“What Went Right and What Went Wrong”: An Analysis of 155 Postmortems from Game Development

Michael Washburn Jr.[1], Pavithra Sathiyanarayanan[1], Meiyappan Nagappan[1], Thomas Zimmermann[2], Christian Bird[2]
[1]Rochester Institute of Technology, Rochester, NY, USA
[2]Microsoft Research, Redmond, WA, USA
{mdw7326, ps2908}@rit.edu, mei@se.rit.edu, {tzimmer, cbird}@microsoft.com

ABSTRACT

In game development, software teams often conduct postmortems to reflect on what went well and what went wrong in a project. The postmortems are shared publicly on gaming sites or at developer conferences. In this paper, we present an analysis of 155 postmortems published on the gaming site Gamasutra.com. We identify characteristics of game development, link the characteristics to positive and negative experiences in the postmortems and distill a set of best practices and pitfalls for game development.

Keywords

Games, Postmortems, Qualitative analysis.

1. INTRODUCTION

Over the past thirty years, the importance and marketshare of video games in the world of software has grown by leaps and bounds. In lockstep with this growth, the scale of work required to develop games, whether in terms of budget, size of codebase, or team makeup, has ballooned and is on par with or exceeds any other software endeavors [13]. Games are arguably the most sophisticated and complex forms of software [18].

Indeed, games have been the driving factors behind many technological advances including high performance graphics cards, virtual reality, and distributed computing [16, 13]. Games also represent a substantial portion of software revenue; in 2013, video game revenue totaled over 93 billion dollars [21]! As such, the money, manpower, and effort put into video game development is likely to continue to increase in the coming years.

From a development perspective, games differ from more traditional software projects in a number of ways. Requirements are more subjective (e.g. “must be fun”), maintainability is often sacrificed for performance, testing and quality assurance are approached completely differently (e.g. live testers and few automated tests), most games require tools created from scratch, and deadlines are incredibly tight [12].

Therefore it is important to understand both the challenges that game development efforts face as well as the best practices that teams use to build games more effectively. The challenges are real problems faced by complex software efforts and represent avenues for research for our community. Successes and best practices embody knowledge that can aid future game development efforts and in some cases may generalize to or can be adapted for software development in non-game contexts. Because game development makes up a large slice of commercial software, a non-trivial proportion of students in computer science and software engineering programs will work on games during their careers. An understanding of game development can help educate and prepare such students.

Interestingly, game development has received very little attention in the academic community, as only three of the 116 open and closed source projects studied in the major software engineering conferences in two years were games [15]. Thus, one might reasonably expect that getting an inside view of game development is limited to a select few. Fortunately, the game development community has a unique practice that belies this assumption. Development teams often conduct postmortem retrospectives and share them publicly on gaming sites such as `Gamasutra.com` and at gaming conferences such as the Game Developers Conference (GDC). These postmortems offer an open and honest window into the development of games, often sharing the mistakes, setbacks, and wasted effort just as much as the successes and heroics that go into game building.

To address the limited study of this domain and shed light on the practice of game development we qualitatively and quantitatively analyze 155 retrospective postmortems published on `Gamasutra.com` over 16 years. These postmortems cover games for PCs, mobile devices, and consoles and range from small independent efforts to large AAA game franchises. As such, this represents the largest and most diverse study of game development to date and this data allows us to draw conclusions from a broad spectrum of game development. We make the following contributions in this paper.

- We present an empirically derived taxonomy of characteristics or dimensions of game development.
- We synthesize the best practices and commonly encountered challenges in game development and identify those areas that impact project outcomes the most.
- We provide recommendations for future game development based on the experiences shared in over one hundred postmortems.

ICSE '16 Companion, May 14-22, 2016, Austin, TX, USA
© 2016 ACM. ISBN 978-1-4503-4205-6/16/05…$15.00
DOI: http://dx.doi.org/10.1145/2889160.2889253

2. RELATED WORK

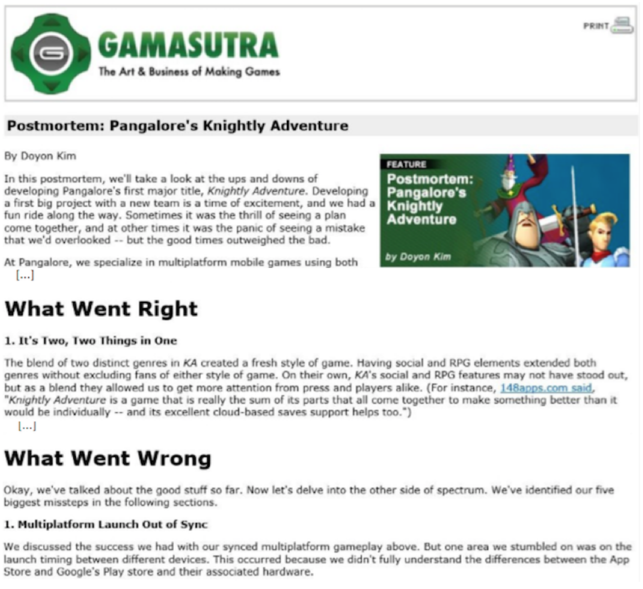
There are many books and articles which inspect the practices relating to the game development process [2, 5, 10, 11, 22, 3]. These studies primarily focus on how game development should be in the industry and are primarily based off the authors' experience. However, our findings focus on how game development is in practice based off of the postmortem reviews written by game developers and posted on Gamasutra.com [1]. There have been several studies where developers were interviewed or surveyed to understand several specific characteristics of game development [4, 8, 20, 7, 14, 12]. These studies tend to focus more on software engineering processes and methodologies in game development. Additionally, these studies solicited information from game developers by either conducting interviews or surveying developers. In contrast, we retrieved our data more organically by analyzing self reported postmortem reviews that had already been written by developers. Similar to what we do, there have been studies that examine what went wrong during game development using bug data [9], and 20 postmortem reviews [17]. However, in this study, we not only focus on what went wrong but also look at what went right in order to distill best practices and pitfalls.

