The responsibility of the game producer on this task is to collect the

estimates from all the members and produce a viable schedule that the team members can follow and is within the budget allocated for the specific game project.

Another important activity of the game producer is to create the game project budget and present it to the game studio administration [22, 37]. The budget of the project is created taking into account mainly the scope of the project and the resources needed to develop the project (programming team, design team, art team, audio team, outsourcing and licensing).

A game producer is also responsible of acting as a liaison between the development team and the stakeholders [22, 37]. By this, the stakeholders have frequent updates of the project progress. In addition, the game producer ensures that the development team is able to meet the deadlines of deliverables (milestones) that have been set at the game plan. In order to achieve it, the producer is keeping track of the tasks that need to be done and compares the estimates with the actual time spent.

Also, the game producer is updating the planning if there are any important changes to meet the current plan, taking, also, into account the budget changes. This is the most demanding task of a game producer, because the progress of the game development needs to be observed daily and if any bottlenecks are detected, internal or external, they need to be tackled as soon as possible. Otherwise, the development of a game can be stalled or even cancelled [15, 20, 21]. Also, if the game studio is developing multiple games simultaneously, then this task of the game producer becomes even more demanding and nontrivial.

These activities can also be noticed in every software development, but there are activities that distinguish the role of the game producer with the project manager. An important task of a game producer is to encourage the communication between development members of different disciplines (art, engineering, design and audio). A video game development team consists of people of different disciplines and, therefore, different way of thinking. There are occasions were team members of different disciplines get into conflicts [37]. The role of the game producer is to encourage the discussion in these situations and be the intermediate in finding a solution and eventually resolve the conflicts.

Lastly, the game producer is responsible for scheduling the Quality Assurance (QA) together with the lead testers. In large projects the game producer in association with the lead tester create the test plan, a document stating all the important information the testing team needs to know, regarding the testing methods corresponding game features and the goals that the testing team needs to accomplish [38].

Figure 5 presents a mind-map of all the game producer's activities and responsibilities. The mind-map has been constructed by creating an initial version from literature and personal involvement as a game producer. After this version was constructed, it was delivered to game producers and through iterative process the final mind-map was developed.

In agile development, the game producer can also serve the role of the *Scrum Master* [31]. The *Scrum Master* is the person making sure that the development team does not encounter any obstacles in reaching the goals and also assists the team follow the process of scrum as it should [31].

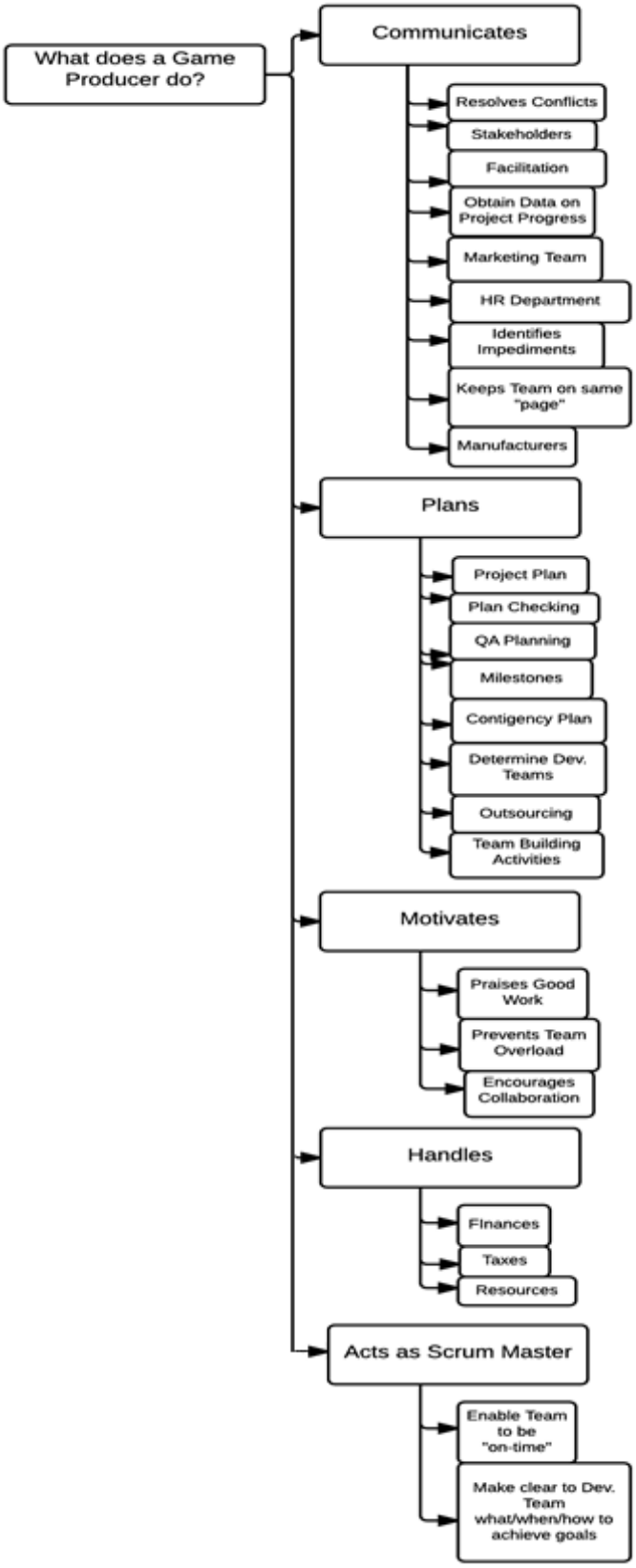

Figure 5: Game Producer's Mind-map.

2.3 Game Testing and QA

The process of game testing, known also as quality assurance (QA), is a very necessary stage of the game development [39]. During the pre-production phase, the development team with the stakeholders construct a set of requirements that the game needs to fulfill. Game requirements is a set of necessities that the game must support, the constraints that the game should have and finally, the appropriate documentation, such as the GDD [22]. However, the game requirements are a non-static set and they might change during development [40].

The complexity of game development has risen, making the need of quality game testing even more important and necessary [39]. From pre-production until the end of the project the project is being tested by the testing team, according to the test plan. By this, the game producer and the development team is assured that the project is heading towards the proper route and the quality of the game is as it should. Also, with game testing the development team and the game producer evaluate whether the game meets the requirements set on the pre-production and offer the experience the target group is expecting [38].

Figure 6 presents the cycle of game testing. The process begins with the development team delivering the feature to be tested. According to the test plan, the tester tests the feature using a specific testing technique. After the testing has finished, the testers report if there are found bugs, i.e. malfunctions of features. As soon as the game producer receives the reports, prioritizes the bugs according to the severity and necessity and assign them to the appropriate development member to perform the necessary fixes [22, 38, 39].

In agile game development, the QA process occurs daily by the development team members, by testing the feature they worked on [31]. However, the importance of the testing team is not restricted. The testing team belongs to the development team, so it is present in all the agile activities, such as sprint planning and sprint retrospectives [31].

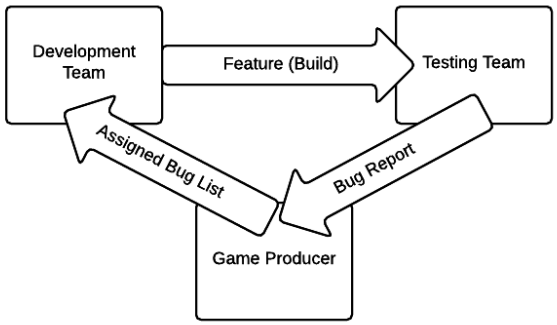

Figure 6. The game testing cycle.

As mentioned before the testing process is using certain testing methods, or testing techniques [22, 38, 39]. There are many testing techniques in the software engineering field, however there are some techniques that are more productive in game testing [39].

Combinatorial Testing is a technique that tests game elements in small sets, in order to determine which combinations can cause bugs. It is a very effective technique with the ability to identify many bugs fast and efficiently. The generation of a combinatorial test is constructed as a table, as shown in Figure 7.

Gender	Light Saber
Male	1H
Male	2H
Female	1H
Female	2H

Figure 7: An example of a pairwise combinatorial test.
Game Testing All in One by Charles P. Schulz, Robert Bryant and Tim Langdell (2005).

Test Flow diagrams (TFD) is another testing technique used often in game testing. This technique is based on models that correspond to the behavior of the game from the player's point of view. In order to create testing scenarios, the TFD technique uses certain maps, known as flows, which assist in executing the testing process. An example of a flow is depicted in Figure 8.

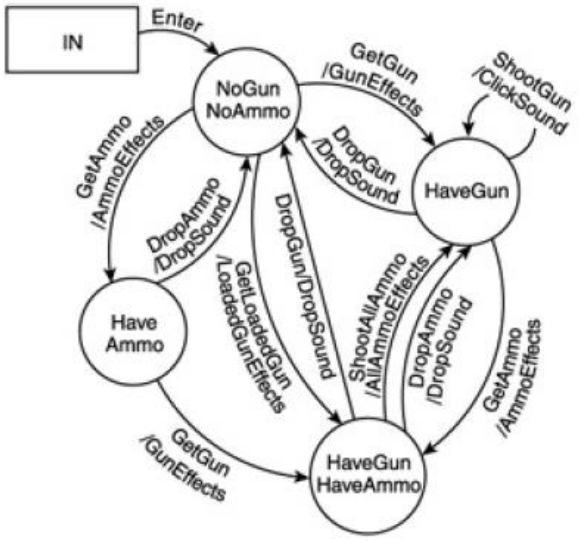

Figure 8: A TFD map.
Game Testing All in One by Charles P. Schulz, Robert Bryant and Tim Langdell (2005).

Another game testing technique is the *Test Trees*. This testing method uses trees in order to break down complex game features in separate game elements that are easier to be tested. The most common game testing technique is *Playtesting*. This technique is used to test the game generally and not specific game elements. The importance of playtesting lies in the fact that it aims to evaluate the "fun factor" [39]. So, the tester is behaving as a potential player and not as a tester. That is the reason why this testing technique is mainly focusing in aesthetics, rather than facts. *Ad-Hoc Testing*

is a non-structured testing technique with the testers explore the game generally without any pre-determined rules. It can appear in two different states; as *Free Testing*, where it occurs with no rules and scenarios, and *Direct Testing*, which is a technique aiming to test an exact game feature or element.

On Table 1 we determine which testing techniques are most suited for which game artifacts, according to [38, 39]. This table has been constructed, iteratively. Firstly, the game testing field was explored and the matching to most of the game features and the testing techniques was done. After the initial construction, the case company was observed daily on how the testing process was constructed and the table was completed.

Game Features	**Testing Techniques**					
	Combinatorial	*TFD*	*Trees*	*Playtesting*	*Free*	*Direct*
Artificial Intelligence			✓	✓	✓	✓
Animations				✓	✓	✓
Dialogs				✓	✓	✓
Game Balance				✓		
Game Controls	✓			✓		
Game Design				✓	✓	✓
Game Difficulty				✓	✓	✓
Game Graphics				✓	✓	✓
Game Modes			✓			
Game Paths		✓				
Game Rules			✓			
Game Settings	✓		✓			
Game Story				✓		
Gameplay		✓				
Online Features			✓			
Parallel Choices	✓	✓				
Special Effects				✓	✓	✓
Text				✓	✓	✓
User Interface			✓			✓

Table 1: The association between game features and testing techniques.

Combinatorial testing is closely related to game features that depend on variables and thus it is more used by programmers [39]. The game features associated with the TFD testing are related to the behavior from the point of view of the player [39]. The test trees technique is associated with more complex programming game features [39]. The rest of the testing techniques have been associated with the game features arbitrarily. However, it is understood from the literature [38, 39] and investigating the testing process of the case company, game features that are related with art are more suitable with these techniques.

After the testing process is over, the QA team reports defects that have been found during testing. The game producer, in order to prioritize the bugs and distribute them to the development team, needs to obtain information regarding each bug. A very important attribute in the bug report is the *Bug Severity*. Although the game producer and the lead tester can determine the bug severity levels differently, the most common categorization is [4]:

- *Critical*, the highest level in the hierarchy, because a bug categorized as critical causes the game to crash and not respond.
- *High*. These bugs do not crash the game, however they are important to be fixed.
- *Medium* level bugs belong to bugs that are importance, but can be postponed if necessary.
- *Low* level bugs are minor bugs which are mainly recommendations that would improve the game experience.

2.4 Game Production Tools

The most known tools within game production are the content creation tools, such as level editing tools, animation tools, music conducting tools, etc. However, the development team consists of people and therefore it is not solely a technical effort. It is also a social effort. In order to meet the requirements for a game title, the development team is required to employ into a collaborative and coordinated activity. That means that the members of the development team should interact with each other efficiently and effectively. At that point the game producer gets involved, to assure that these activities are operated by the team. That is the main reason that a game producer's tasks are demanding [8, 22, 37].

