2009 Sixth International Conference on Information Technology: New Generations

# Software Engineering Challenges in Game Development

Christopher M. Kanode and Hisham M. Haddad
*Computer Science Department*
*Kennesaw State University*
*Kennesaw, GA 30144*

## Abstract

*In Software Engineering (SE), video game development is unique yet similar to other software endeavors. It is unique in that it combines the work of teams covering multiple disciplines (art, music, acting, programming, etc.), and that engaging game play is sought after through the use of prototypes and iterations. With that, game development is faced with challenges that can be addressed using traditional SE practices. The industry needs to adopt sound SE practices for their distinct needs such as managing multimedia assets and finding the "fun" in game play. The industry must take on the challenges by evolving SE methods to meet their needs. This work investigates these challenges and highlights engineering practices to mitigate these challenges.*

**Keyword:** challenges, diverse assets, process, project management, project scope, publishing.

## 1. Introduction

"At their best, video games stimulate a state of *flow* in the player, engendering concentration so intense that their perception of time and sense of self become distorted or forgotten." Callele *et al* [3]. Successful video games are more than software. A game should enthrall a user. A game that can capture our full attention is one that development companies hope to produce. Out of hundreds of projects, only a few make it into the hands of the consumer. Even then, success is not guaranteed. Games that make it to the store shelf can still fail from flawed code or a lack of entertainment value. The potentially "fun" game that has a beautiful story line and art or has an engaging interface will achieve fame or notoriety based on its software foundation. The video game is a synthesis of code, images, music, and acting that come together into a form of entertainment.

Video games have evolved to be large projects employing hundreds of people and development time measured in years. Unlike most other software application domains, game development presents unique challenges that stem from the multiple disciplines that contribute to the video game. The project combines all of these assets into one application. The work by these multi–disciplinary teams complicates the development process and project management. In addition, game developers must search for engaging game play through the use of prototypes and iterations (finding the "fun").

The video game industry is in need of sound engineering practices to fit their distinct characteristics: multimedia asset management and engaging game play. As games become more complex, and consumer expectations go higher, the video game industry must face the challenges by evolving SE methods to meet their needs. There are many ways for game developers to improve their processes. A rigorous, yet flexible, application of proven SE processes and practices can be applied to video game development in order to better manage the projects and to reduce risks. This work investigates current practice of game development for its unique challenges and the sound SE practices that can help developers mitigate these challenges.

## 2. Overview of Game Development

A common assumption is that game development industry uses the waterfall model. Based on research, this assumption may be incorrect or at least technically it may be a waterfall model but with modifications. To paraphrase Electronics Arts veteran and Emergent VP David Gregory, it is an accepted practice in the games industry to create iterations of software in order to progress towards achieving engaging game play [7]. Clinton Keith, another industry veteran, stated that, approximately a decade after the game industry crash in the mid-1980s, the hardware for gaming increased in power, games took longer to produce and cost more. According to him, informal incremental and iterative development broke down. In reaction, the industry introduced traditional methods of which the waterfall model was the most common [8]. The incremental model would fit much better, as it is a combination of waterfall with iterations. And add in prototyping, since game developers must often experiment to find the "fun" element.

A major issue leveled against the games industry is that most adopt a poor methodology for software creation. Petrillo *et al* refers to data collected by the Standish Group. Only 16% of projects are actually completed on time and on budget. Clearly, there is a problem. Based on the statistics gathered by Petrillo *et al* [10] of completed games, the errors with the greatest occurrence (over the

978-0-7695-3596-8/09 $25.00 © 2009 IEEE
DOI 10.1109/ITNG.2009.74

50% mark) fall roughly under project management, requirements engineering, and risk management.

Prototyping and iterations are, perhaps, inescapable in video game development due to the indefinable nature of creating a "fun" game. Prototyping should primarily be done in the preproduction stage in order to define what the game is. Requirements engineering should take place at the end of preproduction, once the game designers have found the type of game that is to be created. Gathering all the needed requirements will cut down on the number of iterations needed, and mitigate the late addition of features (feature creep). Once the preproduction phase has been completed, the project manager would take the game design document, and formulate a project scope. The manager should take into account a certain percentage of feature creep.