The work most similar to ours is that of Ara Shirinian who conducted a study in which he analyzed 24 postmortem reviews written between 2008 and 2010 from Gamasutra.com in order to see if there were any interesting trends occurring in game development [19]. Also like this study, he analyzed the postmortems and grouped them into categories to determine what went right and what went wrong while developing games. In contrast, we analyzed a much larger set of postmortem reviews, spanning from 1998 until 2015. In addition to this, Shirinian categorized items from the postmortems into 7 categories of what went right and what went wrong, while we had a total of 22 categories in order to more precisely determine the trends in game development (which could be because of the increased set of postmortem reviews that we analyzed).

3. METHODOLOGY

To conduct this case study, we analyzed 215 postmortem reviews listed before Jan 2014 on Gamasutra.com, a game development news website. These reviews contain an introduction with some context, sections describing what went right and what went wrong during the development process, and finally some more contextual information in a table. An example of a postmortem is shown in Figure 1. We (Michael Washburn and Pavithra Sathiyanarayanan) analyzed each postmortem review, categorizing items discussed in the what went right and what went wrong sections into groups of common themes. We used the contextual information to determine which platform the game was designed for, how many people were in the development team, how much time did it take to develop the game among other things.

Ignoring Off-Topic Reviews: While analyzing the postmortem reviews, we found that some reviews were off-topic. For example, some cases were just reviews of an individual's experience at the Game Developers Conference. Sometimes the case only described a specific tool or technology used during development, rather than how development went. In other cases, authors focused on a particular aspect of their process, rather than their experience as a whole. Cases classified as off-topic were ignored in our study.


GAMASUTRA
The Art & Business of Making Games
PRINT

Postmortem: Pangalore's Knightly Adventure

By Doyon Kim

In this postmortem, we'll take a look at the ups and downs of developing Pangalore's first major title, *Knightly Adventure*. Developing a first big project with a new team is a time of excitement, and we had a fun ride along the way. Sometimes it was the thrill of seeing a plan come together, and at other times it was the panic of seeing a mistake that we'd overlooked -- but the good times outweighed the bad.

FEATURE
Postmortem: Pangalore's Knightly Adventure
by Doyon Kim

At Pangalore, we specialize in multiplatform mobile games using both
[...]

What Went Right

1. It's Two, Two Things in One

The blend of two distinct genres in *KA* created a fresh style of game. Having social and RPG elements extended both genres without excluding fans of either style of game. On their own, *KA*'s social and RPG features may not have stood out, but as a blend they allowed us to get more attention from press and players alike. (For instance, 148apps.com said, "*Knightly Adventure* is a game that is really the sum of its parts that all come together to make something better than it would be individually -- and its excellent cloud-based saves support helps too.")
[...]

What Went Wrong

Okay, we've talked about the good stuff so far. Now let's delve into the other side of spectrum. We've identified our five biggest missteps in the following sections.

1. Multiplatform Launch Out of Sync

We discussed the success we had with our synced multiplatform gameplay above. But one area we stumbled on was on the launch timing between different devices. This occurred because we didn't fully understand the differences between the App Store and Google's Play store and their associated hardware.

Figure 1: Screenshot of the first of a six page post mortem for *Pangalore's Knightly Adventure* on `Gamasutra.com`

If a case was in fact a postmortem review for a game, and detailed what went right and what went wrong during development, then it was used in this study. Out of the original 215 post-mortem reviews, 60 cases were ignored, leaving 155 postmortem reviews to be analyzed.

Identifying Categories: Initially, we started with 12 categories of common aspects of development. These categories were based on the categories Ara Shirinian identified in his analysis of postmortem reviews [19].

In order to identify additional categories, we performed 3 iterations of analysis and identification. The first week, we each read and analyzed 3 postmortem reviews each, classifying the items discussed in each section into the 12 predetermined categories of common aspects that impact development. While analyzing these reviews, we identified additional categories of items that went right or wrong during development, and revisited the reviews we had already analyzed to update the categorization of items. For the next two weeks we repeated this process of analyzing postmortems and identifying categories, analyzing 10 postmortems each in week 2, and 15 postmortems each in week 3. After each iteration, we discussed the additional categories we identified, and determined if they were viable.

Analysis: After our initial iterations for identifying additional categories, we had completed the analysis of 60 postmortem reviews. We then stopped identifying new categories, and began analyzing postmortems at a combined rate of about 40 postmortem reviews per week. After each week we reviewed what we had done to ensure we both had the same understanding of each category. This continued until we had analyzed all the postmortem reviews.

4. CONTEXTUAL INFORMATION

The characteristics of the games mentioned in the postmortem reviews we inspected were diverse in areas such as their supported platforms, the size of the development team, the amount of time it took to develop them, the type of publisher they had, and more. We will further describe the context of these games, based on data we gathered, which is available in our online appendix [6].