In order to fulfill successfully these activities, the game producer uses specific tools:

- Version Control. In software engineering, version control is the process of managing and tracking changes to source code. Thus, version control is implemented in game development as well [41]. There are a lot of software solutions in version control, such as *CVS*[1] and *SVN*[2]. These tools allow the development team work on the same project simultaneously and successfully upload new functionalities. The game producer uses this tool to manage changes in the game and documents.
- Bug Tracking. In order to keep track of the bugs that the testing team faces, the development team and the game producer, use bug tracking tools. By these tools the testing team can construct the bug reports and deliver to the game producer. An example of a bug tracking solution is *Jira*[3].
- Project Management. There are many challenges that the game producer encounters during game development. First of all, keeping track of the game development progress is a key activity that leads to a successful game title. For example, if a development team members is late in delivery the game producer, needs to identify this situation and make decisions to handle this situation. Also, the previous example is related to another game producer's activity, scheduling. In order for the game producer to identify this problematic situation, a software solution is needed. A software that can provide this kind of information is a project management tool [23]. A very popular software for that purpose is *Microsoft Project*[4]. The composition of a successful development team is another challenging task for the game producer. The way to form a development team depends on the scope of the game and the available budget that can be distributed to the team. For example, a 3D high quality game requires much more artists than a 2D arcade game. Also, if a game has a minimum budget set, it affords less developers than a high budget project. Therefore,

[1] http://savannah.nongnu.org/projects/cvs
[2] http://subversion.apache.org/
[3]https://www.atlassian.com/software/jira
[4] https://products.office.com/nl-NL/project?legRedir=true&CorrelationId=c55e43f3-355a-42d5-82da-27c201489c9d

budgeting is also a task that a game producer needs to fulfill. There is a variety of project management software that the project budget can be reported and monitored, in combination to tracking the progress of the development. A prime example of this software solution is *Clarizen*[5].

2.5 Business and Game Analytics

Companies willing to evaluate the way they function, use business analytics [42]. This method assists companies collect data on how a company perform in certain activities, either internal or external. The data that are collected are not structured and thus they need to be transformed to information that can be valuable for taking business decisions. The process of this transformation is called business intelligence [43]. The processed data can be represented in different forms, such as dashboards, analytical reports and table reports. The data analysis can be dealing with the working process and the personnel of the company or customers. The process of business analysis is presented in Figure 9.

Similarly, game development companies use game analytics. These analytics, however, focus on the end user of the game that is developed [44] and are usually referred as *User Analytics* [45]. Game analytics have been introduced in the game development field recently and have been implemented in many game studios [44, 45]. The purpose of the game analytics is to improve the process of game development in different levels. The *performance data* are accompanying the technical related aspects of game development, such as the FPS (frame rates per second) or the server behavior. The *process data* is mainly related with the process of developing games, mainly data obtained from a game producer and are important in monitoring the progress of the development. Finally, the *user data* correspond to the data obtained from the user/player, such as active users per day, or total playtime per user. In order to statistically analyze the data, some measures need to be obtained. These measures are called *Game Metrics* [44, 45].

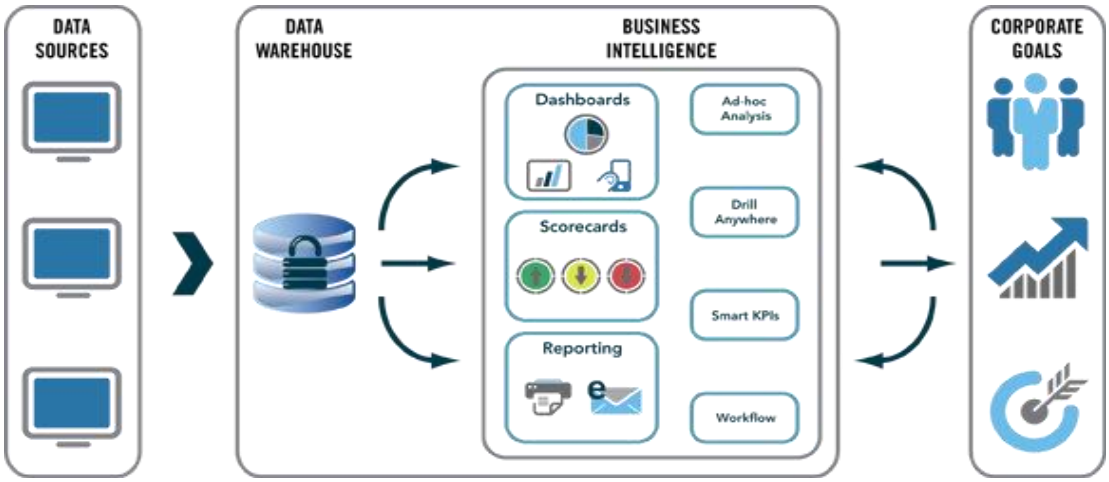

Figure 9: An overview diagram of business analytics.
Source*: https://motivitysolutions.com/business-intelligence/*

2.6 Conclusions

The literature study provided useful information regarding the game development process and the role of the game producer in it:

[5] http://www.clarizen.com/

- The main artifacts of game development were presented and identified and a comparison between the waterfall and agile method was presented. Through this comparison we identified that game studios that have a small and medium size, lean towards agile game development.
- The differences between a project manager and a game producer were determined and a mind-map of the overall game producer activities was constructed. The key differences of a game producer and a project manager is that the game producer copes with people from different disciplines and therefore different way of thinking and that the role of a game producer is adjustable and dynamic in a game studio. By this exploration, we managed to identify the key responsibilities of a game producer.
- The association between game testing techniques and game features was created and the most trending game testing techniques were explained.
- The presentation of the state-of-the-art game production tools used by game producers was held that will help us determine what the proposed software solution needs to integrate.
- Business and game analytics were introduced. In this part we draw an important conclusion. In game development, experts focus in user metrics, which gives us the opportunity to explore the process data from the scope of the game producer.

3 Design of the Solution

This chapter aims to introduce a conceptual model of the agile game production and introduce metrics aiming to assist game producers in monitoring the progress of game development and the effectiveness of the development team. Also, the suitability of the testing techniques for different game features is presented.

3.1 The Conceptual Model

In order to design the software solution, certain tasks needed to be performed. The first task was to identify which activities, artifacts and people are involved during the agile game development process. These are necessary for the game producer to obtain valuable information on whether the game development is heading towards the direction set during the project planning. These are transformed in classes and relations, which construct a conceptual model. An overview of the model can be observed in Figure 10, as a class diagram and the process can be observed in Figure 11.

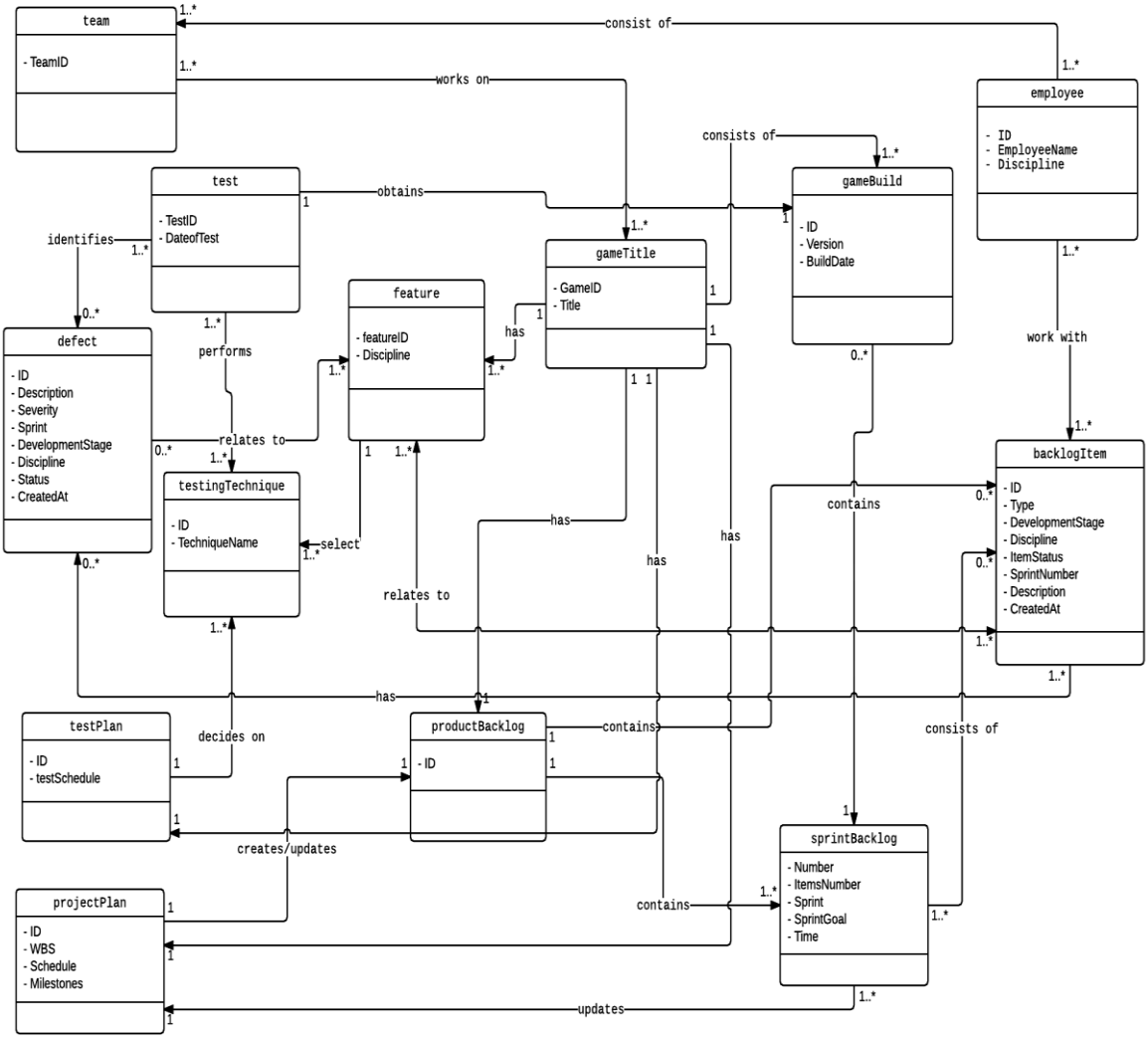

Figure 10: The class diagram of the conceptual model.

In a game development company, there exist one or multiple teams working on different game projects. Each team consists of members that belong to different disciplines. For example there are teams that consist of artists, designers, programmers and testers and teams that only include programmers and testers, on a very late stage of development. Every game title possesses a product backlog, in which belong all the known stories to complete the game development, according to the project plan.

During the development, new stories may arise that the development team needs to fulfill, so the product backlog is a non-static set of stories [31]. Instead, it is updated in every change that might occur but is in compliance with the company's stakeholders and within the time and financial borders [22, 31]. For instance, during the Alpha stage, the development team observe that it is impossible budget-wise and time-wise to develop a new level. So, the game producer with the development team decide to drop the creation of a new level and the project plan is updated according to this decision.

The agile process, consists of multiple sprints, so there exist multiple sprint backlogs. Before the start of the sprint the development team with the game producer create the sprint backlog, which is a set of multiple backlog items and a sprint goal. For example, a development team creates a sprint goal "Add Level 3 in game with basic functionality".

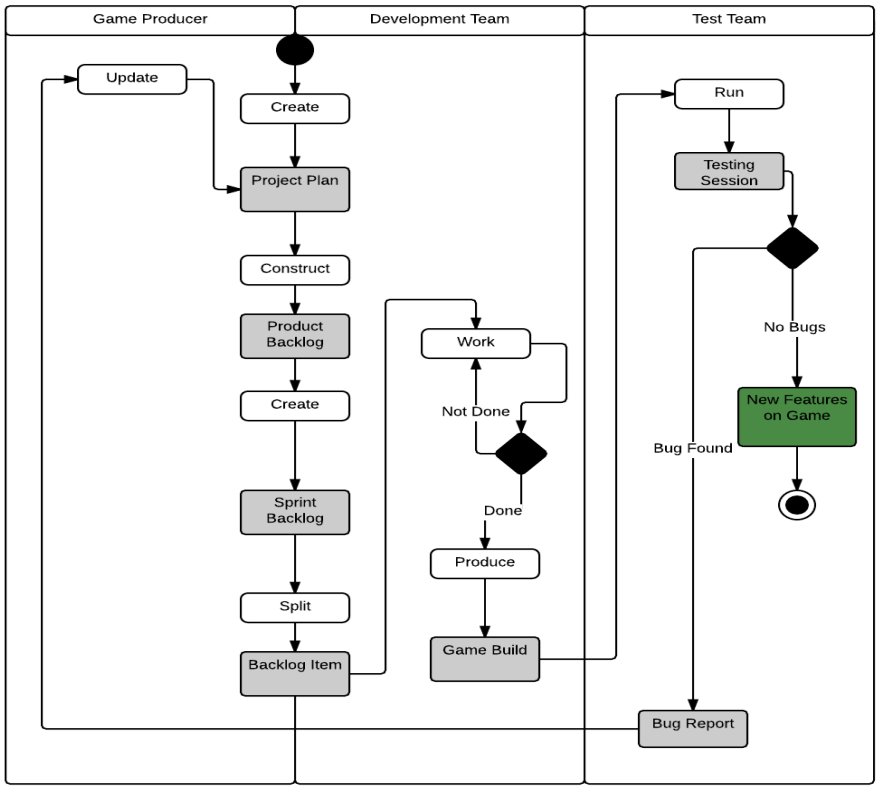

Figure 11: The activity diagram of the model.

The backlog items consist of product backlog items and also new stories that have been discussed and have been acknowledged from the team, during the sprint planning [31]. An example of a new story can be a new found bug that wasn't identified during the testing session. Each backlog item is related to a game feature and is assigned to a development team member or multiple members from the same or different disciplines. As soon as the sprint is close to the end and the sprint backlog

items are finished, the development team members construct the game build, which is a playable version of the game consisting of all the delivered stories from the sprint backlog.