Game developers need to adopt SE practices to address the challenges they face. They need to evolve SE methods and processes to meet their unique needs and to improve game development practices. Adoption and adaptation of SE practices are paramount to the future of the industry, especially in light of rising development costs and a shrinking number of publishers.

## 3. Challenges in Game Development

The challenges that face game development industry are not insurmountable. Many of their problems have been solved in the software industry. As with most issues, the problems must be recognized and understood. Upon gaining insight, management must expend the effort to correct existing problems. Companies may experience brief lulls in productivity, but in the long term, the benefits will outweigh the initial investment in addressing current challenges.

A video game application differs from general software by the preproduction stage and through its extensive use and integration of multimedia assets. Preproduction is a type of requirements gathering from the "customer" (game designers), except that the game designers are creating prototypes and laying out a creative vision of the game. The primary purpose of preproduction is to develop the Game Design Document (GDD) in which the game is described. The descriptions can be storyboards, concept artwork, paper prototypes, software prototypes, and much more. Paul Miller [9] contends that preproduction for a great product involves a multitude of prototyping. Successful preproduction (probably a rare occurrence) defines an exciting and absorbing game. Great preproduction reduces the need to find that elusive element of "fun" during the production stage, and allows the team to focus on implementing the game, rather than experimenting with it.

### 3.1 Diverse Assets

Diverse assets such as 3D models, textures, animations, sound, music, dialog, video and other resources are integrated by the development pipeline into a final product. The multimedia is created by specialists who work in tandem with the programmers whom create the code framework into which all assets fit. The code framework spans over multiple types of software (tools, plug-ins, compression algorithms, runtime components, etc.). A video game project may use software engines for physics and game runtime. During development, specialized tools or plug-ins may be coded to allow animators to integrate models from third party software into the game, and level designers to create the playing field. It is a multi-disciplinary undertaking to produce a game, and is becoming increasingly difficult to manage. Unlike other application domains, the diversity of these assets in game development presents a challenge to building quality game applications.

Players desire more content; content is becoming more complex; and team members require more tools to create and transform content [7]. These factors add additional overhead to the tasks that the project manager must handle. The creation of an efficient pipeline to handle the assets of a video game project is essential and could be a major project in itself.

### 3.2 Project Scope

The scale of a video game can be massive. The following example is taken from a postmortem of *BioShock* [6]. The game took three years to develop, and was scoped to be about two years – approximately 50% longer to develop. At its peak, the companies employed 93 in-house developers, 30 contractors, and 8 on-site publisher testers. The companies produced 3,775 files for the game, wrote 758,903 lines of native C++ code, and 187,114 lines of script code. The teams were located on three different continents: North America, Australia, and China. Coordinating across multiple geographies is a major task in itself. This is listed by David Gregory addressing problems in iterating rapidly [7].

A major failing in the industry is a poorly established project scope which is further compounded by feature creep (adding new functionalities during development that increase the project's size) [10]. Developing a clear, well-formed project scope can only help the overall project. It needs to be developed with a measure of commonsense and risk aversion. Targets must be established and requirements gathered. Requirements engineering has been shown to be invaluable through accumulated evidence and experience, yet many projects fail due to the lack of requirements gathering [3]. Feature creep is a high risk endeavor. Any new functionality should be evaluated carefully. Unmanaged feature creep will lead to missed deadlines and increased errors, defects and the chance of failure. Some level of feature creep is inevitable since it can be tied to game play, and game play **must** be fun. The goal is to evaluate unplanned features based on the potential expense (time, coding, errors, etc.) against added value to game play. Every unplanned feature has potential to derail a project from its timeline.

A project scope is not meant to quell creativity, nor should it be followed dogmatically. It is a tool with which to estimate the time and resources needed to bring its project to a successful conclusion. When properly created, it can help the team develop more realistic schedules and timelines. By the iterative nature of video games, a "perfect" project scope will never be achieved, but it is the goal of the manager to develop a solid scope that will help guide the project to its conclusion.

