SENG 403
Software Development in Teams and Organizations
Topic 5: Game Development - Paper Assignment

Thomas Condon, George Nikolov Dimitrov, Jessica Huynh, Lisa Jacobe, Gabriela Jurca, Riane Vardeleon, Constantine Vlahos
Department of Software Engineering
University of Calgary
Calgary, Canada

Abstract - In this paper we will be discussing the process of design and development steps taken to create an interactive video game, known as Game development, and comparing it to Software engineering principles. The focus of the paper will be to go over the differences in regards to requirements elicitation, design, and testing. We will also touch on the creative interactions design process, what it is, and how it fits into the game development process. Lastly the commonly used frameworks for game development will be discussed in the later parts of the paper.

Keywords—

I. INTRODUCTION (*Heading 1*)

Game development is the process of design and development taken to create interactive games while Software Engineering is the use of engineering principles in the design, development and maintenance of software. For the purpose of this paper we will be discussing Game development in respect to videogames, the software related to their creation and comparing the different Game development and Software Engineering analogies. Though this does not limit the methods and techniques used to develop video games to be restricted solely to those that originated from the software development industry.

II. REQUIREMENT ELICITATION

A. Background

Requirement Elicitation is more commonly known as requirement gathering; it is an important step to finding out what the stakeholders, customers and potential users want from the system. Using good interpersonal communication and collaboration with all the involved parties helps support both the developers and the users, it makes it easier to understand how the process works and helps to explain the different methods and techniques used to gather the requirements. Examples of a few of the mentioned techniques and methods would be use cases, task analysis, scenarios, questionnaires and task observation [Barry]

After gathering the requirements using one of the listed or other methods, we then use requirement validation to check for correctness. The client's expectations, completeness of all scenarios described, consistency (to avoid contradicting requirements), clarity to ensure unambiguity, realism of implementations, traceability of requirements and verifiability through repeatable tests are all determined and proven/ disproven using requirement validation [Barry]. All of this is to ensure the software meets the standards that are set by either the company or the stakeholders.

Now that the standards are met we must also consider the functional and nonfunctional requirements in the development process. Functional requirements gives us the ability to see what user tasks the system will be supporting, such as being able to play in PvP (Player vs Player), vs. AI (Player vs Computer), custom modes, or even being able to use game money or real-world currency to buy equipment and items in-game. Non-functional requirements are not related to the functionality of the system, but are properties and qualities of the system. This is seen in things such as usability, reliability, performance, supportability, and portability, just to name a few. Other types of non-functional requirements include constraints or pseudo-requirements, like implementation, interface, operation, packaging and legal issues [Bruegge]

Other than the given examples of requirement elicitation there exists other types of requirement elicitations, such as greenfield engineering, re-engineering projects, and interface engineering. Greenfield engineering does not require any removal or remodelling of existing models in order to be created, this is usually decided by user needs. Re-engineering projects have the same purpose as greenfield but may also extend functionality of the software and unlike greenfield is requested by technology enablers. Re-engineered projects are found in As-is scenarios, which is more current and based on the users descriptions of the system. Interface engineering is remodeling existing UI designs based on technology enablers and the market needs. Both greenfield and re-engineering projects can typically be found in visionary scenarios, which is futuristic and cannot be done by users and developers alone [Bruegge].

III. GAME DESIGN

A. Background

Game design is a process in a game's development that determines every aspect of how the game will function [Rouse]. These aspects are the theory and practical necessities of any game. Examples of such aspects are possible course of action for users along with the results of said action, information that needs to be relayed to the users, difficulty of the game and the win or lose criteria. All the examples given describe gameplay and mechanics, in which the importance of the game design step is explained.

The game design process is described as being player-centric, which is a design method centering around the player [Brathwaite]. This means that designers focus on entertaining the players and empathizing with them [aarongdj.blogspot.ca]. Game designers end up spending most of their time thinking about what users would like to see in a game, and how they would like to interact with the game. The main challenge for the designers however is how to make their game interesting. One solution can be derived by answering the question of what players want. Players want games that are either challenging, provide an emotional experience, are used to socialize, allow for bragging rights and others or a combination of the above. Designers can analyze these successful games and try to understand why such a game was able to capture player's interests and attention, while hopefully not just cloning the idea. Most importantly though the game designer should also be able to enjoy the game he is making and not just taking into consideration what other players want.