Platforms Used: The majority of the games created in these postmortem reviews were developed for the PC platform, with the next most common platforms being the Xbox 360, followed by the PlayStation 2 (PS2). Of the postmortem reviews we analyzed, 87% of them included the platforms they were developed for. Games can be developed for a single platform or for multiple platforms. Of the postmortems that indicated platform information, 37.86% of them were developed for a single platform, while 62.14% were developed for multiple platforms.

Length of Development: The majority of projects were developed in less then 2 years. Of the 73% of developers who put the length of development in their postmortem reviews, 39% of the games were developed in 1-2 years and 38% were developed in 1 year or less.

Team Size: Out of the 81% of postmortems which stated the number of developers the team had, 74% of the teams contained 20 team members or less. The remaining 26% of the teams were of various sizes greater than 20.

Publishers: Developers often have to make a choice between whether they want to, or can, publish their game themselves, or whether they should sign with a publisher. 90% of developers listed the publisher of their game in their postmortem. Out of these 74.4% used a third party publisher, while 25.6% decided to publish the game themselves.

5. IDENTIFIED CATEGORIES

We identified the following categories as common aspects of things that go right or wrong for teams during game development. Each of these categories are further divided into sub-categories. In each sub-category, we give a description of them and an example. Note that all the categories are present in both best practices and pitfalls. This is because a developer could talk about doing an activity like testing as a best practice, or talk about the lack of testing as a pitfall.

5.1 Product

In this category, we present and discuss seven sub-categories that are related to the artifacts that make the game.

(a) Art: Game art is a contributing factor to the overall quality of a game. Items that can go right or wrong within this category include artist skill, art development process, art quality, the effect of art on gameplay, and the effect of art on user experience.

Example of what went right: Nicole Goodfellow of Torus Games said that while they were developing *Scooby-Doo! First Frights* they made the decision to pair level designers with artists during development. According to Nicole, “They would sit together, eat together, live and breathe their level.” The entire team was impressed with the quality of the levels, and each pair took great pride in the levels they created.

Example of what went wrong: Søren Lund of Deadline Games said that they decided to focus less on creating unique art content and had their artists focus on small areas of the game instead when making *Chili Con Carnage*. The team recycled much of the game art from their previous title *Total Overdose*, under the assumption that few people would play both games. However, the reviewers had played both games, and gave the game bad reviews due to the lack of unique content.

(b) Creativity: The creativity of the development team is often a main factor in the success or failure of a game.

Example of what went right: According to Wim Coveliers, during the making of *American McGee’s Grimm* Spicy Horse did “a great job at keeping creative thinking alive throughout the whole project.” They were able to inject large amounts of creative content into their game by encouraging an open dialog within their company. They encouraged anyone with an idea come forward and lead a discussion. Decisions were made by a democratic vote, to prevent any one person from taking control of the game.

Example of what went wrong: Tim Turner, of developer Intergalactic Crime Prevention Unit, said that they chose to build an improvement of an existing game mechanic as their debut title, and while it was “not a knockoff,” it was not “groundbreaking” either. He stated that they “wasted many months on a title that, ultimately, we decided would not serve us well as our initial commercial release.”

(c) Features: Many developers listed features as something that went right when they had a lot of very unique and fun content in their games, or they had one feature that went very well and drastically contributed to the quality of their game. Developers typically listed features as going wrong when they poorly implemented a feature, discovered after implementation that a feature was a bad idea, or when they experienced feature creep.

Example of what went right: Lionhead Studios’ Peter Molyneux stated that while they were developing *Black & White* they were able to create an AI that was actually able to learn from the user and develop a personality such that no two users’ AIs would ever have the same personalities. Molyneux described the AI as an “astonishing piece of work.”

Example of what went wrong: John Cutler, of Cutler Creative, said that in *Last Call*, “Feature creep ruled our roost.” They kept adding features to their game even though they had a deadline and budget to meet, which resulted in tension between them and their publisher.

(d) Game Design: Developers listed this as something that went right or wrong when they had made good or bad design decisions that impacted the quality of their game.

Example of what went right: Stuart Denman of Surreal Software stated that “A good design will not only sell a game – it can also help smooth the development process.” During the making of *Sanitarium*, Surreal Software was able to create a game design that was unique, kept the team interested, offered many possibilities for new features, and allowed some freedom for artists and designed to work with, according to Denman.

Example of what went wrong: Søren Lund of Deadline Games wrote that during the making of *Chili Con Carnage*, they designed a game under the assumption that “players would ‘get’ the game and [the developers’] desire to make players perfect their score.” Upon release they discovered players weren’t interested in perfecting scores, but wanted more depth to the game. They stated that they should spent more time to lengthen the story of the game.

(e) Gameplay: How players interact with the game often determines how easily they will learn the controls as well as how entertaining it is for them.

Example of what went right: According to Prithvi Virasinghe and Jeremy Mahler of Pipeworks Software, while creating *The Deadliest Warrior* game (based on the popular TV show), they actually had some members of the team train with the weapons featured in the game. Additionally, they said “The show also weighed in on the warrior and weapons designs, as well as gameplay itself.”

Example of what went wrong: According to Bong Koo Shin of Gamevil, while developing *Nom 2*, they wanted to make the game very intense for the user in order to keep them interested in the game. However, many users complained that the gameplay was too intense, going as far as to say that "their eyes hurt because the game is so fast."

(f) Product Evolution: Software evolution in general indicates the changes happening in the software over time. In our study we link product evolution to those cases where the games evolve from one idea into another idea.