### 3.3 Game Publishing

In general, bringing a video game to market involves a game development company convincing a game publisher to back them financially. Keith summed it up that game developers create the games, and the publishers pay for development, then market, mass produce, and distribute the games [8]. A common method for catching a publisher's eye is to show a demo of the game showcasing its concept and design. For example, 2K Boston/Australia's *BioShock* project was signed after a graphics demo ran on GameSpot (www.gamespot.com) as an exclusive feature [6]. The process can occur in reverse. A game publisher may have video game rights to a movie, for example, and sought out a development company to take on the work.

In most cases, if not all, contracts are drawn up for the project. The contracts generally have a schedule of deliverables which is used to deliver payments to the game developers. Missed deliverables can cause fines to be levied against the game developers. Publishers expect milestones to be made, and publishers are becoming increasingly risk averse due to rising costs and diminishing returns [11].

The game development industry has to deal with new technology on a regular basis, and more often than the rest of the software industry. As customers, publishers may want developers to take advantage of a new hardware (a video card, for example) or build the game for a new platform [4]. There are risks involved when using new technology (learning curves and potential hardware problems) [10]. Publishers are also driven by the market in their requests for added features or functionality [4]. A hit game of the same genre by a rival may motivate the publisher to request changes to the project. The market is very competitive and is driven by consumer trends. Publishers must respond faster to these market trends than the majority of the software industry.

### 3.4 Project Management

As with all industries, management is important. Poor management can negatively affect even the best of teams, whereas excellent management can make a mediocre team outstanding. As video games increase in complexity, the number of team members increases, and that places a strain on management capabilities. Good communication in a company is necessary for success. Sometimes, management's lack of information prevents them from identifying problem areas [7]. Companies need to invest in grooming good managers. Some managers have been promoted through the ranks, and are often lacking in managerial skills. Just as training is important for a development team, managerial training is also vital. Callele *et al* [3] points out that from their analysis of postmortems, a large percentage of internal problems were related to classic project management issues. Furthermore, internal issues were the predominant cause of project calamity.

Paul Miller [9] talked of best practices in light of Scrum methodology. His point was to emphasis that there are existing best practices learned through experience, and that managers should not ignore those regardless of the methodology. To generalize his conclusions about Scrum, software processes are simply tools in the project manager's toolbox. One must learn to use the proper tool for the job at hand.

The management of a game development project involves the oversight of multidisciplinary teams which include non-programming positions such as artists and musicians to name a few. These teams produce non-code assets that must be integrated into the game pipeline properly. These assets are very important to the success of a game. And programmers must create the software tools to integrate the assets.

### 3.5 Team Organization

Team organization varies from company to company. Teams are often segregated by specialty, such as a programming group and a design group. Groups may have sub-groups such as an AI team or a textures team. A common organization method is for each group to have a lead whom is an experienced employee in their area. Companies that have been implementing Agile methods have broken down the traditional groups to create functional units that are combinations of specialties. An example would be a unit composed of two programmers, a texture artist, and an animator. Combining groups does seem to enhance communication across disciplines. Bringing the diverse groups together can enhance understanding and communication between teams, and communication has been a failing point in many organizations.

Team organization does vary according to company culture. However, Petrillo *et al* do point out that the multidisciplinary teams often split between "artists" and "programmers" [10]. This division can have an effect on communication between the teams. Keith [8] stated that before his company adopted agile methodologies, the project workforce was split into groups of programmers, designers, and artists. Sharing of knowledge in the groups was a benefit, but caused increased overhead for communicating effectively between the teams. Based on the type of work involved, it does seem logical that the teams would be composed of members of similar skills. Unfortunately, indications point towards causing an "us versus them" mentality.

### 3.6 Development Process

The over-arching phases of game development are preproduction, production, and testing. Preproduction entails the conception of a game, and the GDD. By the end of preproduction, the GDD should be finished, though it will be continuously updated during the other phases. This is the phase where the game designers and developers do game prototyping in order to find the fun element of a game. If this can be accomplished, then production will go smoother. Actions in preproduction determine requirements and affect production.

Production is where the majority of assets are created, including code. This phase is the one most fraught with problems. A poor GDD affects project scope which affects production negatively. Feature creep happens at this stage, and can cause delays. A poorly managed production phase results in delays, missed milestones, errors, and defects. In production, the developers often create prototypes, iterations and/or increments of the game. Changes in prototypes or iterations of the game can cause drastic changes to the GDD. Unmanaged changes (or poorly managed ones) can cause widespread problems affecting functionality, scheduling, resources, and more.