B. Examples of Good Game Designs

A good game design leaves the players with a good experience from playing a game. Good experiences can come from good game design choices such as continuous challenge, flexibility, immediate and useful rewards, and more [21].

The continuous challenge is the aspect of the game that keeps the players interested on playing a game. This can be achieved by setting clear, short-term goals appropriate to the level of the player and the context within the game. Quests are examples of this. Giving players quests to do fulfills the continuous challenge aspect of the game. Quests can start as easy at firsts and becomes harder upon completion, but not too hard that it becomes inappropriate for the level of the player and the context of the game.

The flexibility makes sure that players can choose from a variety of ways to accomplish each goal. Adding flexibility helps in making players interested in playing game as they get to finish tasks in a way that they prefer more. Players are not limited to what actions they can perform, which makes the players more engage in the game as they can exercise their problem solving skills to solve certain goals. For example, being able to finish a quest through team or individual efforts. This example shows two different ways to accomplish certain goals within a game. Players can choose to do the task alone or with other players. This way, players are given the flexibility of the game where they can interact with other players, if they choose to or not.

Another good game design choice is giving players immediate and useful rewards. Instead of just points towards victory, successful players can be rewarded with new capabilities, a new part of the game world to explore or even a new task. These are surprisingly motivating, as the point of the game is not just to win it, but also to keep playing.

C. Examples of Bad Game Designs

Bad game design choices is the cause of unsuccessful and poorly designed games that do not provide its players good experiences of the game. These bad designs should be avoided to avoid failures to a game. Some bad design choices are games without maps, trial-and-error games, and games you cannot pause.

First person shooter (FPS) games are the type of games that needs a map. If there is no map, then it decreases the gameplay experience for players. Games without maps leaves the users confused on where they are in the course of the gameplay. The maps provide valuable information in games like FPS. The map is the guide for players on what steps to take and gives them the navigation they need throughout the gameplay.

Another bad game design choice are games that has a time limit to accomplish certain tasks. These games are the trial and error type of games, which are frustrating to the players. This type of games requires a lot of dedicated time and trials to accomplish certain set of tasks. Giving players time limits to solve certain levels or quests add pressures to the players. Sometimes, these types of games are not realistic, which causes players to stop from playing a game.

Games that cannot be paused are also another bad game design choices. Imagine yourself in a middle of defeating a boss in a gameplay, then suddenly you needed to take a break to take a trip to the washroom. This leads you to the action of pausing the game. In games where there is no pause functionality, players are forced to keep playing the game till the very end.

IV. CREATIVE INTERACTIVE DESIGN

A. Background

Creative Interaction Design is a design process used in the creation of interactive products. These products are typically designed to facilitate communication and interactions between people as well as interactions between individuals and products. [Rogers, Jarvinen]. Interaction designers are responsible for creating and maintaining environments where the users interacting with the system can gain a valuable and meaningful experience from it. The user experience pertains to the overall impression people get

during the use of the system/product. How good or bad this experience is is determined by every aspect of the product, from the obvious details (i.e. how it works, etc) all the way down to the more subtle details such as the shape of a tv remote which can determine how comfortable it is. As Rogers et al has said "it is important to point out that one cannot design a user experience, only design for a user experience. In particular, one cannot design a sensual experience, but only create the design features that can evoke it." This all points toward the ultimate goal of interactions design, that being to enhance positive user experiences, like enjoyment and how well a user is engaged by it, all while reducing negative experiences such as annoyance and frustration. In some cases even negative experiences can help reinforce gameplay. One such example would be *Dark Souls*, a game that was designed to be as brutal as possible to the users, so that players feel a sense of accomplishment when they finally succeed.

To further explain gameplay decisions we can classify the different types of interactions. One such type is social interactions. Games with this type of interaction tend to provide a social experience in the gameplay and the end result solely depends on the people involved. The game *Cards Against Humanity* would be a good example of a social game. In this game players select a card from their hand to complete a phrase, the winner of the round is determined by the personality of the player who initially read the incomplete phrase. To keep the game balanced players anonymously play response cards. The other type of interaction is one that focuses on participation and execution of commands [Christa], this one is more common for single player video games, where players participate in gameplay to "beat" the game using a chain of available commands. By deciding which interaction model fits the game being designed, designers can focus on how to make the game more enjoyable for the intended audience.