Example of what went right: Tom Leonard of Looking Glass Studios said that the *Thief: The Dark Project* game was originally called *The Dark Project*, but was renamed during development. Leonard stated that the renaming was "a seemingly minor decision that in truth gave the team a concrete ideological focus."

Example of what went wrong: Alyssa Finley of 2K Games stated that while developing *BioShock* "The spec of *BioShock* changed so much over the course of development that we spent the majority of the time making the wrong game." The game also changed from being a RPG first person shooter to being just a first person shooter. Finley said that "the real turning point for BioShock came when we had to present the game to the outside world, which forced us to carefully consider the story and takeaway message." She concluded by stating that they should have taken some time to develop those ideas sooner.

(g) Scope: The set of features that developers choose to include in their product is always going to make a difference in the quality of their product, and whether or not they will complete it on time.

Example of what went right: Brad Wardell of Stardock Entertainment stated that in *Galactic Civilizations*, they decided not to include a multiplayer mode. By limiting their scope they were able add more advanced features to single player mode. Wardell said "it's much easier to add features to a game when you don't have to worry about how they'll impact issues like synchronization, latency, and game flow." User feedback from beta testing also played a role in this decision. They concluded that making this decision made *Galactic Civilizations* a much better game.

Example of what went wrong: Schadenfreude Interactive was developing *Age Of Ornithology* and kept increasing the size of their scope. They stated that "Despite our attempts to cut back, the programmers were sneaking in Easter eggs until the last minute." This lack of control led to some player confusion after the game was released.

5.2 Development

In this category, we present six sub-categories that are related to the development process in building the game.

(a) Development Process: The process teams use while developing always affects the quality of the product. This was one of the more common categories that developers often listed in their postmortems.

Example of what went right: Jeferson Valadares and Mikko Kodisoja of Sumea Games stated that in Tower Bloxx, they needed to improve their game mechanics and design. Their solution was to pair their Senior Developer and Senior Designer together, and do small, rapid iterations. In these iterations the developer would make a change, then the designer would give them feedback and say how it should be fixed in the next iteration. Each iteration lasted less than 3 hours, and allowed a rapid evolution of the game. They said that this "allowed us [the developer] to get the core game mechanics to the point where we wanted [them] to be."

Example of what went wrong: Naked Sky Entertainment dove right into the development of *RoboBlitz* with almost no work done planning or designing the product. According to them, "When we started development on RoboBlitz, we jumped into the level creation process with very little pre-production." Unfortunately, this led to them designing their levels before they had fully decided what they wanted the game mechanics to be like. Additionally, when the game was nearly finished, the team had trouble making even minor design changes. They stated that "it became extremely burdensome whenever we had to add new features or tweak character movements."

(b) Documents: Generally, developers who listed something in the documentation category as going right when they produced designs and other documentation which benefited the project in the long run. When developers listed documentation as going wrong, they usually suffered from a lack of proper documentation.

Example of what went right: Wu Dong Hao of Ubisoft said that they dedicated 3 months to preparing for the development of *Tom Clancy's Splinter Cell.* During this time, one of the things they did was create technical design documents. The team stated that "One of the most important steps we took during the pre-production stage was to create Technical Design Documents (TDD), into which we poured all the knowledge we gleaned from our prototype." The documents also helped everyone to understand the major technical issues of their game. However, one team did mention that they spent as little time doing documentation as possible, and "As long as the documentation provided a high-level idea of the game and how it would play, we [the developers] dropped it like a hot potato and got back to work." They could do this because they they strongly emphasized verbal communication during development, and used a very iterative, agile approach to development.

Example of what went wrong: According to Marek and Ondrej Spanel of Bohemia Interactive Studio, in *Operation Flashpoint* they didn't document their design or have any documentation on what features had already been built during development. This led to problems in the late stages of development. At one point they said that "hours were spent trying to investigate how something had originally been meant to work." However, some teams also suffered from over-documentation. Paul Jobling of Eutechnyx stated that during the development of *Big Motha Truckers* they tried producing documentation that also doubled as an in-house marketing tool in order to get people interested in the game. Jobling stated that "As a result, instead of concentrating on the 'hows' and 'whys' of the game's production, it was instead focused on the 'whos' and the 'wheres.'"

(c) Obstacles: Teams often face obstacles during development. A situation will arise which makes it difficult for the team to continue development, and how the team reacts when faced with obstacles will impact their product for better or for worse. Obstacles are more likely to have a negative impact on a team.

Example of what went right: Alberto Moreno and Carlos Abril of Crocodile Entertainment stated that they didn't have proper office space for the development of *Zack Zero*, so they were forced to improve their process, communication, and task organization, which benefited them in the long run.

Example of what went wrong: John Li of Pixelogic stated that during the development of *The Italian Job* game, their team member Rob was in a car accident and was unable to work. After the injured team member was able to work, the team sent his PC to his home, so that he could work when he was feeling up to it. Overall, they said their schedule was delayed 2 weeks because of this incident.

(d) Team: Generally, developers had positive experiences with their teams. However, some teams do have trouble during development.

Example of what went right: During the development of *Dungeon Siege*, Bartosz Kijanka of Gas Powered Games described the development team as "the single best thing thing about *Dungeon Siege*." They attribute this to the individuals' strong values, personalities, and respect.