The testing phase is usually the last before the game goes "gold" (handed off to the publisher for production and distribution). The testing phase involves stressing the game under play conditions. The testers, not only look for defects, but push the game to the limits (game options set to maximum resolution, textures, etc.). These phases are more involved than what is stated. The production phase does include planning and testing, and other activities that can be found the preproduction and testing phases. Looking at the phases in a broad view, they do seem to fit a waterfall model, though the activities in production break the model with the occurrence of iterations and increments.

The challenge that rises to the top is the translation of preproduction work to the production phase. The preproduction phase produces the GDD, and there are often problems translating this document into a project plan. The GDD contains stated requirements and unstated ones. The project manager must be prepared for such challenges.

Agile Methodologies: High Moon Studios adopted Agile methods sometime before November 2008 based on the article written by the chief technology officer, Clinton Keith [8]. One could assume that the company used the waterfall model, since they wanted to switch back to iterative and incremental development. The company adopted Scrum initially and saw improvements in productivity and morale. The company took an intelligent view on using Scrum. They would only modify a practice once they fully understood its use. Keith goes on to state that the programming team adopted Extreme Programming, and gained significant benefits. High Moon Studios has two major titles out which are indicative of their continued success. The question remains on whether they were on time and under budget.

Based on industry news from the site, Gamasutra.com, there are a few companies that use Agile or Agile-like methods. Crystal Dynamics whom made *Tomb Raider: Anniversary* employs Agile and Agile-like processes [5]. Even though details are lacking, the interview implies that the methods have benefited the company. Disney-owned Black Rock used aspects of Agile for projects that came out in 2006 and 2007, and fully embraced Agile for their latest, well-received *Pure* racing game [13]. Avent of Black Rock felt that it made the people happier and more productive. Again, details are scarce on what methods were implemented and how.

### 3.7 Third-Party Technology

Due to rising costs, increasing complexity, and higher consumer expectations, game developers are beginning to use more components from third parties [1]. There are software companies that create software engines (physics engines, game engines, AI, etc.) and components for gaming, but do not produce games of their own. The successful companies have solid software that aids the game developers in the creation of their games. Game developers can lower costs by using third-party technology. Two examples of games using third-party engines are *Prince of Persia 3D* and *BioShock*.

Red Orb Entertainment's *Prince of Persia 3D* employed several such technologies [1]. The developers chose Numerical Design's NetImmerse 3D for their game engine. This cuts down on development time, cost and resources. An added benefit is that the engine allowed for faster creation of prototypes [2]. For character animation, they used an animation package, Motion Factory's Motivate character animation environment. This enabled them to insert characters quickly into their 3D environments.

2K Boston/Australia's *BioShock* game made use of a couple of engines. Their game engine used a modified version of Epic's Unreal engine. They also made use of the Havok physics engine by Havok. By using these engines, they did not have to code solutions in-house.

There are issues to deal with when selecting third-party technology. Game engines are specific to a style of game (example, a flight simulator engine would not be a good choice for a role playing game) [2]. For the game, *BioShock*, the team chose the Unreal engine because the programmers had previous experience with it. The disadvantage was that the tool was slower for them to use [6]. Depending on the game engine, projects produced using it could be too similar to one another. Games using the old Doom engine looked and played like the game, *Doom* [2].

## 4. Learning from Software Engineering

Video game development process can improve and companies are beginning to recognize the need for change. Project managers need to look carefully at their current processes and identify what development process

works and what doesn't. They need to understand their own organization first. Without a good understanding, project managers cannot make the proper decisions in regards to a methodology that will work for their teams and organization. To use a cliché, there are no silver bullets to solve the problems. Even so, there are plenty of proven methods.