B. Creative Interaction Design and Game Development

Interaction design is a crucial portion of the game development process. This can be seen when you take a look at the individual components. Interactions design is a very human-centric design process, and as previously mentioned game design itself is player-centric. Comparing this to video games, which are defined as an *interactive* entertainment media platforms, it goes to show the importance of the interaction design process. Just as there is a difference between designing and building a bridge, there is also a difference between designing a game and engineering the software behind it. As per Jarvinen et al, game design is a subset of interaction design that has a focus on the facilitation of play and games as particular entertainment systems [Jarvinen]. Creative Interaction Design is a process that follows iterative design procedures, where playtesting is fundamentally important. Much like Software Engineering, game development practices which invest in interaction design, are iterative and have a strong emphasis on testing.

In general the interaction design process follows these steps:

1) Establishing system requirements.
2) Designing alternatives.
3) Prototyping.
4) Evaluation.

Each of these four steps is intended to provide a basis for the next step and are supposed to be repeated in an iterative manner [Rogers]. The measure of usability of an iteration provides feedback that will be essential for future iterations.

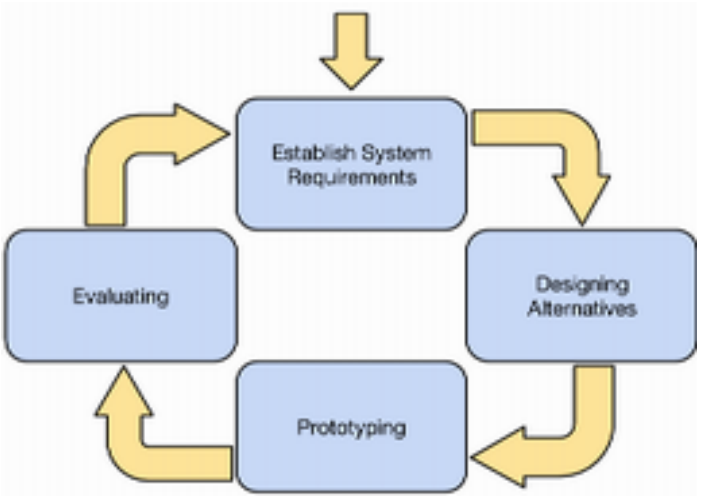

Fig. # The iterative creative interaction design process

Creative Interaction Design has three main characteristics [REF]:

1) Focus on users.
5) Has specific usability and user experience goals.
6) Iteration of the design process.

In Interaction Design the users or players are involved from the beginning of the design process, where in the needs and requirements for the system are identified, right up until evaluation.[22] In establishing system requirements, particular usability and user experience goals for the game are identified. These are then used to evaluate the design later on in the process. Designs are then refined through different iterations as a response to user feedback.

V. TESTING

A. Background

Games are incredibly complex pieces of software and like any software system debugging is a critical part of the development process. Overlooked bugs that remain in released software could needlessly waste system resources and slow overall performance, while others could cause bizarre graphical glitches, crash the game, or even corrupt saved files on the consumer's console or PC

B. Types of Games Testing

There are two principle types of Game Testing, playtesting and quality assurance. Playtesting is intended to make sure that the game is balanced and is interesting to play. The objective of playtesting is not to find bugs, but to analyze the behaviours, pacing, and overall feel of the game as well as to measure it's relative difficulty. This can be incredibly subjective, making it difficult to define from a software engineering perspective. Quality assurance on the other hand primarily focuses on searching for software bugs, or game breaking glitches.

Quality assurance often involves hiring teams of game testers to go over any and all imaginable scenarios that occur within the game, multiple times in most cases. The main purpose of this is to manually find and document the occurrences of any breaks or glitches within the game. An example of one scenario would be walking/running a character into every wall in a specific area of the game to search for holes in the collision geometry, to find any unintended exploits that would allow a player to cheat their way through a game, or to find combination of items or commands that can cause the game to crash. The problem with this however is the sheer number of possible scenarios that could exist within the game environment requires incredibly thorough testing, and even then there is still a good chance that bugs of varying severity will remain. In most cases game developers find themselves releasing software patches post-release once a missed bug is discovered by the player base. This is also an effective but risky method to finding bugs, because while it is cheaper than hiring testers, game breaking bugs can turn people off from playing a game even after it has been fixed due to the negative experience.