Example of what went wrong: Scott Bilas of Sierra Studios said that the team who developed *Gabriel Knight 3* suffered from bad role casting. The original team was not experienced enough to implement the advanced features of the game, and so there was a lot of turnover within the team. Bilas said "Many of the problems with GK3 resulted from developers being badly cast in their roles, usually because the project requirements were so severely underestimated."

(e) Testing: Developers put testing down under what went right when they performed some kind of testing during development such as unit/alpha/beta/usability testing, etc. When developers listed testing as something that went wrong, they generally suffered from a lack of testing.

Example of what went right: Paul Dennen of Nayantara Studios said that in *Star Chamber* they performed alpha and beta testing. This allowed them to make adjustments that they wouldn't have identified otherwise. Specifically, Dennen said "Long, meticulous early testing periods allowed players to acquire deep understanding of the gameplay, and they were then able to provide well-informed feedback that was invaluable in balancing the game."

Example of what went wrong: Ichiro Lambe, Dan Brainerd, and Leo Jaitley of Dejobaan Games stated that they didn't test *Aaaaa! – A Reckless Disregard for Gravity* enough. Additionally, even though they did eventually bring in beta testers, they said "when we did bring testers in to toss the game around, we underused the feedback we received."

(f) Tools: The tools you utilize to develop a game often impact the ease of development, the complexity of the features, and the quality of the end product.

Example of what went right: Eric Peterson, Wayne Harvey, Mike Pearson, and Dave Ellis of Vicious Cycle Software said that while developing *Dead Head Fred* they had previously developed their own tool, the Vicious Engine. They were able to implement all the game features for *Dead Head Fred* using preexisting code in the Vicious Engine, allowing them not to write any game specific engine code.

Example of what went wrong: Alexander Seropian of Wideload Games said that in *Stubbs the Zombie* they decided to use the Halo engine. However, the Halo engine had never been licensed before, and therefore had little documentation. Additionally, Seropian stated that "the learning curve slowed us [the developers] down and wasted time."

5.3 Resources

In this subsection, we present four things that developers talk about in their postmortems that are related to the resources required to develop a game.

(a) Budget: A team's budget can play a huge role during the development of any software. Often times in game development, the budget affects other aspects of the game such as marketing, which contractors get hired, and technology choices. Typically when developers listed the budget as something that went right, it wasn't because they had massive budgets, but rather they had average or limited budgets which they were able to stick to during development. Conversely, developers listed the budget as one of the things that went wrong when it was so limited that it hurt the quality of the game, or they were unable to stay on budget.

Example of what went right: Mike Goslin of VR Studios said that they were operating under a tight budget to minimize risk during the creation of *Toontown*. They were still able to release the quality MMORPG by designing it to be as low cost as possible. Goslin said "No in-game support is required, which eliminates a significant component of traditional customer service costs for this genre of game. In addition, we consume a fraction of the bandwidth of other MMORPGs. Both of these costs scale with the number of players, so they are important ones to minimize."

Example of what went wrong: During the development of *RoboBlitz*, Naked Sky Entertainment ran out of money midway through developing a game because they incorrectly estimated their schedule. Additionally they said "Because we weren't willing to put out an inferior product, we just kept going until we were satisfied with the quality." The developers were forced to borrow money from family members and close friends in order to continue development.

(b) Hardware: Developers listed hardware as something that went right or wrong when either they were able to work with very good hardware, allowing them access to advanced features, or they were working with very limited hardware, which hindered development. Developers were more likely to list this as something that went right or wrong if they were working on a single platform. In fact, about 90% of the developers who listed this as something that went right or wrong were working on a single platform.

Example of what went right: Motohideo Eshiro and Kuniomi Matsushita of Capcom described some of the Nintendo DS hardware features they benefited from during the development of *Okamiden*. In the PlayStation 2 and Wii prequels, *Okami*, users had some difficulty using the drawing feature of the game because of the limitations of the PlayStation controller and Wii Remote. Additionally, Eshiro and Matsushita stated that "with the Nintendo DS stylus, not only is one able to draw even more complex lines precisely, but we were able to create even better puzzles and missions that utilize the celestial brush mechanic."

Example of what went wrong: Bong Koo Shin of Gamevil described the hardware limitations they encountered on Korean mobile devices for *Nom 2*. At the time, it was very difficult for the devices to play two sounds at once. The only way they were able to get this to work was to cut the background music, then play the game sound, and resume the music. They were forced to cut the sound effects, and just stick with background music, which had a small negative effect on the experience of playing the game.

(c) Publisher Relations: If a developer chose to work with a publisher, then they often experienced benefits or drawbacks from that relationship. In general, those who listed items in this category in their postmortems had good relationships with their publisher.

Example of what went right: Linda Currie of Blue Fang Games said that the team had a great relationship with Microsoft during the development of *Zoo Tycoon 2: Marine Mania*. Currie said that "the time we [Blue Fang Games] spent with them [Microsoft] both in person and in [dialogging] via phone and email really paid off with both parties being on the same page about product decisions." Currie also expressed that they were able to convey ideas to Microsoft openly and freely, and concluded that "Throughout [*Zoo Tycoon 2: Marine Mania*] we felt like we were working with a helpful partner."

Example of what went wrong: David McQueen of The Games Kitchen described the development of *Wireless Pets* under Digital Bridges. While McQueen did say that "DB [Digital Bridges] is not at all bad to work with," there was some friction at times. Use of intellectual property, and differences of opinion of decisions about the game strained their relationship. They overcame this in the future by laying out very specific ground rules and processes to adhere to, and stated that they now work with Digital Bridges regularly.