### 4.1 Process Improvement

For large projects, the spiral process model may work well. It can allow for iterating a game from low fidelity to high fidelity which is recommended by Miller [9]. Using this model, the project manager would develop a series of tasks derived from the GDD. The project manager should consult with all those involved to gather feedback on what should be developed and when. Requirements engineering is very important in transitioning the preproduction work (the GDD) to the production stage. The manager must clarify all requirements that the game designers have stated or implied in the GDD. By applying requirements engineering, the project manager can reduce errors due to miscommunication with the customers (the designers). Risk management is also an important element due to the common occurrence of feature creep. A certain measure of feature creep may be unavoidable, and can contribute to the success of a game, but there needs to be a measure of control in place. Late-addition features need to be carefully evaluated before inclusion as to how much value would the feature add.

If approached carefully, the project manager can plan the cycles of the spiral model so that the game is playable at the end of each cycle. This has several benefits. For one, testers can check the game for errors in a playable state which mimics what the end-user would encounter. Having a playable game as early as possible helps the team to see the potential of the end product, and the project manager could build in time for game exploration, especially if the project is large. With a playable game early on, and it being developed from low fidelity to high fidelity, the team can adapt to changes in the design a little faster, and can respond efficiently to a tightened deadline.

Scrum is similar to the spiral model in that it produces incremental iterations of the software. Its strength is to have the goal of having usable software at the end of every cycle. High Moon Studios adopted Scrum and XP [8], and found that it worked well for their teams. Clinton Keith of High Moon Studios stated that the teams sought to understand the methodologies first in implementation, before adapting them to fit their game development process. It is very important to understand these tools.

Another interesting approach would be to combine methods. Using lightweight agile methods during preproduction could be advantageous in exploring game playability and user interaction. In production, the team would use the more formal spiral process model which requires more structure, because the majority of experimentation has been completed. The point is that there is much experimentation and creativity during preproduction stages, and agile methods could work well and not stymie the creative spirit of the endeavor. Schofield, a proponent of XP in game development, beliefs that XP can help the designer to find engaging game play and fun features quicker [12]. This would certainly be beneficial in the preproduction stage where the game's design is laid out.

### 4.2 Project Scope

The problems with project scope often go back to the translation of the game design document to a project plan. Petrillo *et* al discovered that 75% of the projects reported problems with unreal or ambitious scopes [10]. Requirements engineering can help gather the stated and unstated requirements for the project. Involving the leads from the different teams (art, design, programming, etc) will help identify if the plan is realistic. Risk management helps the project manager understand changes to the plan and the potential cost in time and money. The project scope will never be a true reflection of the required effort due to the iterative and exploratory nature of game development, but it can be an effective guide in predicting success when discussing milestones, timelines, and budget.

### 4.3 Management and Team Organization

Good management is an issue in almost every industry, if not all. There are people whom are gifted managers and others that simply are not. But good management skills can be learned. An investment in training can help an organization improve the quality of their managers which is a key to success. Management should have the necessary skills to evaluate potential methods for their teams. To paraphrase Paul Miller [9], there is no value in practicing a method just for the method's sake. The manager must understand a method in order to understand its potential value.

People working in game development are from a variety of backgrounds and have different talents. These differences can create communication divides among the teams. In addition, there is a Machiavellian attitude among controlling seniors [10]. Those attitudes can only add to the poor communication already present. The best that management can do is to emphasize that everyone is important to the project. Video game projects are group projects. An individual or a small team may have developed the creative vision for the game, but it requires a strong effort from everyone. Part of this is a question of morale. Happy people do better work, and people that feel needed are often happier.

**Table 1: Challenges and SE Practices.**

| Challenge | Software Engineering Practices |
|---|---|
| Diverse Assets | Optimize tools and pipeline for integrating assets into the game. |

| | |
|---|---|
| Project Scope | Apply requirements engineering and risk management when translating the GDD to the project scope. Consult with the teams involved so that the project scope is realistic. Consider time needed for game exploration and feature creep. |
| Game Publishing | Develop deeper communications between the publisher and the development house. Publishers need to be clear with their requirements. Developers need to keep the publisher informed of project progress. |
| Project Management | Invest in managerial training with an emphasis on project management practices. |
| Team Organization | Evaluate potential process methods based on team organization and corporate culture. Encourage an attitude of the team as a whole and less importance on individuals. |
| Development Process | Understand current process and the problems with it. Identify processes that will benefit the project. |
| Third-Party Technology | Apply risk management to selection of third-party technology in order to identify which, if any, components would work best for the current project, and for future projects. |

## 5. Conclusion

Game development has unique characteristics that represent challenges to this industry. Applying SE principles and sound practices can help overcome these challenges. Development companies must invest in adopting proven methods, found in traditional SE, to fit with the peculiarities of game development such as the management of multimedia assets and the need for game play exploration. Many issues in video game development point to project management. Development companies need to invest in grooming skilled management through training (not only managing people, but teaching solid project management skills).