Fig. # This is a guard from Dragon Age. A graphical bug has prevented the head geometry from loading.

VI. FRAMEWORKS

A. Background

A framework is a real or abstract structure which serves as the support for building something useful [1]. Frameworks provide programmers with utility methods, as well as application programming interfaces (API's) with other programs and resources. Additionally, they provide access to a collection of code within a library that has been written in a specific programming language like JavaScript. Overall, frameworks are just the basic skeleton and still require a fair bit of implementation from the users end. Having access to a framework provides developers with a number of benefits, one advantage is that the developer can reuse code which has already been tested by other people in the programming community. This decreases the development time and cost and increases the reliability of the program. Another benefit is the modularity of the code is increased, this is due to the lower level tasks being handled by the framework.

In game development frameworks are heavily used in the game industry, programming education, and for personal projects. In game development specifically frameworks are used as a set of functions that are loosely created to handle tasks such as rendering, physics, audio, user input, collision detection and so forth. Frameworks speed up the game development process in such a way that developers do not have to continually reinvent the wheel each time they create a game.

Most often in game development, frameworks are implemented in Game Engines, and then the game engine is used to build games significantly more rapidly than by just using a framework by itself [4]. A game engine is a set of tools and scripts that contain more powerful logic in order to perform more complex tasks and to manage the workflow of the program [3]. However, the terms "framework" and "game engine" are often used interchangeably in practice, we will be using "framework" to refer to both unless otherwise stated.

B. Types of frameworks

In literature, two types of game development frameworks have been described, and their uses are as follows [Wu 2011]:

Fig. # A screen shot of Alice, a framework that is targeted for novice developers.

1) Novice - Provides a visual interface for customizing game templates, and not much programming is required

to create a game. The user is introduced to programming concepts through visually manipulating objects.

2) Developer - These are toolkits that support development of high quality 2D/3D rendering, special effects, physics, animations, sounds playback and network communication in languages such as C++, C# and Java.

C. Recommendations

Some aspects that can be considered for choosing a good framework are given as follows [Wu 2011, 3]:

1) Features - Game development will be sped up if the framework includes collision, physics and handles input.

2) Usability - For high usability, the framework should have comprehensive documentation and an active community in order to help answer questions.

3) Skill Level - Non-coders should consider a novice type with a high-level API, with interactions like drag and drop. Seasoned developers may want types with low-level access where the source code of the framework is available.

4) Technical Aspects - The framework may be restricted to particular environments. For example, the XNA framework only runs on Windows.

5) Distribution - The ease of multi-platform distribution should be considered.

D. Notable Examples

1) XNA Game Studio

This is a Microsoft technology developed back in 2006 and this was supposed to allow indie developers an easy way to get into developing for the Xbox 360 [17]. Whether the developers were a studio or a 14 year old kid at home, this brought an onslaught of new developers and games to the market. This created a special market for games made with the XNA framework on the Xbox marketplace. The marketplace allows people to publish their games without having to go through layers of bureaucracy just to get their name out there for a small fee. Unfortunately Microsoft has discontinued the use of the XNA framework and it officially stopped supporting it as of April 1st, 2014. Notable games that were created using the XNA framework include Schizoid and Terraria.

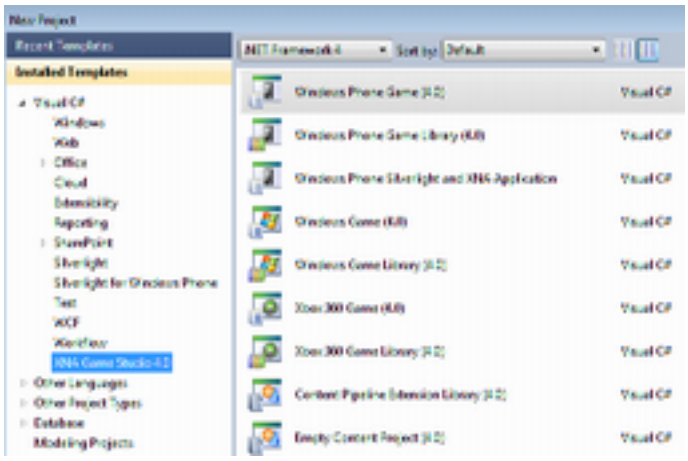

Fig. # A screen shot of the start-up screen for creating an XNA project through Microsoft Visual Studio [REF]