(d) Schedule: Most developers who listed the schedule as something that went right did it because they were able to meet their milestones, or because they were granted extra time to complete their milestones. Conversely, those who listed the schedule as something that went wrong usually missed milestones or delivered them late.

Example of what went right: Wu Dong Hao of Ubisoft described their unique experience of having to port *Tom Clancy's Splinter Cell* from the Xbox to the PlayStation 2 within a 4 month schedule. Hao said they were able to make this work because "the company was willing to dedicate people and spend money to gain time." They also did a lot of preparation prior to when they were handed the Xbox version of the game, which helped to organize the team.

Example of what went wrong: Brian Reynolds of Big Huge Games described the developing of *Rise of Nations*. They underestimated their schedule, which made them work long hours later on in the project. Reynolds stated that"We underestimated the amount of coding time necessary, which resulted in an extremely overworked programming staff." Additionally, overloaded their lead programmer, making him a bottleneck for development. They also did not hire enough programmers to start. They stated that the root of their mistake was not listening to the postmortem reviews they had read in the *Game Developer* magazine.

5.4 Customer facing

Here we present four themes that are related to customer facing issues that developers discussed in their postmortems.

(a) Community Support: The support of a community can greatly benefit any game, whether it be an online community spreading the word about your game, or a group of outside individuals who lend a helping hand during development. Other aspects of community support that can go right or wrong for a team are the presence or lack of an online community, publicity by users of message boards, a loyal fan base. We found that developers listed this as something that went well during development rather than something that went wrong.

Example of what went right: Art Min of game developer Multitude said that they created tools for the community to utilize such within their game lobby and community website during the making of *Fireteam*. They stated that "Players are not only a source of revenue for a project, but they are a feature of your game." They argue that when players of a game develop a sense of community, it's like adding a whole other feature to that game.

Example of what went wrong: Jamie Cheng of Klei Entertainment said that they had a thriving online community when they released their first preview of their game *Eets: Hunger. It's emotional.* during development, which was merely a throwaway prototype of their game designed to gather feedback from users. However, the community was begging for more content and when Klei Entertainment could not deliver, the community died and their game lost popularity before it was even released. Cheng also said "A large part of the problem was we didn't know what to tell them. Were we going to be on handheld? Self-published online? Retail PC?"

(b) Feedback: In most cases developers listed feedback as something that went right when they actively sought feedback during development to make sure they were building the right product, or when they received good feedback after release. In general, developers listed feedback as something that went wrong when they received bad or insufficient feedback upon release.

Example of what went right: Alyssa Finley of 2K Games stated that while developing *BioShock*, 2K Games conducted tests with a focus group to see how they reacted to the game. They described the feedback as "brutal." Finley also said that "Based on this humbling feedback, we came to the realization that our own instincts were not serving us well," and they were able to make the necessary corrections.

Example of what went wrong: Ethan Einhorn of Other Ocean stated that after the release of *Super Monkey Ball 2*, the team realized that players were not interested in their game's multiplayer mode when all of the players' feedback was on the single-player mode. Einhorn stated "if we had known that interest in the multiplayer would be so limited, we may have dropped it." If they had sought feedback prior to release, they could have avoided this situation.

(c) Marketing: Typically, developers who published their own game listed marketing more than those who had publishers. About 70% of the developers that listed marketing as going right or wrong published their game themselves. This is because in most cases where a publisher is present, the developer won't deal with the marketing aspect.

Example of what went right: Frank Wilson and Josh Bear of Twisted Pixel Games described after developing *Splosion Man*, they were able to take their game to Microsoft's Summer of Arcade promotional event. Attending the event definitely helped get the word out about their game, they said. They stated that "The extra attention the game received was something that would have been very hard for a small company like ours to get recognition for on our own."

Example of what went wrong: Dave Gilbert of Wadjet Eye Games stated that after releasing *The Blackwell Convergence* on various game portals (online game distribution websites), the game was not getting noticed. Gilbert later said "I learned my lesson very quickly: I could no longer rely on the game portals to sustain the majority of my income," and that in the future he would have to do some marketing and PR work for his games.

(d) Piracy/Licensing: Piracy and licensing has always been an issue in the gaming industry. Whether it be a game losing sales because of piracy, or a lack of quality music because the developer couldn't get the license.

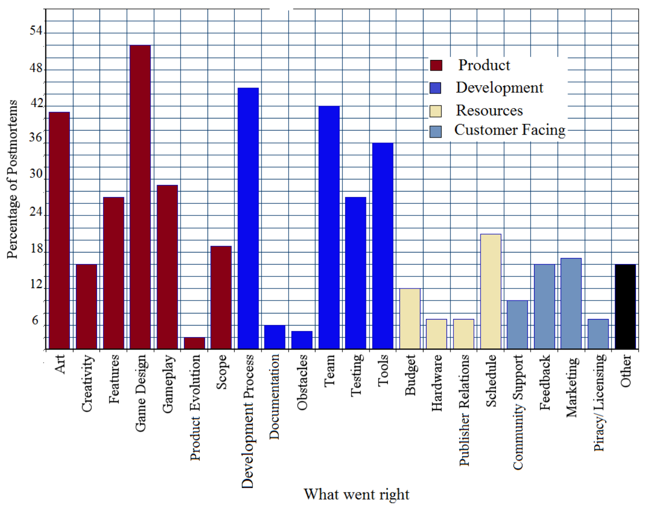


Figure 2: What went right

Example of what went right: Blair Fraser and Brad Wardell of developer Ironclad said that they chose to release *Sins of a Solar Empire* without any CD/DVD copy protection. They calculated that users would still buy the game to avoid the inconvenience of pirating it. Therefore, they were able to ignore the issue, and save a little time and money.