Agile methods could be applied to preproduction to allow for faster game exploration through the use of prototypes during this stage. The manager must apply requirements engineering to gather all requirements from the design and determine a feasible project scope through the consultation of teams involved. The project manager must also use risk management to handle feature creep and changing requirements. A more traditional process such as the spiral process model is recommended for the production stage, especially since most of the experimentation of game play should be completed during preproduction.

Game development projects generate large amounts of assets as players have come to expect greater amounts of content. The developers must optimize their tools and pipeline for the management of these assets. The teams, themselves, are composed of people with diverse backgrounds and skills. The teams need to be encouraged to work together, and to realize that all members are important for the success of the project. And as projects have grown larger, projects have required the usage of third-party technology. Components must be evaluated properly, best tool must be selected for the job, and previous experience should not be the lone deciding factor in the selection of third-party technology.

## 6. References


[1] J. Abouaf, "Adventure Game Tools Get Smarter-"Prince of Persia 3D"," *IEEE Computer Graphics and Applications*, vol. 19, no. 4, pp. 4-5, Jul/Aug, 1999.

[2] L. Bishop, D. Eberly, T. Whitted, M. Finch, M. Shantz, "Designing a PC Game Engine," *IEEE Computer Graphics and Applications*, vol. 18, no. 1, pp. 46-53, Jan., 1998.

[3] D. Callele, E. Neufeld, K. Schneider, "Requirements Engineering and the Creative Process in the Video Game Industry," *Requirements Engineering, IEEE International Conference on*, vol. 0, no. 0, pp. 240-252, 13th IEEE International Requirements Engineering Conference (RE'05), 2005.

[4] T. Demarchy, *Extreme Game Development: Right on Time, Every Time*, http://www.gamasutra.com/resource_guide/20030714/ demachy_01.shtml

[5] J. Dobson, *Industry News: Q&A: Crystal Dynamics' LaMer On 10 Years of Tomb Raiding*, http://www.gamasutra.com/php-bin/news_index.php?story=14172

[6] A. Finly, *Postmortem: 2K Boston/2K Australia's BioShock*, http://www.gamasutra.com/view/feature/3774/postmortem_2k_boston2k_.php

[7] D. Gregory, *Building a Mindset for Rapid Iteration Part 1: The Problem*, http://www.gamasutra.com/view/feature/3645/

[8] C. Keith,November 2006. Get in the Game: What others can learn from game developers. *Better Software Magazine*. http://www.agilegamedevelopment.com/articles_&_presentations.htm

[9] P. Miller, *Top 10 Pitfalls Using Scrum Methodology for Video Game Development*, http://www.gamasutra.com/view/feature/3724/

[10] F. Petrillo, M. Pimenta, F. Trindade, and C. Dietrich, "Houston, we have a problem..: a survey of actual problems in computer games development", *Proceedings of the 2008 ACM Symposium on Applied Computing*, Fortaleza, Ceara, Brazil, March 16 - 20, 2008, SAC '08. ACM, New York, NY, pp707-711.

[11] J. Rocca, H. Howie, S. Meretzky, J. Minton, K. Quirk, and T. Rosenthal-Newsom, "In the trenches: game developers and the quest for innovation", *Proceedings of the 2006 ACM SIGGRAPH Symposium on Videogames*, Boston, Massachusetts, July 30 - 31, 2006, Sandbox '06. ACM, New York, NY, pp9-11.

[12] B. Schofield, *Embracing Fun: Why Extreme Programming is Great for Game Development*, http://www.gamasutra.com/features/20070301/schofield_01.shtml

[13] B. Sheffied, L. Alexander, *Industry News: Q&A: Black Rock Tricks Out, Gets Agile With Pure*, http://www.gamasutra.com/php-bin/news_index.php?story=20336