Then, the test team obtains the game build to perform the quality assurance of the current sprint. According to the test plan, the testers implement a certain testing technique for the feature that is tested. During the testing session the test team identifies defects on features. These defects are reported after the testing session is over.

When the testing session has ended, the defects are prioritized, as mentioned in the literature study, by the game producer and are handed into the corresponding team. During the next sprint planning, the development team creates the new sprint backlog, taking into account, also, the results of the testing session and the list of defects the game producer has delivered.

3.2 Metrics of the Conceptual Model

The next task to be performed, in order to design the game producer's console, is the identification of the metrics. These metrics are extracted from the literature study and also the conceptual model. As mentioned in the previous chapter, the game metrics that currently exist are dealing mainly with the user experience [44] and how the game can be improved from the user's perspective. The metrics proposed and introduced in this chapter aim to assist the game producers evaluate the process of game development for multiple projects. So, these metrics are closely related to the process data analytics, presented in the literature study. Most metrics deal with defects, because the testing process is a very important activity in game development [38, 39], thus identifying and fixing bugs are crucial steps in the process of developing a successful game title.

The metrics need to be formally determined to avoid ambiguity. In order to achieve this the metrics definition is accompanied by creating SQL queries from the class diagram in Figure 10 and also providing results, by using the following scenario:

We assume that there is a company that has in development 3 different projects, a project close to the end with many bugs (A), a game that has started 2 sprints ago (B) and a project that is in production and is bug free (C). The company has 3 development phases, Pre-Production, Needs phase and Wants phase. In Needs phase the team develops the aspects of the game that have been set during the requirements determination, whereas the Wants phase are features that add additional value to the game, but are not required.

The Appendix presents the Microsoft Access diagram that was constructed, in order to execute the queries.

In the following list the proposed metrics, are presented, with fictional results based on the assumption. The reason to provide the results is to illustrate the differences that exist in projects that are on different stages and status.

General Metrics

The metrics that belong on this category, are metrics that give a general idea of the game development progress. These metrics present a high-level activity of the games that are developed. The metrics that belong on the general category are presented below:

- Defects per Game. This metric presents the bugs each game currently has that have not been resolved yet. The game producer is able to determine the overall unresolved defects in the ongoing projects. The query is:

```
SELECT
      COUNT(defect.ID), feature.gameTitleID
```

```
FROM
      defect INNER JOIN feature ON defect.FeatureID =
      feature.ID
WHERE
      defect.Status<>"Fixed"
GROUP BY
      feature.gameTitleID;
```

The query returns 17 bugs for project A, because it is a problematic project, as mentioned on the hypothesis. By this result, the game producer is able to identify that there is need in attention for project A, in order to identify what causes these problems.
Projects B and C have 0 zero bugs currently, because the first project is new and the latter project is currently bugless.

- Defects per Game (based on Development Stage). This metric is an overview of all the defects that a development team encountered during the process of developing a game, acting as a reference for the development team and the game producer to draw conclusions of the process.

```
SELECT
      COUNT(defect.ID), feature.gameTitleID,
      defect.DevelopmentStage
FROM
      defect INNER JOIN feature ON
      defect.FeatureID=feature.ID
GROUP BY
      feature.gameTitleID, defect.DevelopmentStage;
```

The query returns for Project A 8 pre-production bugs, 15 needs bugs and 33 wants bugs and it is clear that the game producer needs to take actions on that problematic project.
For Project B 4 pre-production bugs and 0 bugs for needs and wants, because the project is still in pre-production stage. The game producer could be able to identify where these bugs came from and take precautions for the future.
For Project C there are 10 pre-production bugs, 3 needs bugs and 0 wants bugs. By this results, the game producer could praise the work of the development team, since the current phase is the needs phase and the bugs on that stage, are limited compared to previous projects.

- Defects per Severity. Using this metric the game producer is able to observe the defects that the development teams are dealing with, based on the severity. The severity of each bug is given from the associate tester, so the game producer is able to prioritize them and observe in which project there are problems that need to be tackled.

```
SELECT
      COUNT (defect.ID), feature.gameTitleID,
      defect.Severity
FROM
      defect INNER JOIN feature
      ON defect.[FeatureID]=feature.[ID]
GROUP_BY
      feature.gameTitle, defect.Severity;
```

In out scenario this query would return 2 low bugs, 20 medium bugs, 25 high bugs and 9 critical bugs for Project A, meaning that the game producer could conclude the fact that this project's bugs are very severe and take actions, regarding the nature of these bugs.

4 low bugs for Project B that the game producer is able to conclude that having low bugs in pre-production is not something to worry about.
2 low bugs, 8 medium bugs, 3 high bugs and 0 critical bugs for Project C, helping the game producer come to a conclusion that the project has bugs that do not fall off the average bug rate for the company.

- Trend of Open Bugs. This metric aims to present the number of unresolved bugs in the game development process over time. Observing this metric the game producer can judge the progress of the development and the team's productivity.

```
SELECT
      Count(defect.ID), feature.gameTitleID, defect.Sprint
FROM
      defect INNER JOIN feature ON defect.FeatureID =
      feature.ID
GROUP BY
      feature.gameTitleID, defect.Sprint
ORDER BY
      feature.gameTitleID , defect.Sprint;
```

This query returns values for every sprint and the two adjacent values are calculated the trend. In Project A there is an increasing trend, something that can aid the game observe the time period that the bugs started to raise. In Project B there is a small increasing trend, because the project just started. This trend, however, does not allow the game make conclusions, because the game is still in a very early stage. In Project C the trend is gradually decreasing and is currently in 0, because of the "ideal" development so far, so the game producer can conclude that the development is on the correct track.

- Goal Completion per Development Stage. Every backlog item is characterized, amongst others, by the development stage it belongs. This metric aims to assist the game producer observe the percentage of the done stories regarding the development stage. By this, there can be a better evaluation on the progress of development.

```
SELECT
      (SELECT Count(backlogItem.ID)
      FROM backlogItem
      WHERE backlogItem.ItemStatus="Accepted")
      /
      (SELECT Count(backlogItem.ID)
      FROM backlogItem)*100, backlogItem.DevelopmentStage,
      feature.gameTItleID
FROM
      feature INNER JOIN backlogItem ON feature.ID =
      backlogItem.featureID
GROUP BY
      feature.gameTitleID, backlogItem.DevelopmentStage
ORDER BY
      feature.gameTItleID;
```

This query returns for Project A 85% in pre-production, 40% in needs phase and 25% in wants phase, helping the game producer observe that the game is not on the stage that it should be and some actions need to be made.

For Project B the values returned are 50% for pre-production and 0% for needs and wants phase, which is accepted from the game producer, since the development of this project started recently.
Project C has completed 100% of pre-production, 85% of needs and 60% of wants stories, results that the game producer can conclude that the game development is praiseworthy.

- Goal Completion per Time Period. This is a similar metric with the abovementioned, with the exception that it corresponds to the amount of successful sprints in a specific time frame, such as month or trimester. The following query returns the goals completed in the period between 2/2015 and 3/2015.

```
SELECT
      Count(gameBuild.ID), gameBuild.gameTitleID
FROM
      gameBuild
WHERE
      gameBuild.[BuildDate]>=#2/1/2015# AND
      gameBuild.[BuildDate]<#3/1/2015#
GROUP BY
      gameBuild.gameTitleID;
```

The response of this query provides values for every month or trimester as trends. Project A had an increasing trend (1 or 2 goals completed per month), but currently the project has 0 goals completed on the last 3 months. This trend can help the game producer identify that there is a problematic situation that started some months ago in this project and actions need to be taken.
Project B has 2 successful sprints so far, concluding that so far the development is accepted. Finally, Project C has an almost straight trend of goals completed per time (1 or 2 goals completed per month), aiding the game producer understand that the progress of the development is ideal.

- Trend of Newly added stories per Game. As mentioned in the conceptual model, there can be stories that were not inserted in the product backlog, but as the development progresses they are essential. This metric calculates, during the development of every game, the number of these stories. By this, the game producer might be able to observe the uncertainty of the game project throughout time.

```
SELECT
      (SELECT
      Count(backlogItem.ID)
      FROM
      (backlogItem INNER JOIN sprintBacklog ON
      backlogItem.ID = sprintBacklog.backlogItemID) INNER
      JOIN productBacklog ON backlogItem.ID =
      productBacklog.backlogItemID
      WHERE
      (backlogItem.CreatedAt)<=[sprintBacklog].[StartedAt])
      /
      (SELECT Count(backlogItem.ID) AS CountOfID
      FROM (backlogItem INNER JOIN sprintBacklog ON
      backlogItem.ID = sprintBacklog.backlogItemID) INNER
      JOIN productBacklog ON backlogItem.ID =
```

```
      productBacklog.backlogItemID)*100,
      productBacklog.gameTitleID
FROM
      (backlogItem INNER JOIN sprintBacklog ON
      backlogItem.ID = sprintBacklog.backlogItemID) INNER
      JOIN productBacklog ON backlogItem.ID =
      productBacklog.backlogItemID
GROUP BY
      productBacklog.gameTitleID, backlogItem.SprintNumber;
```

In Project A there are a lot of issues, so the trend is increasing and decreasing. This trend helps the game producer understand that the product backlog is changing often, so that could be one of the reasons of the problematic situation.
Project B is in pre-production phase so the percentage is still to 0, because so far the project plan is being followed. Due to the success of Project C there are limited stories that do not belong to the product backlog that the game producer can conclude the fact that, in order to have a successful game development, the project plan has to be followed with limited additions.

- Percentage of Product Backlog Items moved to Sprint Backlog (based on Development Stage). This metric can aid the game producer observe how well-defined is the product backlog, depending the development stage of the game.

```
SELECT
      (SELECT Count(backlogItem.ID) AS CountOfID
      FROM (backlogItem INNER JOIN sprintBacklog ON
      backlogItem.ID = sprintBacklog.backlogItemID) INNER
      JOIN productBacklog ON backlogItem.ID =
      productBacklog.backlogItemID
      WHERE
      (backlogItem.CreatedAt)<=[sprintBacklog].[StartedAt])
      /
      (SELECT Count(backlogItem.ID) AS CountOfID
      FROM (backlogItem INNER JOIN sprintBacklog ON
      backlogItem.ID = sprintBacklog.backlogItemID) INNER
      JOIN productBacklog ON backlogItem.ID =
      productBacklog.backlogItemID)*100,
      backlogItem.DevelopmentStage,
      productBacklog.gameTitleID
FROM
      (backlogItem INNER JOIN sprintBacklog ON
      backlogItem.ID = sprintBacklog.backlogItemID) INNER
      JOIN productBacklog ON backlogItem.ID =
      productBacklog.backlogItemID
GROUP BY
      productBacklog.gameTitleID,
      backlogItem.DevelopmentStage;
```

The query returns high percentages for Project C in all development stages, concluding that the stories that the project plan has determined are being followed. On the other hand, Project A has much lower values, especially in needs phase. This result concludes that the product backlog is changing often in needs phase, so the game producer can identify that the needs phase is the most problematic. Project B is still in pre-production so the values

on needs and wants phase are 0. Whereas the percentage on pre-production is 100%, meaning that so far the project plan is completely followed.

.

- Percentage of Product Backlog Items moved to Sprint Backlog (based on Story Type). This metric is visualizing the percentage of the stories that exist in the product backlog, based on their type. The story type can be trifold; feature, bug and chore. The definition of a feature and a bug are straightforward. Story type chore is a task that one or multiple team members need to fulfill that normally is not directly related to game features. An example of a chore is a meeting with stakeholders.

```
SELECT
      (SELECT Count(backlogItem.ID) AS CountOfID
      FROM (backlogItem INNER JOIN sprintBacklog ON
      backlogItem.ID = sprintBacklog.backlogItemID) INNER
      JOIN productBacklog ON backlogItem.ID =
      productBacklog.backlogItemID
      WHERE
      (backlogItem.CreatedAt)<=[sprintBacklog].[StartedAt])
      /
      (SELECT Count(backlogItem.ID) AS CountOfID
      FROM (backlogItem INNER JOIN sprintBacklog ON
      backlogItem.ID = sprintBacklog.backlogItemID) INNER
      JOIN productBacklog ON backlogItem.ID =
      productBacklog.backlogItemID)*100, backlogItem.Type,
      productBacklog.gameTitleID
FROM
      (backlogItem INNER JOIN sprintBacklog ON
      backlogItem.ID = sprintBacklog.backlogItemID) INNER
      JOIN productBacklog ON backlogItem.ID =
      productBacklog.backlogItemID
GROUP BY
      productBacklog.gameTitleID, backlogItem.Type;
```

This query returns a very high percentage for features in all projects, meaning that the stories the product backlog includes are mainly features.

Feature-related Metrics

The metrics that belong on this category have in common the fact that deal with game features. That means that the results that are visualized depend on the game features each game has. The metrics are:

- Defects per Feature. This metric is based on the bugs of every feature the development team is dealing with. Using this metric the game producer can assess the most problematic features and to investigate whether there should be some changes in developing the features.

```
SELECT
      Count(defect.ID),feature.gameTitleID,
      feature.FeatureName
FROM
      feature INNER JOIN defect ON feature.ID =
      defect.FeatureID
GROUP BY
```

```
      feature.gameTitleID, feature.FeatureName;
```

This query returns for Project B bugs in features that are related to prototyping and documents. Project A has many bugs in different features and mainly in Art-related features and especially the character animations. So, the game producer is able to identify that there is a problem situation in the team members that deal with the animations. Project C has a balance in bugs for all the features, with some more programming-related features. So, the game producer can conclude that the more demanding features belong to the programming game features.