2) Unreal Engine

This widely used powerful game engine was developed by Epic Games and was first seen in the game titled Unreal in 1998 [18]. The engine is written in C++ and is currently on Unreal Engine 4. There is a development kit that Epic Games has released called UDK and this is a free version of Unreal Engine 3 for the general public with slight fees related to it when your product goes commercial. Some big names that use this engine are Epic Games own Gears of War franchise, the Borderlands and Bioshock franchise from 2K Games, and Electronic Art's Mass Effect franchise.

Fig. # A visual comparison of the graphics between Unreal Engine versions 1, 2 and 3 [23]

3) Unity 3D Engine

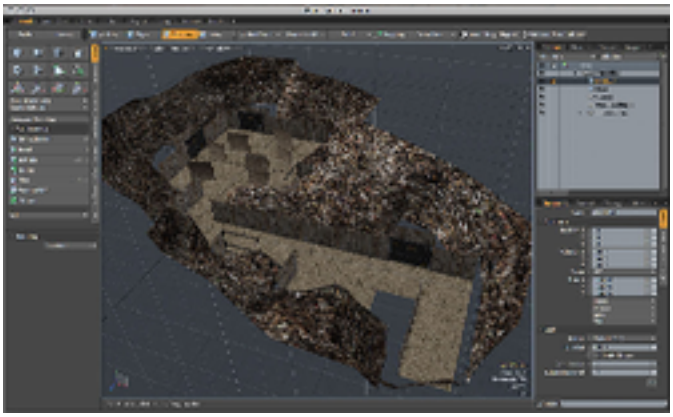

Fig. # A screen shot of the start-up screen for creating an XNA project through Microsoft Visual Studio[24]

This engine was built to be an all-in-one game development package, the engine comes with it's own integrated development environment (IDE) and was originally started back in 2005 as a Mac OS X supported game tool and now it is a multi-platform game engine [16] with the exception of Linux. There are two versions currently available, the free version and the Pro version which adds many features and allows a one-button click to deploy your application over a multitude of platforms. Notable games that are run using this engine are Temple Run, Slender: the Eight Pages and Game of Thrones: Seven Kingdoms.

E. Applications

Usually when people think about frameworks they think that the only use for them is to make games that entertain the gaming enthusiasts. While for the most part this is true, it is also used to create games which are mindless time wasters like *Flappy Bird* where the point of the game is to tap the screen to make your bird endlessly go through as many pipes as possible. Some developers have put effort into creating games that aid in research, are a source for military recruitment, education, fitness and simulators meant to simulate real life scenarios.

One example of a game developed for research purposes is a game called *Play To Cure: Genes in Space,* this game was developed by Cancer Research UK and it uses players to analyze paths through what the game calls Element Alpha [4]. Element Alpha is just a replacement of DNA microarray data, DNA microarrays are a collection of DNA spots that allow scientists to measure the expression levels of a large number of genes simultaneously [20]. As a player you create routes through these small maps trying to collect as much Element Alpha as possible, these paths the players create help scientists spot patterns in all the data they have collected to showcase the DNA faults from the samples collected. This helps in the process of looking for a new cancer treatment.

Moving away from the science side of things there is also another free game called *America's Army*. This game simulates US Military life, and is an FPS meant to get you interested in military life and sign up for the US military. Chris Chambers, former deputy director of development for *America's Army* admitted that this was a recruiting tool [19]; even on the game's website they have links that take you to the US Army's website. This game starts off with the player being a recruit and going about taking school tests at boot camp, learning about different weapons, military vehicles, etc. Later on players choose missions to play and cooperate with others online in varying types of style. The game is made as realistic as possible with medical kits, bleed outs, call outs and weapons. This game's primary goal is to get kids hooked on the game so that when they get old enough they will want to join the cadets and pursue a full time career in the military.

REFERENCES

[1] C. Barry, "Understanding the problems of Requirement Elicitation Process: A Human Perspective. Information systems development challenges in practice, theory, and education.," *New York: Springer,* vol. 1, pp. 210-214, 2009.