Example of what went wrong: Alex Austin of Chronic Logic stated that when they released *Gish*, many players started pirating their software after release. This took away from their profits and they expressed much frustration at watching so many users steal their game. They even said that in some cases there were "people even writing to us [Chronic Logic] for tech support when their stolen keycode [didn't] work with the newest patch."

5.5 Other

We categorized things into this category when they didn't quite fit into the other categories. Examples include - teams believing in themselves, a team self funding their own project, luck, and risks paying off.

6. RESULTS

6.1 Best Practices

Figure 2 shows the percentage of postmortem reviews which list each category as something that went right during development. These percentages do not add up to 100% because each postmortem lists multiple categories that went right during development. The four categories that most frequently went right were game design, development process, team, and art.

(a) Game Design: As shown in Figure 2, Game Design was stated to have went right in 50% of postmortems. In general, teams who marked game design as something that went right emphasized having a "hook" to capture player interest, and having a clear vision. Having a complete story, and developed characters were less prevalent themes.

Example of game hook: Dan Marshall stated that when making *Gibbage*, he emphasized variety in the different levels of *Gibbage*, which helped maintain the player's attention. Marshall said "It's this variety that keeps the gameplay fresh throughout each of its levels."

Example of clear vision: Paul Dennen of Nayantara Studios stated that "Focusing the design down one clearly plotted path from start to finish resulted in a game with strength of clarity and identity" during the creation of *Start Chamber*.

Takeaway: Game developers should create a well-defined concept before beginning development as opposed to an ad hoc method of game design. Developers should also focus the design around capturing the attention of the player.

(b) Development Process: Development process was marked as something that went right in 43% of the postmortem reviews, according to Figure 2. Among the teams that listed the development process as something that went right when making a game, there were 3 common practices teams did: First, they spent time planning and preparing for the game prior to development. Additionally, they often created prototypes which they would use as a proof of concept for the game in general, or a specific feature of the game. Many teams also used their prototypes as a basis of development. It was also common for these teams to use an iterative development process.

Example of planning: Brian Gilman of Full Sail described how before starting development on *Romeo and Juliet*, they were able to spend an extra month planning how they were going to complete the project. They said that "a single additional month in pre-production can make all of the difference in the world."

Example of prototyping: Wu Dong Hao of Ubisoft stated that they made a prototype for their title *Tom Clancy's Splinter Cell* which "helped determine resources and scheduling" for the whole project. They also said that the prototype helped them organize the production schedule and proved that they could overcome the technical issues of the problem. Additionally, they were able to use the prototype as a base for production.

Takeaway: For a better development process, game developers should invest time in the beginning of the project planning and designing. Game developers should also build prototypes during development, and if possible continue building off of these prototypes using an iterative process.

(c) Team: The team was listed as something that went right in 40% of the postmortem reviews that we analyzed, as depicted in Figure 2. Of the developers that listed the team as going right during development, many of them credited their experienced team members. Many developers also stated that their team members were very motivated.

Example of experienced team: According to Rade Stojsavljevic, the team who built *Command and Conquer: Tiberian Sun* was extremely experienced with real time strategy games. Many of the members had worked on at least 4 real time strategy games before, and according to Stojsavljevic "This level of experience was key in allowing the team to conquer all the obstacles thrown in their path."

Example of motivated team: Deng Yi Wen of Ubisoft described how the team of *Music Up – Summer Rainbow* was highly motivated. He stated that "A willing crew is the most important factor shipping a title on time, and we had a team who [was] highly motivated by the chance to finally make a game for local players"

Takeaway: Game developers should spend more time training their team members in the environment they will be developing in. However, game development companies should focus on recruiting not only talented individuals, but they have to be motivated as well.

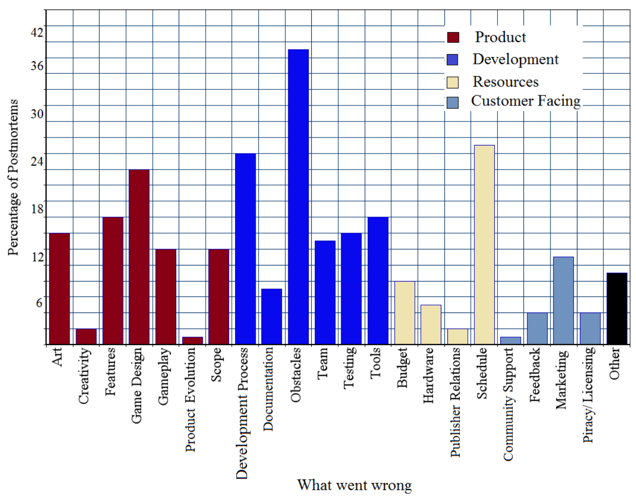


Figure 3: What went wrong

(d) Art: In Figure 2 it is shown that art went right in 39% of the postmortem reviews. Those developers who marked art as something that went right in development typically had creative artists on their teams, or were able to contract creative artists.