- Defects per Feature per Sprint. It is a relevant to the previous metric, which visualizes the defects that exist for every feature for every sprint, giving the opportunity to the game producer together with the development team tackle the most defective features.

```
SELECT
      Count(defect.ID), feature.gameTitleID,
      feature.FeatureName, defect.Sprint
FROM
      feature INNER JOIN defect ON feature.ID =
      defect.FeatureID
GROUP BY
      feature.gameTitleID, feature.FeatureName,
      defect.Sprint
ORDER BY
      defect.Sprint, feature.gameTitleID;
```

The results for Project A provide information that the animations have defects in many consecutive sprints, so the game producer is able to take actions regarding the members responsible for the specific game feature.
In Project B there are bugs in design and documentation. The project is in very early stage, so the game producer does not need to take actions.
Project C has bugs related to AI that the game producer can conclude that the most demanding game feature is the AI and try to take actions regarding this feature.

- Defects per Feature (based on Development Stage). This would aid the game producer identify in which stages of game development each feature was more problematic.

```
SELECT
      Count(defect.ID), feature.gameTitleID,
      feature.FeatureName, defect.DevelopmentStage
FROM
      feature INNER JOIN defect ON feature.ID =
      defect.FeatureID
GROUP BY
      feature.gameTitleID, feature.FeatureName,
      defect.DevelopmentStage
ORDER BY
      feature.gameTitleID, defect.DevelopmentStage;
```

In all development stages of Project A, the most problematic features are the art features. Another way for the game producer to identify that the art features are producing many bugs.
In Project B the game development has just started and the design bugs do not need any attention.

In Project C the bugs related to programming features are higher and especially regarding the AI. These bugs have been resolved already, but the game producer can conclude that the AI feature on the next games to be developed might need extra attention.

- Trend of Defects per Feature (based on Time Period). By this metric the game producer is able to identify the progress a feature has made in time and whether the development team needs to increase the productivity. Also, this metric can be helpful in identifying if the deadlines and milestones that have been set, during the project planning, can be met. The following query returns the defects per feature in the 2nd trimester of 2015.

```
SELECT
      Count(defect.ID), feature.FeatureName,
      feature.gameTitleID
FROM
      feature INNER JOIN defect ON feature.ID =
      defect.FeatureID
WHERE
      defect.[CreatedAt]>=#4/1/2015# AND
      defect.[CreatedAt]<#7/1/2015#
GROUP BY
      feature.FeatureName, feature.gameTitleID;
```

Amongst the values that this metric returns in Project A, the character model feature seems still with bugs, whereas it should have been already completed. By this, the game producer can conclude that there need to be an addition in the art discipline, in order to complete the feature and meet the goals.
Project B is very early in development so there cannot be any conclusions.
Project C seemed to have bugs regarding the menu art for some time, but the feature was complete in schedule and eventually no new bugs were aroused. On this project, the game producer can praise the art team for resolving the bugs fast and effective.

Discipline-related Metrics

On this category the proposed metrics are group per discipline. So, they handle the results according to the different disciplines that exist on the game development teams. The proposed metrics are:

- Defects per Discipline. Another important metric for the game producer, due to the fact that this metric helps understand which teams encounters the most challenges, within the game development process. Therefore, the game producer can claim additional members to assist this problematic situation. Also, it might assist the teams understand that some projects, based on the requirements, need more effort in specific disciplines.

```
SELECT
      Count(defect.ID, feature.gameTitleID,
      defect.Discipline
FROM
      feature INNER JOIN defect ON feature.ID =
      defect.FeatureID
GROUP BY
      feature.gameTitleID, defect.Discipline;
```

This query returns an increasing value for programming bugs in Project C that the game producer can conclude the fact that the programming team has more bugs.

The bugs that belong to the art discipline are much more compared to the other disciplines on Project A that the game producer can conclude that the art team is more problematic. Only design bugs are returned for Project B. The game producer could conclude that the design team faces issues in the current state.

- Defects per Discipline per Development Stage. This is a metric that could assist the game producer as the previous metric and act more as a reference for future game planning.

```
SELECT
      Count(defect.ID), feature.gameTitleID,
      defect.Discipline, defect.DevelopmentStage
FROM
      feature INNER JOIN defect ON feature.ID =
      defect.FeatureID
GROUP BY
      feature.gameTitleID, defect.Discipline,
      defect.DevelopmentStage;
```

The query returns the design bugs from Project B in Pre-production. The game producer cannot make conclusions for this metric, because the project is in a very early stage.
Project C has few pre-production and needs phase bugs that are spread in all the disciplines and minimum programming bugs in wants phase. The game producer is able to make conclusions that in all the disciplines issues raised, but none of them converges on a specific discipline.
Project A has multiple pre-production bugs and much more art bugs in needs phase. So, the game producer can observe that in the needs phase there is a problematic situation in the art department.

Testing Technique-related Metrics

This category presents metrics that are related to the testing techniques. These metrics are:

- Defects per Testing Technique. This metric is focusing on the defects that each testing technique a game developing company is facing. By this metric, the game producer might be able to evaluate the necessity and efficiency of each testing technique and also propose changes in the testing process.

```
SELECT
      Count(defect.ID), feature.TestingTechniqueID,
      feature.gameTitleID
FROM
      defect INNER JOIN feature ON defect.featureID =
      feature.ID
GROUP BY
      feature.TestingTechniqueID, feature.gameTitleID
ORDER BY
      feature.gameTitleID;
```

The resulting values of this query is almost the same in projects A and C. The testing technique with the most bugs is playtesting. The game producer can conclude that the most problematic testing technique is playtesting, however, in combination with the next metric can be identified, whether there is a problem on the testing technique or it is the most popular one.
However, project B has free testing bugs, due to the fact that the bugs are design-related.

- Use of Testing Techniques per Development Stage. This metric is a statistical representation of the testing techniques for every development stage. The metric is able to assist the game producer observe which is the most popular testing technique for each development stage and also assist is defining the test plan in future projects.

```
SELECT
      Count(backlogItem.ID), feature.gameTitleID,
      backlogItem.DevelopmentStage,
      testingTechnique.TechniqueName
FROM
      testingTechnique INNER JOIN (feature INNER JOIN
      backlogItem ON feature.ID = backlogItem.featureID) ON
      testingTechnique.ID = feature.testingTechniqueID
GROUP BY
      feature.gameTitleID, backlogItem.DevelopmentStage,
      testingTechnique.TechniqueName
ORDER BY
      feature.gameTitleID;
```

  As an outcome from the previous metric, the most used testing technique, in every stage of projects A and C, is playtesting. However, there is an increased value in direct testing for project A, because the art assets to be tested are more specific. Project B has zero values for wants and needs phase for all the testing techniques.

- Use of Testing Technique per Test Plan. This metric aims to identify the testing techniques that the test plans are scheduling. By this metric, the game producer and the test lead can observe which testing techniques is scheduled from the test plan. This metric with the combination of the *Use of Testing Techniques per Development Stage*, can provide valuable information regarding what is scheduled and what is actually performed.

  On this metric, there are many direct testing results that contradict to the actual use of the testing techniques from the previous metric. So, the game producer can conclude that the test plan is not followed as it should. For the other 2 projects the metric returns that the most used testing technique in respect to the test plan is playtesting.

3.3 Conclusions

On this chapter the conceptual model of the agile game development, from the game producer's point of view, was presented. Also, the right metrics regarding the process of an agile game development were introduced and formalized. Those two steps are the fundamentals of developing a console that would assist game producers manage the agile game development. The following chapter, presents the way that this console was developed.

4 Prototype

The goal of this chapter is to present the console that has been developed, according to the conceptual model and the metrics defined on the previous chapter. Also, the technical elements on how the console was developed are explained.

4.1 PivotalTracker

As described in the previous chapters, the role of the game producer is very demanding. That is the reason that game producers rely on software solutions that can assist them on their tasks. PivotalTracker[6] is an example of that software. It is an agile project management tool that allows teams track and log their daily progress, but also promote collaboration and the game producers to observe the progress of the project. An overview of a project in PivotalTracker can be viewed in Figure 12.

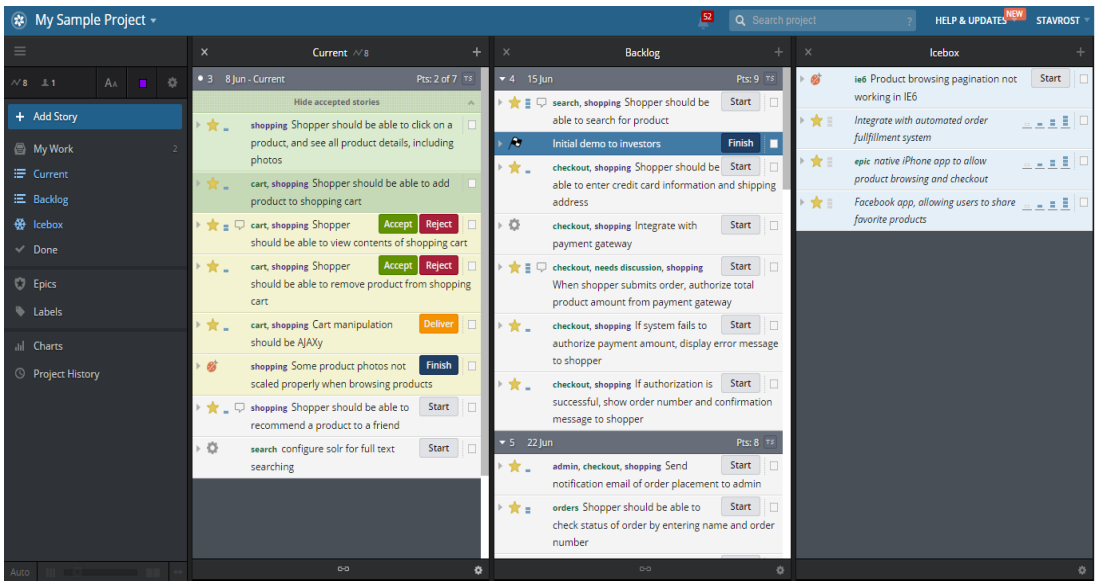

Figure 12: An example of a Project overview in PivotalTracker.

In order to use the tool, all the development team members need to create an account and be invited by the administrator of the project.

As it can be observed from Figure 12, the project consists mainly of 3 panels. The ***current*** panel is concerning the sprint backlog, the ***backlog*** panel is concerning the product backlog and the ***icebox*** panel is representing the stories that respond to stories that have yet to be determined and might not be executed. For the purpose of developing the console, the role of these panels is slightly modified.

The ***current*** and ***backlog*** panels correspond to the sprint backlog and the ***icebox*** panel includes the stories that correspond to the product backlog. This modifications allowed us to gain an advantage. The number of stories included in the sprint backlog is automatically generated from the project's *velocity*. The velocity is calculated from the points that the development team obtained from previous sprints. These points can be days or difficulty levels, but we use hour estimates. Due to this automatic planning, the ***backlog*** panel hosts some stories from the current sprint. This is also

[6] http://www.pivotaltracker.com/

the reason that the ***icebox*** panel hosts product backlog items. Apart from this reason, our case company uses the system in such a manner.

Each panel consists of multiple stories that have a description. The description corresponds either on the activity that the user is able to perform after the completion of the story or the tasks that the development team member has to perform. Each story has a certain story type that are presented below and are valuable in calculating the metrics of the console:

- Feature. It is the story type that, after its completion, a game feature is created or new functionalities of a game feature are added. An example of a feature story type is *"Create level start and end sequence functionality"*.
- Bug. This story type corresponds to defects that the testing or development team has encountered during the testing session. An example of a bug story type is *"Fix character jumping animation"*.
- Chore. It is a story type that the development team uses, in order to complete a task that does not add direct value to the game. An example is *"Buy plug-in from Unity Store"*.
- Release. This story type mainly corresponds to milestones that the game development team has set during the project planning. Also, releases can be set as sprint goals for every sprint.

When stories are created they can be accompanied with multiple labels. These labels are created from the development team and group the user stories. For the purpose of developing the console, some label groups were introduced and added to the stories. The categories and the values are:

- Development Team. This category determines which development discipline works on the story, such as *Art*, *Design*, *Programming* and *Audio*.
- Feature. On this category, the stories are group depending on which game feature the story adds value to, after the story is completed. Examples of game features added on PivotalTracker as labels are *menu, animations, character, server, models*.
- Development Stage. With this label category, the stories are grouped depending on the development stage that they belong. The labels that are used for this category are *pre-production*, *needs* and *wants*.
- Testing Technique. The labels from the testing technique category correspond to the testing method that is followed, in order to assess the game feature on the story. The testing techniques are presented in Chapter 3.
- Severity. This label presents which severity level are the bug-type stories. The severity levels are discussed in Chapter 2.

Also, apart from the title of the story and the accompanying labels, the story consists of some other elements. For example, the points that the story has. Most of the times, the points are the estimated hours to finish the story.

Figure 13 presents an example of a story and the labels accompanied. On the left, the star corresponds to the type of the story, which is a feature. The number next to the star is the points that the story has and the large text is the title of the story. The green words on the bottom of the story are the labels that are followed by the story.

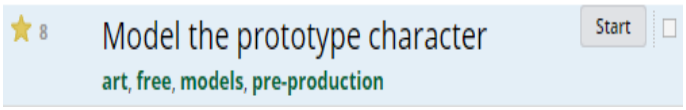

Figure 13: A story example.