[2] B. Bruegge and A. H. Dutoit, Object-oriented software engineering: using UML, patterns, and Java (2. ed), Upper Saddle River, NJ: Pearson Prentice Hall, 2004.

[3] B. Wu and A. I. Wang, "Game Development Frameworks for SE Education," in IGIC '11 Proceedings of the 2011 IEEE International Games Innovation , Washington, DC, USA, 2011.

[4] "Play to Cure: Genes in Space," Cancer Research UK, [Online]. Available: http://www.cancerresearchuk.org/support-us/play-to-cure-genes-in-space. [Accessed 24 February 2014]

[5] P. Jalote, "3," in *A Concise Introduction To Software Engineering*, London, Springer, 2008, pp. 37-68.

[6] A. Järvinen, ÈGame design for social networks: Interaction Design for Playful Dispositions. In ACM Proceedings of the 2009 ACM SIGGRAPH Symposium on Video Games, 2008, pp. 95-102

[7] Sommerer, C. "Ubiquitous Gaming Interaction: Engaging Play Anywhere." The art and science of interface and interaction design. Berlin: Springer, 2008. 115-130.

[8] Rogers, Yvonne, J. Preece, and H. Sharp. "What is interaction design?." Interaction design. 3rd ed. Hoboken, N.J.: Wiley, 2011. 1-18.

[9] R. Rouse. "Introduction." Game design theory & practice. 2nd ed. Plano, Tex.: Wordware Pub., 2005. xx-18.

[10] A. DeChamplain, Game Development Journal - Blogspot [Online] Available:http://aarongdj.blogspot.ca/2011/01/player-centric-design-assignment.html

[11] Brathwaite, Brenda, and Ian Schreiber. Challenges for game designers. Boston, Mass.: Charles River Media, a part of Course Technology, 2009.

[12] M. Rouse, "framework," WhatIs, September 2005. [Online]. Available: http://whatis.techtarget.com/definition/framework. [Accessed 23 February 2014].

[13] "Framework," DocForge, [Online]. Available: http://docforge.com/wiki/
Framework. [Accessed 23 February 2014].

[14] J. Freeman, "How To Make A Game Part 1: Picking a Framework," [Online]. Available: http://jessefreeman.com/game-dev/getting-started-
making-games-part-1-picking-framework/. [Accessed 23 February 2014].

[15] J. Voss, "Unity vs. XNA (Engine vs. Framework)," Pixelsoft Games, 20
February 2014. [Online]. Available: http://pixelsoftgames.com/?p=45.
[Accessed 23 February 2014].

[16] Unity Technologies, *Unity - Game Engine* [Online] Available: http://unity3d.com/

[17] Various Authors, *Microsoft XNA - Wikipedia, the free encyclopedia* [Online] Available: http://en.wikipedia.org/wiki/Microsoft_XNA#XNA_Game_Studio_Express

[18] Various Authors, *Unreal Engine - Wikipedia, the free encyclopedia* [Online] Available: http://en.wikipedia.org/wiki/Unreal_Engine

[19] Various Authors, *America's Army - Wikipedia, the free encyclopedia*[Online] Available: http://en.wikipedia.org/wiki/America's_Army

[20] Various Authors, *DNA microarray - Wikipedia, the free encyclopedia*[Online] Available: http://en.wikipedia.org/wiki/DNA_microarray

[21] Various Authors, What makes a good game? - Starting point [Online] Available: http://serc.carleton.edu/introgeo/games/goodgame.html

[22] Various Authors, The Process of Interaction Design - QuickBooks DocStoc [Online] Available: http://www.docstoc.com/docs/48379744/The-Process-of-Interaction-Design

[23] Torinir. (2006, September 14). "Unreal_Engine_Comparision.jpg," in Wikipedia, the free encyclopedia [Online] Available: http://upload.wikimedia.org/wikipedia/en/thumb/8/81/Unreal_Engine_Comparison.jpg/1280px-Unreal_Engine_Comparison.jpg

[24] Unknown. (2011, February 22). "5991_luxology_modo_501_unity_3.2_export_screenshot_game_engine_lg.jpg" [Online] Available: http://www.creativeobserver.com/img/5991_luxology_modo_501_unity_3.2_export_screenshot_game_engine_lg.jpg