PivotalTracker is able to store user data through resources that have specific attributes and provides an API that can be used to retrieve data from projects that are hosted by the system. In order to have access to the data that PivotalTracker stores, the user obtains a unique API Token. This value is important, because it is necessary for authentication. Through the API we are able to calculate the metrics introduced in the previous chapter. In order to fulfill this task there are certain activities to occur. The first activity is to identify and align the conceptual model elements with resources from the PivotalTracker data model.

Table 2 presents the alignment between the elements of the conceptual model and the resources.

Conceptual Model Element	**PivotalTracker Resource**
gameTitle	*Project_ID*
productBacklog	ALL *stories*
backlogItem	*story*
employee	*person*
defect	*story* WITH *story_type=bug*
testingTechnique	*label*
feature	*label*

Table 2. Conceptual model elements to PivotalTracker resources.

The next activity to be followed is to develop the console to retrieve data from PivotalTracker and presents visually the metrics, which were introduced in the previous chapter. However, the metric *Use of Testing Technique per Test Plan* is not implemented in the console, due to the fact that the case data that we obtained do not include the projects' test plans. Also, the test plan is a document that cannot be integrated, at least on the prototype phase of the console.

4.2 Technology

The prototype of the console, proposed to assist the game producers in managing the game development process, is a web-based application. For the purposes of the case study, the prototype is hosted on a local host. The reason that the console is a web-application relies on the fact that most of the project management software is hosted on the web.

The console is developed in *HTML 5* and *Javascript* language, by using the *Bootstrap framework*[7]. In order to obtain data, to calculate the metrics from PivotalTracker, there exist *AJAX* requests. To visualize the metrics, the *Morris Chart* [8]library has been used, a library that provides bar, donut, line and area charts.

Figure 14 presents the high level architecture of the console. By this architecture we are able to identify the process of how the metrics are obtained from PivotalTracker and then presented to the users. The user requests to view a specific metric. As soon as there is this inquiry, the console sends

[7] http://getbootstrap.com/
[8] http://morrisjs.github.io/morris.js/

a request to the database of PivotalTracker. The response of the request is translated in a chart and the user is able to view the result.

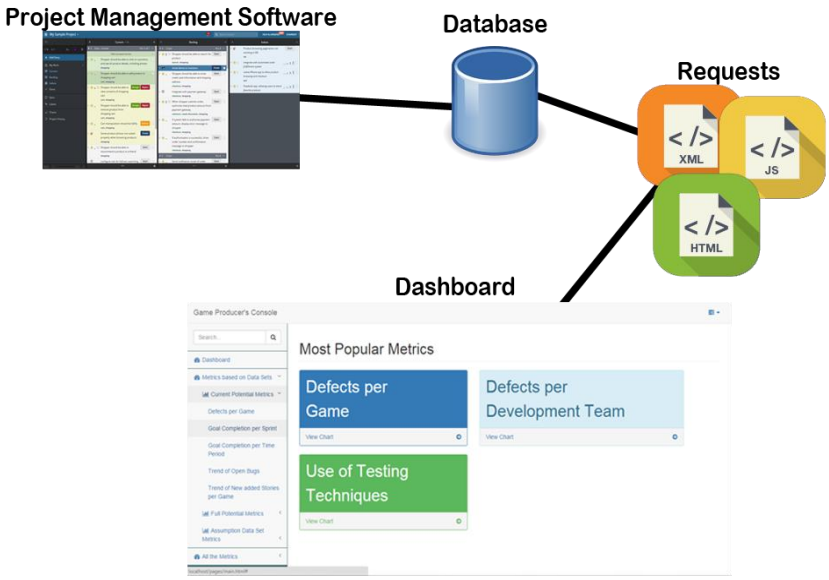

Figure 14. Architecture of the console.

Figure 15 presents the component diagram of the procedure, which provides further details on how the software is behaving.

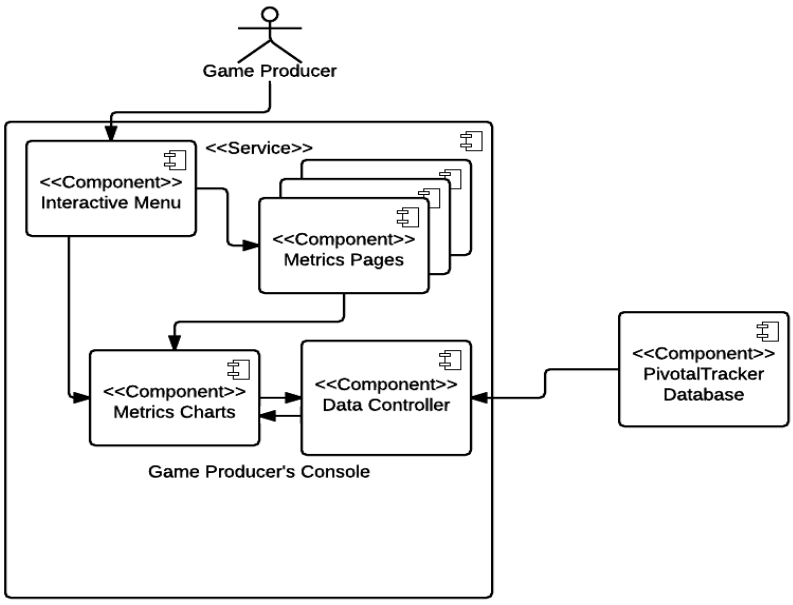

Figure 15. The component diagram of the console.

The requests that data are obtained from the PivotalTracker server, are based in cURL and the HTTP protocol. cURL uses the URL syntax, with an example of a metric request returning the bugs of project A that have high severity level for the console presented:

```
https://www.pivotaltracker.com/services/v5/projects/Project A/stories?filter=story_type:bug label:'high'
```

4.3 Dashboard

The tool consists of multiple pages that each one represents a metric. The metrics are directly available from each page, because of a menu that exists on the left side of the page that consists of 4 menu items. The first element is the "Dashboard", where the user can return to the main page of the tool. The main page, as shown on Figure 16, presents the most popular metrics that the user has navigated to. For the purpose of the prototype, the main page presents 3 pre-determined metrics, rather than the most popular ones.

The remaining 3 menu items correspond to categorization of the metrics. The first category is *"based on Data Sets"*, which is categorization according to which data sets are obtained. This kind of categorization is explained thoroughly on Chapter 5. The *"All the Metrics"* sub-menu presents in alphabetical order all the metrics and the *"based on Category"* sub-menu presents the metrics by dividing them in categories, such as testing technique, feature, development stage, etc. On the top of the dashboard there exists a dropdown menu, where the milestones of all the ongoing projects are presented. The purpose of this option is to allow game producers have immediate access to the milestones and be able to monitor the progress of the projects, in respect to the deadlines.

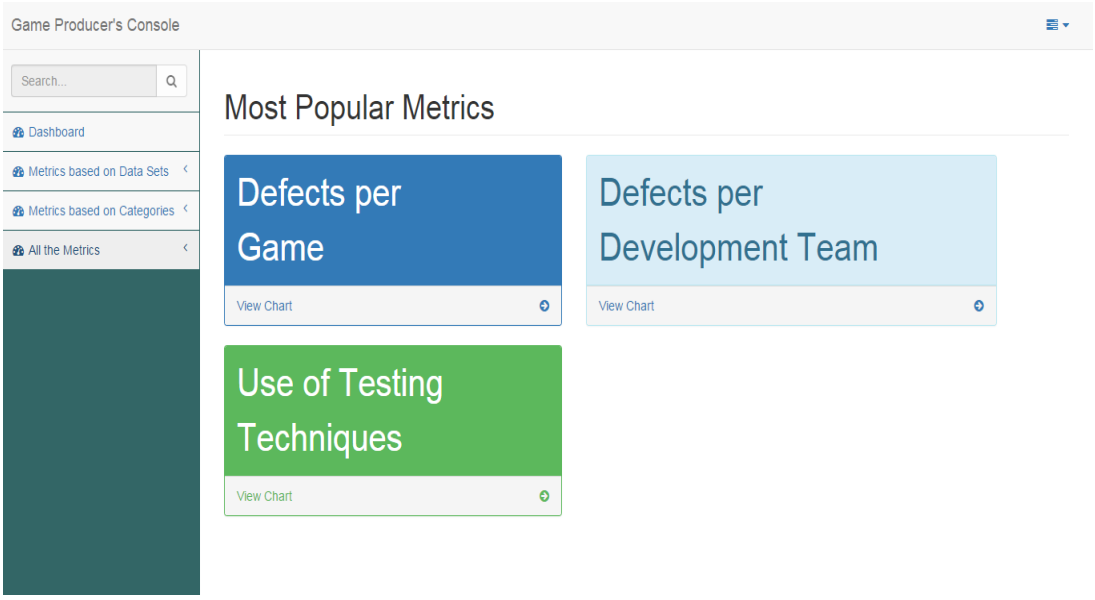

Figure 16. The main page of the dashboard. On the left side there exist the menu items.

On every pages there is a chart that the metric is visualized and there are options to edit the visualization. For example, in the metrics that the results depend on time, the chart can be visualized per month or per quarter. Another option regarding the visualization of the metrics is to filter the games and observe the metric for every game separately or altogether. Figure 17 presents an example of a metric with the options on the top.

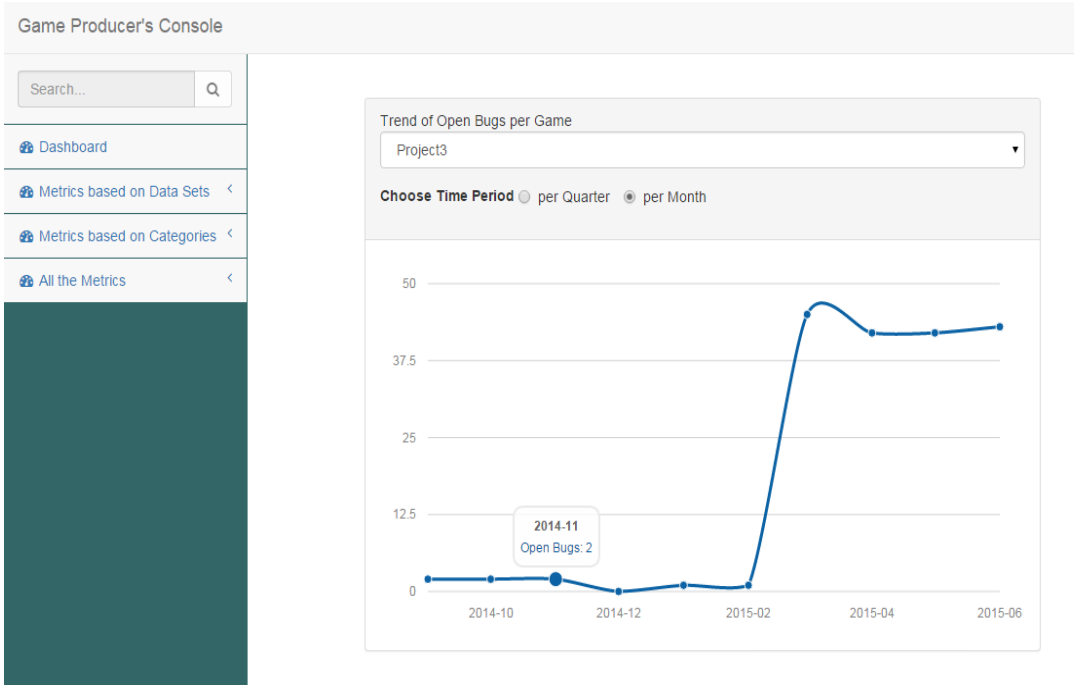

Figure 17. The *Trend of Open Bugs* metric.

An example of a different metric is presented in Figure 18, where a bar chart is depicted with different options.

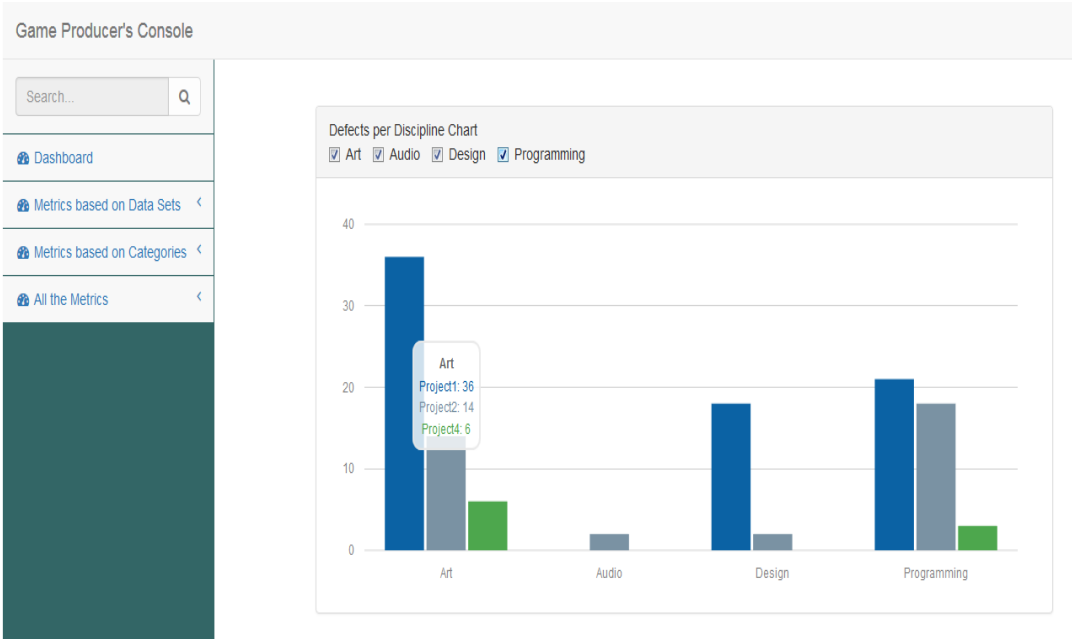

Figure 18. The *Defects per Discipline* metric.

A final example is presented in Figure 19, where a donut chart is visualizing a different metric.

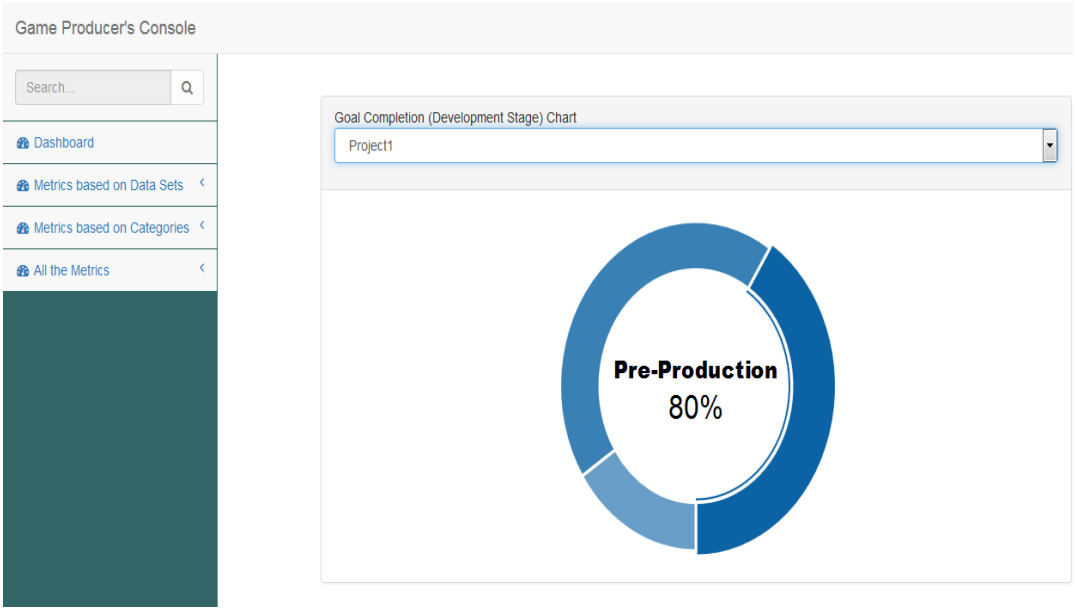

Figure 19. The *Goal Completion per Development Stage* metric.

5 Case Study

This goal of this chapter is to introduce the case study that was executed, in order to evaluate the console that was explained in the previous chapters.

5.1 Preparation

In order to conduct the case study, we needed to perform activities to prepare the evaluation. The first activity was to identify the goals of the case study, which are presented on the following list:

- ***Evaluate*** the current functionality of the console. By this goal we aim to identify, whether the console that we developed is sufficient to cover the needs of game producers regarding the task of managing the agile game development.
- ***Identify potential*** of the console, during the agile game development management. This goal aims to identify whether the console that is proposed is able to make the agile game development more efficient.
- ***Evaluate*** the proposed ***metrics*** that aim to assist the game producers in managing the game development process.

After the scope and the goals were defined, the next step was to formulate the hypothesis of our case study:

H_0: The proposed console and metrics do not help the game producers manage game development effectively.

H_1: The proposed console and metrics help the game producers manage game development effectively.

By defining the null hypothesis and the alternative hypothesis, we try to reject the fact that the console and the metrics that we developed do not assist game producers.

The term *"effectively"* that we employed in the hypotheses, refers to the capability of the game producer to have a better overview of the current projects. It also refers to its capability to anticipate or quickly react to negative events within those projects. We hypothesize that the console, especially through the metrics that it embeds, will be conductive to effective management.

The next activity was to decide on which kind of evaluation should be held. The most suited evaluation method seemed the semi-structured interviews. The reason that the semi-structured interview was chosen is because this way of evaluation is more suited in qualitative research [46]. Also, this method of interviewing is allowing the field experts to provide with new ideas on a field that is currently limited.

The protocol of the interview is presented below:

- The first phase of the protocol is a presentation of the console, in order to introduce to the subjects the console and the purpose of its development. The presentation lasts for 5-10 minutes.
- Next, the subject explores the console, by observing the metrics that exist and the functionality that the dashboard provides. By the meantime, the interviewer is keeping track of the activity of the subject. The exploration lasts 10-15 minutes, so the subject is able to identify the full potential of the console.

- The final phase of the protocol is the interview, in which the subject responds to the pre-determined questions that concern the console, the metrics and the game producer's benefits from it. The duration of the interview lasts 15-20 minutes.

An important aspect of the preparation of the case study was to determine the sample that will participate in the experiment. We decided to draw the sample by using non-probabilistic convenience sampling. This way of selecting samples helps us conduct the experiments with people who are directly available and we had access to [47, 48]. The main advantage on selecting this method is the fact that we managed to perform the experiment in short period, without delays. A main drawback of this method is that the subjects that were selected do not represent the field experts, because the number of participants is rather small, compared to the number of our potential users. Although the sample is not representative, in a sense that the people that were selected are limited, the experiments give us the opportunity to conduct a qualitative research.

For the cause of obtaining relevant to the research information, the interview questions were grouped according to the research questions, as follows:

1. **What are the key responsibilities of a Game Producer?**

    a. *Do you believe this console would be able to improve the process of managing the game development? Why?*

This question is related to the research question, because one of the key responsibilities of a game producer, according to the literature study, is tracking the progress of the game development and therefore managing the process.

2. **What are the problems that come from a non-coordinated game development?**

    a. *Do you believe that using this console, you would have avoided some issues during the game development?*

The above question is related to the research question, due to the fact that with the interview question we try to extract if there were issues that have appeared in the process of game development and observe whether, by using this console, some situations might have been obviated.

3. **What are the metrics to assess the effectiveness of coordinating the game development?**

    a. *Do you believe that the metrics proposed on the console are sufficient to fulfill the purpose of the console?*

With this question, we try to identify whether the metrics we propose are enough to help the game producer coordinate the game development effectively. That is the reason that the question is related to the research question.

    b. *What additional metrics would you like to observe from the console?*

This question aims to help us understand what other necessities game producers have, in order to manage the game development effectively. It belongs to the research question 2, because the answers of this question are a valuable addition of the already proposed metrics and therefore give new effective metrics.

    c. *Which of the metric(s) proposed from the console you believe is (are) more important? Why?*

This is the question that help us evaluate the metrics that we propose, because it gives the opportunity to the field experts determine the most important metrics. The outcome of this question is a set of metrics that are more effective and necessary for a game producer.

4. What is an effective software solution to support the game producer's coordination?

a. Are there any additional functionalities that you think that can improve the use of this console?

With this question we try to evaluate what additional functionalities are able to make the prototype of the console improve and become a valuable tool for the coordination of game development. It falls under research question 4, because field experts give new insights that can make the software more effective.

b. Would you use this console for managing the progress of the projects?

The answers of that question give us the opportunity to evaluate the console and identify the potential that the console has. It is related to the research question 4, because the results of this question help us conclude on what is an effective software solution for the game development management.

c. Would you use a custom console, based on your needs, in order to manage the game development process? Or rely on existing software solutions?

The goal of this question is to identify the preference of game producers in selecting a software solution to manage the process of game development. The question is part of research question 4, in the sense that the interview subjects declare what are the needs of an effective software that helps game producers.

The questions are presented to the participants in a random order, since the order of the questions do not affect the outcome of the interviews. We define a semi-structured interview, so the participant is allowed to answer a question other than the one asked, by combining more than one questions on the answer.

After the definition of the questions and the experiment design, the next step was to define threats to validity. The threats are presented below:

- The number of participants is low, so the conclusions are hard to be generalized. By this, we were able to draw conclusions regarding small and medium game studios within the Netherlands.
- Another threat to validity is the fact that the subjects that we selected by the convenience sampling method, are not only game producers. However, they are experts in other positions in game development that have also management activities.
- The experiment allows the participants to explore the console for 15' in data that they do not have personal experience. A threat that is extracted from that is the fact that the participants are not able to use the console for their own projects.

5.2 Data Sets

As mentioned on the previous chapter, the menu accommodates 3 categories of accessing the metrics. The first of them is *"Based on Data Sets"*. This type of categorization was held, in compliance with the data that we had access to. The company that the case study obtained data from uses PivotalTracker and has multiple ongoing projects.

The *"Current Potential Data Set"* sub-category, requests the metrics that can be immediately obtained from the projects un-edited. This means that the stories that each project consists of did not get any modification or change. The metrics that belong on this sub-category are:

- ***Defects per Game***
- ***Goal Completion per Time Period***
- ***Trend of Open Bugs***
- ***Trend of New Added Stories***
- ***Percentage of Product Backlog Items moved to Sprint Backlog based on Story Type*.**

The next sub-category is *"Full Potential Metrics"* that an enriched version of the previous data set are used to calculate the metrics. The additions of this data set rely on the fact that stories obtained labels for the discipline they occupy, the development stage they belong and the feature they add value to. If the stories are bugs, there is an additional label that concerns the severity of the bug. We were able to give the appropriate labels to the stories, due to the fact that we had access to the project plan of the company that the case study deals with. The following metrics belong to this sub-category:

- ***Defects per Feature***. On this metric the addition of the game feature label was necessary.
- ***Defects per Feature per Sprint***. On this metric the addition of the game feature label was necessary.
- ***Trend of Defects per Feature based on Development Stage***. The additions required to calculate this metric were the game feature and development stage label.
- ***Trend of Defect per Feature based on Time Period***. On this metric the addition of the game feature label was necessary.
- ***Defects per Severity***. In order to calculate this metric the bugs were label on different severity levels.
- ***Defects per Discipline***. This metric was calculated by adding the discipline on which the bugs refer to.
- ***Defects per Discipline based on Development Stage***. To calculate this metric the labels of the discipline the bug relates to and the development stage each bug concerns were added.
- ***Defects per Game based on Development Stage***. The addition of the development stage of each bug was necessary to determine this metric.
- ***Goal Completion per Development Stage***. The same addition with the previous metric applies to this one. The development stage that each story belongs to was added.
- ***Percentage of Product Backlog Items moved to Sprint Backlog based on Development Stage***. In order to calculate the metric, the addition of the development stage on every story was necessary.

The last sub-category is *"Assumption Data Set Metrics"*. The reason that metrics belong on this data set, relies on the fact that the company that we obtained the data to develop the console does not follow a certain plan on testing. Thus, we propose the testing techniques that perform better, based on the game feature each story concerns. This assumption is underpinned by the literature

study on game testing. In order to calculate the metrics we added the testing techniques on each story as a label. The metrics that reside in this data set are:

- ***Defects per Testing Technique***
- ***Use of Testing Techniques based on Development Stage***

5.3 Results

After the structure of the case study was designed and all the appropriate activities were fulfilled, we run the experiment. With the convenience sampling we managed to select 6 participants that belong to the game industry. These participants are game producers or have managerial duties within the game industry, such as art, design or technical directors. All of the participants belong to companies that develop games with the agile method.

In order to keep the anonymity, the participants are addressed as Participant X, where X is the unique number of every person. Table 3 provides demographic information regarding the participants of the experiment.

Participant	Sex	Age	Position	Company Size	Years in Game Industry
Participant 1	M	35-40	CEO, Technical Director, Game Producer	10-15	12
Participant 2	M	30-35	Art Director, Scrum Master	10-15	1
Participant 3	M	30-35	Design Director	10-15	7
Participant 4	M	25-30	Senior Designer, Scrum Master	5-10	3
Participant 5	M	25-30	Technical Director	5-10	4
Participant 6	M	25-30	Game Producer, Scrum Master	5-10	9

Table 3. The participants profile.

As mentioned on the protocol presentation, while the participants were exploring the console, we were tracking their activity. During this phase we managed to observe some interesting points:

- All of the participants seemed motivated in exploring the console and understanding the purpose of the proposed metrics.
- The participants within the company that the data was collected, spent a lot of the exploration time in the metrics that correspond to the un-edited data.
- Also, participants with different background focus on different metrics. Participants having game design background spent more time in features, whereas participants in project management positions were exploring more the metrics that were dealing with trends.

The next phase of the experiment consisted of individual interviews. The results of the questions, determined in section 5.1 are presented below:

1. ***Are there any additional functionalities that you think that can improve the use of this console?***
   The answers of this question were ranging on different aspects. Participant 4 responded that *'I think the console can be more user friendly and having more interactive features'*. Also, apart from the design enhancements, all the participants believed that the console can be much more effective in managing the agile game development process by providing feedback in situations where the game producer should take actions. *'As the console currently is, I still need to rely on external sheets or plans'* was the response from

Participant 1. Furthermore, the response *'I would like to see some guidelines to which metric to view on which occasion'*, from Participant 2 was also given from Participant 1 and Participant 3. Another interesting response from Participant 1, Participant 2, Participant 4 and Participant 6 was to integrate a time tracking software, so new metrics could be determined. Another potential addition on the console was proposed from Participant 5 stating *'I would like to see labels on the Y-axes, so I can understand better what is represented'*. He also added *'I think adding a description on every metric is handy, so I can have a clear view on what I am observing'*. A response that Participant 3 also proposed. Finally, Participant 6 added that *'I would like to be able to visualize the trends based on future references, so I could be able to prevent future hazards'*.

2. ***Do you believe that using this console, you would have avoided some issues during the game development?***

   Participant 4 stated that *'Yes, I believe I would, because the proposed metrics give me the ability of steering the game development and understand if there are any issues within the game development that need to be resolved'*. Participant 5 also believed that the console would have prevented some issues, stating *'The charts that are presented pinpoint the necessary information for a game producer and also provide an overview of the games' progress'*. Participant 6 also supported that the console would have been able to prevent problematic situations, saying *'Yes, I think I would have an idea to plan better future games, by pinpointing which discipline will be more problematic and which features will be more demanding'*. However, most of the rest participants agreed that the current potential of the console is not helping in understanding the situation clearly and take actions in problematic situations. In particular, Participant 2 responded *'No, because the console should provide some feedback, because there are differences in how things work on each project. So, I am not sure, whether I had to take actions or not'*.

3. ***Which of the metric(s) proposed from the console you believe is (are) more important? Why?***

   This considered to be a very interesting question, because almost all the participants gave a different response and for different reasons. Participant 1 and Participant 2 agreed that *'From the proposed metrics, the ones that correspond to time are more important'*. Participant 1 stated *'because this is how I imagine the progress can be tracked'*, on the other hand, Participant 2 explained that *'because they relate to performance'*. Participant 4 believed that *'The metrics seem very important, however I believe that the goal completion per development stage metric is of most concern. The reason is that I can check how far in development are the games'*. Participant 6 also agreed that goal completion per development stage is the most important, because *'I am able to evaluate what the status of the development is, compared to the deadlines'*. Although the previous participants were focusing on metrics that are independent of bugs, Participant 3 responded that *'I think the bug-related metrics are more important, because you can easier identify if there is a problem and where to take actions'*. An interesting response on this question was given from Participant 5 who stated *'For me and I think for the company that I work, the most important metrics are the ones that concern the disciplines. Defects per discipline metric helps me realize how much workload each discipline has so I can take actions. Also, the Defects per Discipline per Development Stage metric helps me tackle issues in every stage and take actions to improve the process of the game development'*.

4. ***Do you believe that the metrics proposed on the console are sufficient to fulfill the purpose of the console?***

   The majority of the participants believed that the metrics are sufficient. Participant 2 stated that *'I believe the metrics are elaborate and cover the important aspects of the game producer's role in game development'*. A statement that Participant 6 also supported. However, Participant 1 stated that *'I believe that the metrics are sufficient, however I believe that they focus a lot on defects and I would like to see some more metrics in different aspects of the game development'*. Also, Participant 3 added that *'Whereas the metrics are enough, I am missing some metrics that correspond to budget'*. On the other hand, Participant 5 claimed that *'They seem more than enough and I am overwhelmed'*, a statement that Participant 1 believed also.

5. ***What additional metrics would you like to observe from the console?***

   Most of the participants agreed that an important metric that would add value to the console is the **Percentage of Feature Completion**. Participant 4 explained *'This way, I would be able to observe how far our features are, compared to time the development team has spent and the development stage our games are'*. Another interesting response was given from Participant 3 *'Some **budget-related** metrics seem very interesting, because that way I could have an idea how the progress of the games is, compared to the budget'*. A different insight regarding a new metric was defined from Participant 5, who mentioned that *'In order to view the efficiency of every discipline, I would like to observe the **Goal Completion per Discipline**'*. A response that could assist the company that use PivotalTracker was given from Participant 2, explaining that *'I think **Points of Sprint per Owner** would be really useful for our company, since this way would help identify the efficiency of every employee'*.

6. ***Do you believe this console would be able to improve the process of managing the game development? Why?***

   The responses of this question were very positive, regarding the console, because all of the participants agreed that the console can make the process of managing the agile game development more efficient. Participant 3 stated that *'Yes, definitely, because the current software that the company uses makes it impossible to track the progress of the games that are developed, something that this console does'*. Also, another reason that the console would improve the task of a game producer was given from Participant 4 that responded *'I think it would improve the process of managing the game development, because it gives the option to a game producer to keep track of multiple things on the same time and give a clear view of what is happening'*. A reason that Participant 5 and Participant 2 shared as well. Participant 6 added *'The use of this tool seems to give better view for future interventions'*.

7. ***Would you use this console for managing the progress of the projects?***

   Most of the participants acknowledged the potential of the console. Particularly, Participant 3 stated *'I see the potential of the system and I believe that it can add value to the tasks of a game producer'*. However, he continued *'but due to design issues I don't think I am yet to use it'*. Also, similar response was given from Participant 2 who claimed that *'If I could see some information regarding the metrics such as conclusions, I think I could use this tool'*. Participant 4 responded *'Definitely yes, because it is very interesting to view these metrics and compared to our current game development management, it would save me time and effort'*. Another positive response regarding the use of the console was given from

Participant 5. He said *'I believe that a game producer can be really keen on using the console and I believe I would have used it as well'*. Participant 6 also stated *'We see a lot of tools in the market to evaluate which to use. The potential of this console seems one of the tools we might need'*.

8. ***Would you use a custom console, based on your needs, in order to manage the game development process? Or rely on existing software solutions?***

   The responses on this question are categorized in 3 groups. There were participants arguing that existing solutions are more useful. Specifically, Participant 4 claimed that *'I prefer to rely on already existing software, because the needs that the company currently has can be covered from existing tools'*. In addition, Participant 6 responded *'The existing software solutions provide us with the 80% that we need to perform and are sufficient for the time being. However, we are still exploring on which solution is the most ideal for us'*. On the other hand, Participant 1 stated *'I would use a custom console that can present metrics and is able to add new insight on progress of the development'*. He continued *'We are currently working on a custom time tracking software to integrate with our present project management software'*. Also, Participant 2 commented *'By developing a custom system you can adjust your needs and make it as you prefer'*. The last category was defined by Participant 3, who mentioned *'I would prefer to use a customized version of a current system, because I think I would be able to focus on my needs and not to the generic needs of all the companies'*. Participant 5 also agreed on a customized existing software responding *'What I believe would be an ideal solution is to use an existing project management software by adding custom widgets to provide with the information we need'*.

5.4 Enhancements

After the responses were collected, we decided to make some enhancements on the prototype, in order to make the console more usable. The decisions on which suggested improvements to perform and which to drop, depend on the following reasons:

1. The importance of adding value to the purpose of the console. By this, we decide which suggestions would improve the agile game development management.
2. Collective suggestions. During the interviews, there were some suggestions for improving the console that were unanimously stated. An example of suggestion is the addition of a metric that corresponds to the percentage of game feature completion. Also, design enhancements that make the console more applicable.

After taking into consideration those reasons we decided to implement certain improvements.

Design Improvements:

One comment that was mentioned from most of the participants was the fact that the console currently is not very user friendly. So, we decided to implement the following enhancements:

1. Change the development stages view. Currently the bar presenting the development stages is a text that does not provide direct insight on where in development each game is. The enhancement that was implemented was to provide different fonts, depending on the current state of each stage. The greyed out dates mean that the stage should have been finished, the bold dates represent the current stage and the italics dates represent the future stages. By this change, the user can have direct access on the current stage of the game. In Figure 20 the differences between the old and the new design are depicted.

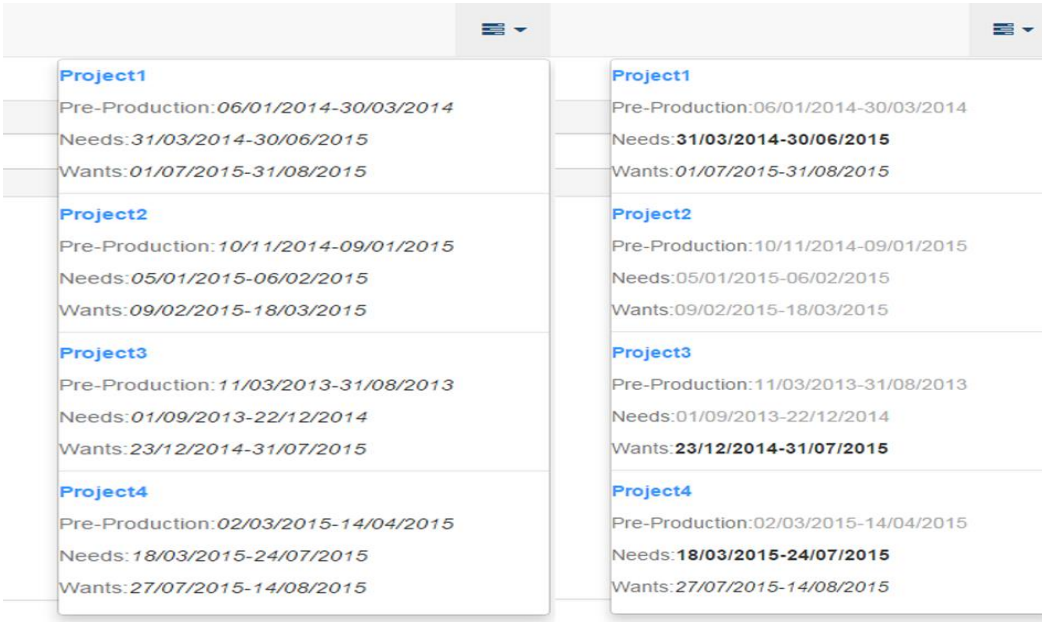

Figure 20. Left: The initial design on the development stages.
Right: The new version of the design.

2. Loading screen on presenting the charts. During the experiment all the participants stated that when a metric is about to be visualized, there is a time period where the console is not responding. This is a known issue and is occurring, due to the fact that the responses that are sent to obtain the data take time to be processed. The AJAX requests demand authentication to provide access to the database and this authorization is time consuming. Because this issue is not depending on the way the console is developed, we decided to add a spinner that can assist the users understand that the console is active and the metric will be visualized as soon as all the data are obtained.

Functional Improvements:

3. New metric. One of the questions that the participants responded was dealing with additional metrics. Although the responsed depend on personal preference, we decided to include a metric that was proposed from the majority of the participants. This metric is **Percentage of Feature Completion**. This metric gives the opportunity to the game producer, observe how completed each game feature is and draw conclusions regading the time and budget that is remaining for the feature. The following SQL-query represents the data that we obtain to calculate this metric:

```
SELECT
      (SELECT COUNT(backlogItem.featureID)
      FROM feature INNER JOIN backlogItem ON feature.ID =
      backlogItem.featureID
      WHERE backlogItem.ItemStatus="Accepted")
      /
      (SELECT COUNT(backlogItem.featureID)
```

```
      FROM feature INNER JOIN backlogItem ON feature.ID =
      backlogItem.featureID)*100, feature.gameTitleID,
      feature.FeatureName
FROM
      feature INNER JOIN backlogItem ON feature.ID =
      backlogItem.featureID
GROUP BY
      feature.gameTitleID, feature.FeatureName;
```

Figure 21 presents the new metric that was constructed by this query.

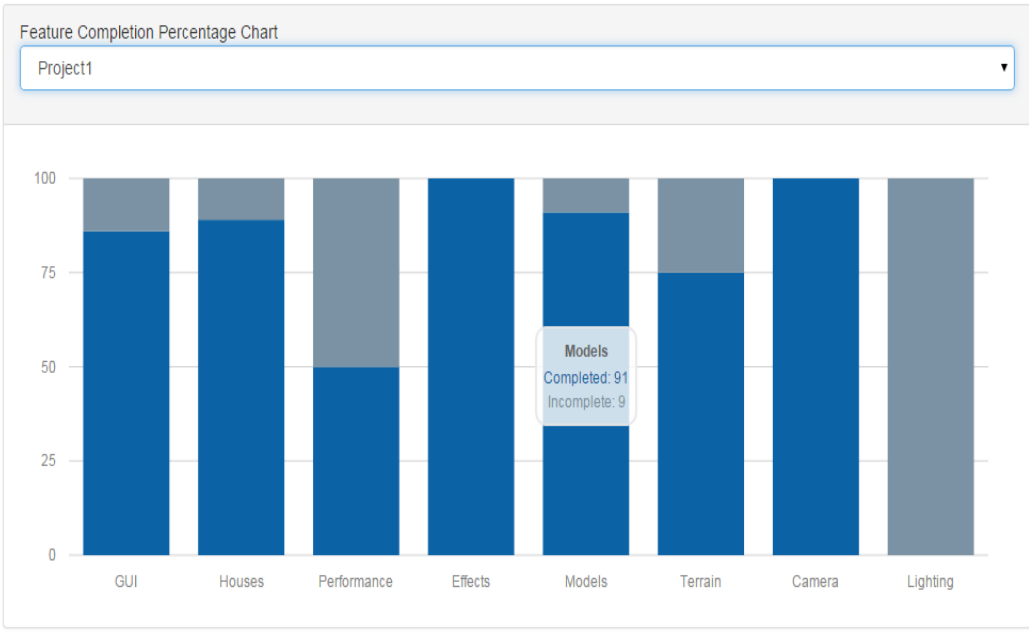

Figure 21. The new metric that presents the completion of every feature.

4. Feedback on charts. According to most of the participants, providing feedback in the charts is the most important addition to transform the console from a prototype to an applicable solution. By adding this functionality on the console, the game producer could be able to identify problematic situations and take actions more direct than before, as stated during the interviews. In order to implement this additional functionality, we identified different alternatives to understand, whether the projects are on track or there are problematic situations.
   1) Definition of static indicators. By this algorithm we create a constant X that is compared to the values of the metric like "more open bugs than X". If the value that we plot is above X then a feedback is given. This algorithm is easy to implement, because only a comparison is executed, however the definition of X is difficult.
   2) Check the norm of the results. This algorithm relies on statistics and the definition of the average μ and the normal distribution σ are required. Using this algorithm, an area of values is compared to the results that the metrics are visualizing and therefore the feedback depends on the results. However, in order to implement this algorithm, the data that are collected need to be normally distributed. Also, the implementation of the

algorithm is rather more complicated, because additional values need to be processed and defined. An example of this algorithm is:

- if $(\mu - \sigma \geq bugs \ OR \ \mu + \sigma \leq bugs)\{ provide \ feedback\}$

3) Observe the path of a trend. This algorithm can be executed only by data that that are dependent on time and can be plotted in curves. The algorithm observes the path of the curve and provides feedback in situations where for example the curve was decreasing and suddenly it started increase.

After examining these alternatives, we decided to select the algorithm where the norm of the results is being examined. The selection of this algorithm was chosen, because we leave out using an arbitrary number as the 1st algorithm does and also most of our metrics do not depend on time, as the last algorithm proposes. An example of given feedback is given in Figure 22.

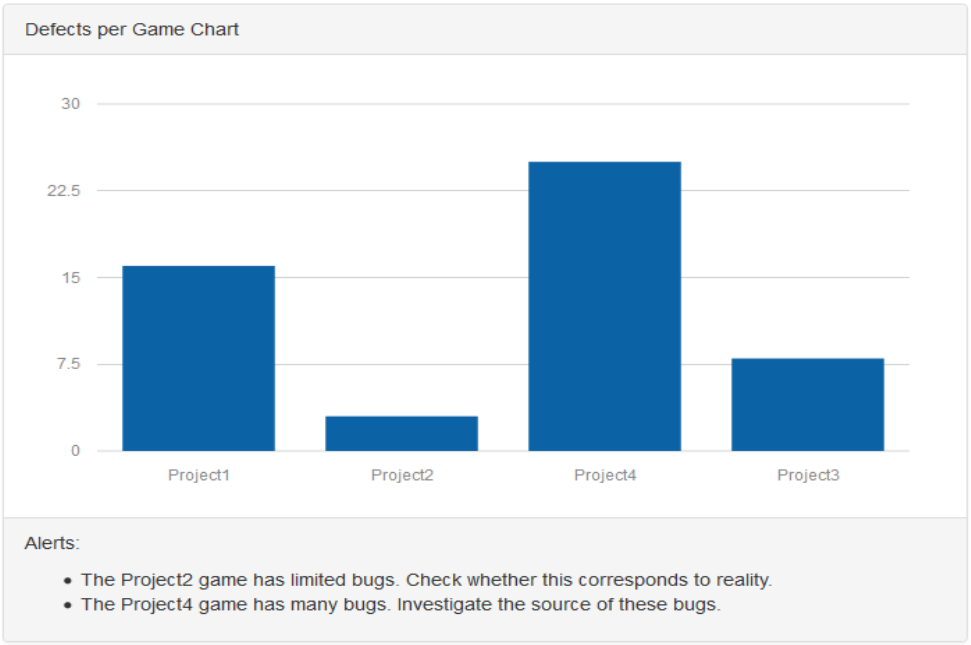

Figure 22. The feedback of the defects per Game metric is observed on the bottom of the chart.

6 Discussion

The last chapter of the thesis presents the conclusions that we were able to draw in respect to the research questions. The conclusions are determined from the literature study that we performed and also the experiments that we executed. Also, the limitations and threats to validity and reliability are presented. Finally, we propose additional work that can improve our research.

6.1 Conclusions

Through the literature study and the design and development of the prototype console, we can draw conclusions regarding our research questions:

- Firstly, we explored the literature study to identify the key responsibilities of a game producer. As can be observed from the mind-map we constructed in Figure 5, a game producer has multiple responsibilities. We concluded that the internal responsibilities, in order to produce a successful game title lie under the game development management. During our research we focused on the agile game development, because as concluded in Chapter 2, agile game development is a trending development method for small and medium game studios.
- Also, through literature study, the interviews and also from personal experience in a game studio as game producer, we were able to identify bottlenecks that game producers face in managing the game development process. These barriers are observed mainly when attempting to keep track of all the activities that occur in a game development process. By this, the coordination of the game producer can be less effective and can result in failed or cancelled games.
- Furthermore, we proposed metrics that aim to improve the management of agile game development. Because the metrics are extracted from a conceptual model, the definition of new queries is possible. Also, the schema can be obtained from specific companies and be customized, allowing for more queries. Throughout the interviews with the case study participants, we can conclude that these metrics are sufficient to improve the coordination of agile game development.
- Finally, we developed a console that aims to assist game producers in managing the agile game development. The evaluation of the console, from the case study participants, helped us conclude that the software solution that we propose has potential of being introduced in the field and can assist game producers. However, as it is still in prototype phase, the console is not directly applicable.

6.2 Limitations

Our research is based on some assumptions or some decisions that we made, which create threats to validity and reliability. To begin with, there is limited scientific research for the field that we explored. So we were unable to substantiate the information we extracted from the web and books that present personal experiences from experts. Furthermore, the case study was performed by not solely game producer, due to the fact that the access to game producers within the Netherlands was limited. So, the evaluation is not performed entirely by potential users and also the number of participants does not allow us to draw robust conclusions on the potential of the console.

6.3 Future Work

According to the results that we obtained and the conclusions that we drew, the field shows potential and it can be further explored to improve the role of a game producer. First of all, defining additional metrics from the ones that we propose would convert the console in a tool that can be used in further activities, apart from agile game development management. Also, we relied on data that we obtained from a single tool, namely PivotalTracker. A further investigation on other tools and integrating them within the console, can provide new functionalities. During our experimentation, we relied on game producers or developers with managerial activities within the Netherlands. Further experimentation, with game producers could make the evaluation process of the console more concrete.

7 References

1. Baer, R., and Burnham, V. "Supercade: A visual history of the videogame age, 1971-1984". MIT Press, 2001.
2. Lowood, H. "Videogames in computer space: The complex history of pong." IEEE Annals of the History of Computing 31.3 (2009): 5-19.
3. Herz, J.C. "Joystick nation: How videogames ate our quarters, won our hearts, and rewired our minds". Little, Brown & Co. Inc., 1997.
4. Bethke, E. "Game development and production." (2003).
5. Peterson, S. "Digital Game Sales Growing 33%." GamesIndustry.biz. 29 Mar. 2013. [Online; Retrieved 4 May 2015].
6. Michaud, L. "World Video Game Market: Eight Key Trends to Watch in 2014." Game Summit. DigiWorld, 12 Dec. 2013. [Online; Retrieved 4 May 2015].
7. "Top 100 Countries Represent 99.8% of $81.5Bn Global Games Market." Newzoo. 23 June 2014. [Online; Retrieved 25 May 2015].
8. Nicholas, K., and Halprin, R. "What Does a Game Producer Do?" WiseGeek. Conjecture, 20 Apr. 2015. [Online; Retrieved 4 May 2015].
9. Crecente, B. "Pirates of the Caribbean Game Canned as Layoffs Hit Propaganda CONFIRMED." Kotaku. Kotaku, 14 Oct. 2010. [Archived; Retrieved 4 May 2015]. http://web.archive.org/web/20130615070634/http://kotaku.com/5664021/rumor-pirates-of-the-caribbean-game-canned-as-layoffs-hit-propaganda
10. Tamaki. "Heroes: The Video Game [Cancelled - PS3, Xbox 360, Wii, PC] - Unseen64." Unseen64. 16 Feb. 2015. [Online; Retrieved 4 May 2015].
11. Makuch, E. "Young Justice: Legacy Canceled for Wii, Wii U." GameSpot. 28 Oct. 2013. [Online; Retrieved 5 May 2015].
12. Briers, M. "Gone Home Console Version Cancelled." We Got This Covered. We Got This Covered, 4 Mar. 2015. [Online; Retrieved 4 May 2015].
13. Plunkett, L. "Star Wars: Battlefront Online Binned As Developers Laid Off." Kotaku. 9 Apr. 2010. [Online; Retrieved 4 May 2015].
14. Goldfarb, A. "NBA Live 13 Canceled - IGN." IGN. Ed. 27 Sept. 2012. [Online; Retrieved 18 May 2015].
15. Makuch, E. "NBA Live 13 Canceled Six Days before Planned Release." GameSpot, 3 Oct. 2012. [Online; Retrieved 18 May 2015].
16. MacDonald, K. "Comically Terrible Ashes 2013 Game Officially Cancelled - IGN." IGN. 28 Nov. 2013. [Online; Retrieved 18 May 2015].
17. Cobbett, R. "The 10 Worst PC Games Of 2013." PCGamesN. 14 Dec. 2013 [Online; Retrieved 18 May 2015].
18. Healey, N. "Bethesda Confirms Prey 2 Cancelled - CNET." CNET. 30 Oct. 2014. [Online; Retrieved 18 May 2015].
19. "EA Unveils The Lord of the Rings, The White Council; Next Generation Lord of the Rings Game Introduces First Epic Open World Adventure." Electronic Arts News RSS. 13 July 2006. [Online; Retrieved 18 May 2015].
20. Hatfield, D. "White Council Adjourns - IGN." IGN. 2 Feb. 2007. [Online; Retrieved 18 May 2015].
21. Remo, C. "LOTR: The White Council Cancelled, Producer Gray Let Go." Shacknews. 5 Jan. 2007. [Online; Retrieved 18 May 2015].
22. Chandler, H. M. "The game production handbook". Jones & Bartlett Publishers, 2009.
23. Bonin, H. "Gamasutra: Harvard Bonin's Blog - The Future of Being a Video Game Producer."
24. Heney, E. "How to Make Flappy Bird, #1 App – Interview with Game Developer Dong Nguyen: Updated." 31 Jan. 2014. [Online; Retrieved 25 May 2015].
25. French, M. "Inside Rockstar North - Part 2: The Studio." Inside Rockstar North. 4 Mar. 2013. [Online; Retrieved 25 May 2015].
26. Weber, R. "On Reflections: First Interview with the Ubisoft Studio's New MD." GamesIndustry.biz. 28 Feb. 2013. [Online; Retrieved 25 May 2015].

27. Warren, T. "Microsoft Launches Xbox One SDK to Let Any Developer Build Apps for Its Console." The Verge. 4 Mar. 2015. [Online; Retrieved 25 May 2015].
28. Studyweb.com Team. "How to Homebrew Wii Games: 73 Tips, Tutorials and Resources - StudyWeb.com." StudyWebcom. 29 Jan. 2008. [Online; Retrieved 25 May 2015].
29. Turke, Z. "Bargaining with Apple: Understanding the IOS Developer Program License Agreement | Law of the Level." Law of the Level. 19 Feb. 2015. [Online; Retrieved 25 May 2015].
30. Rollings, A., and Adams, E. "Andrew Rollings and Ernest Adams on game design". New Riders, 2003.
31. Keith, C. "Agile game development with Scrum". Pearson Education, 2010.
32. Royce, W. W. "Managing the development of large software systems." Proceedings of IEEE WESCON. Vol. 26. No. 8. 1970.
33. "Waterfall Development." 'What Games Are' [Online; Retrieved 25 May 2015].
34. Bates, B., and LaMothe, A. "The game design: The art and business of creating games". Premier Press, 2001.
35. Miller, P. "Top 10 Pitfalls Using Scrum Methodology for Video Game Development." Gamasutra Article. [Online; Retrieved 25 May 2015].
36. Beck, K., et al. "The agile manifesto." (2001): 2009.
37. Cohen, D. S., and Bustamante S. A. "Producing games: from business and budgets to creativity and design". CRC Press, 2012.
38. Schultz, C. P., Bryant, R., and Langdell, T. "Game testing all in one". Course Technology, 2005.
39. Redavid, C, and Farid, A. "An Overview of Game Testing Techniques."
40. Koepke, B., et al. "Agile Game Development".
41. Schultz, W. "The Importance of Version Control Management in Game Development." About.com. [Online; Retrieved 8 June 2015].
42. Kohavi, R, Rothleder, N. J., and Simoudis, E. "Emerging trends in business analytics." Communications of the ACM 45.8 (2002): 45-48.
43. Coker, F. "Pulse: Understanding the Vital Signs of Your Business". 2014. 41-42.
44. El-Nasr, M.S., Drachen, A., and Canossa, A. "Game analytics: Maximizing the value of player data". Springer Science & Business Media, 2013.
45. El-Nasr, M.S., Drachen, A., and Canossa, A. "Intro to User Analytics." Gamasutra Article. 30 May 2013. [Online; Retrieved 28 May 2015].
46. Bjornholt, M., and Farstad, G.R. "'Am I rambling?' on the advantages of interviewing couples together." Qualitative Research (2012): 1468794112459671.
47. Kitchenham, B., and Pfleeger, S.L. "Principles of survey research: part 5: populations and samples." ACM SIGSOFT Software Engineering Notes 27.5 (2002): 17-20.
48. Wohlin, C., et al. "Experimentation in software engineering". Springer Science & Business Media, 2012.

8 Appendix

The Microsoft Access Diagram