Example of creative artists: Brad Wardell of Stardock Entertainment described their method of improving the quality of art in *Galactic Civilizations 2: Dread Lords*. According to Wardell, "Anyone who's followed Gamasutra for a long time and seen Stardock's games mentioned in the past knows one simple fact: Stardock's games look like crap." To improve the art in the new *Galactic Civilizations*, they began working with popular artists like Paul Boyer and Mike Bryant. Additionally, they increased the size of their in-staff art department. Wardell stated that "The result was that we were able to make a game that had competitive graphics for any AAA strategy title."

Takeaway: In order to build a higher quality game, game studies should seek the help of professional artists in order to make visuals that will capture the attention of the player.

6.2 Pitfalls

The percentages of each categories that went wrong in the postmortem reviews can be seen in Figure 3. These percentages do not add up to 100% because each postmortem mentions multiple categories that went wrong during development. The 4 most occurring categories that went wrong for teams during development are obstacles, schedule, development process, and game design.

(a) Obstacles: Teams reported facing various obstacles in 37% of the postmortem reviews, as shown in Figure 3. While there was a large variety in the types of obstacles faced by developers, a significant number of the teams and sometimes companies were newly formed, and faced obstacles related to lack of team dynamic and unfamiliarity among the team.

Example of newly formed team: Scott Alden of DreamForge Entertainment described the troubles in their newly formed team during the development of *Sin*. Alden said "Our newly formed tribe felt little sense of cohesion, as most members were basically strangers to each other." He went on to say that their team had trouble agreeing with each other, which led to problems making decisions during the project.

Takeaway: Developers working in newly formed teams should participate in team building in order to minimize risks during development related to unfamiliar teams. Additionally, new companies and teams should subscribe to a method of risk management, because they are more likely to face obstacles than more seasoned teams.

(b) Schedule: In 25% of the postmortem reviews, developers said that the schedule was something that went wrong, as show in Figure 3. These games faced common problems in estimation and optimistic scheduling. Some teams also faced problems with design changes late in development.

Example of bad estimation: Linda Currie of Blue Fang Games described the scheduling problems surrounding their game *Zoo Tycoon: Marine Mania*. According to Currie, "we [Blue Fang Games] overlooked the fact that the young guests needed their own specific animations." Currie also described additional scheduling problems, saying that "We also underestimated some of the difficulty in transitioning to 3D movement through water."

Example of bad optimistic scheduling: Peter Molyneux of Lionhead Studios described the scheduling difficulties they encountered during the creation of *Black & White*. Molyneaux admitted "I have a reputation for being, shall we say, optimistic about when the projects I'm working on will be completed." Molyneaux also stated that there were many people who didn't believe him when he finally told them the game was finished, and that they "were only convinced when they saw a box with a CD in it."

Takeaway: To avoid schedule slippage, game developers need to spend more time to plan out all the work that needs to be done so that no tasks are overlooked when giving estimates. By planning out tasks, you can also estimate each task individually, and give a more accurate prediction, instead of an optimistic one.

(c) Development Process: The development process was listed as something that went wrong in 24% of postmortem reviews. Many teams listed this as something that went wrong when they did not spend a significant amount of time planning before beginning development. There was a large variety of things that went wrong in the development processes, however, another less common theme within these postmortems was mismanagement.

Example of lack of planning: Wayne Imlach of Muckyfoot Productions described a lack of up-front design work during the making of *Startopia*. The result was a lot of confusion among the team during development. Imlach said that they dove in to development as soon as possible but that "we [Muckyfoot Productions] failed to realize at the time that everybody was carrying a slightly different picture in his head of what the final game would be."

Example of poor project management: Leonard Paul of Moderngroove stated that *Modern Groove – The Ministry of Sound Edition* suffered from poor project management because they had no dedicated project manager. Most of those responsibilities were given to the lead programmer, which resulted in an over burdened team. Paul said "I believe hiring a good manager early in the project and having clearer deadlines would have definitely helped the project."

Takeaway: In order to avoid conflicts during the development process, teams need to have proper management. Developers also need to invest time upfront planning before beginning development.

(d) Game Design: According to Figure 3, game design is something that went wrong in 22% of the postmortem reviews. Many teams fell victim to overly ambitious game designs which could not be implemented, or caused them to go over their planned schedule. In addition to this, many other teams designed games with concepts that confused the player.

Example of ambitious design Jim Napier of Sierra Studios described how the design for *SWAT 3: Close Quarters Battle* was extremely ambitious. Specifically, Napier said that "Most of the individual features seemed doable, but added up they represented a tremendous amount of work." They were able to release the game on time still by cutting out some of the features, which "was painful for everyone because we felt the game wouldn't be nearly as fun without the missing features."

Example of confusing design Bruce Chia and Desmond Wong of Singapore-MIT GAMBIT Game Lab described the confusion surrounding their original prototypes for a game. The game was based around the concept that players would be rewarded by failing to complete an action. However, they said that "testers didn't understand the logic behind these design decisions, and we spent a lot of time making a game that was not well thought-out and was boring to play."

Takeaway: For a better game design, developers should keep implementation in mind while creating a design, and if a implementation cannot be pictured for a part of the design, contingencies should be made. Key game concepts should also be testing before release so the reactions of the players can be verified.

7. DISCUSSION

As described in Section 4, we collected the metadata available in each postmortem as well. Based on this metadata, we split our postmortems based on team size, the presence or absence of publisher, and single or multiplatform, to analyze our results further.

7.1 Small Team vs. Big Team

When looking at what went right and wrong in teams of 20 or less team members, and teams of more than 20 team members separately, we can see some clear differences between the two sets.

Firstly, 61% of teams with 20 or less members listed game design as something that went right during development, while only 50% of teams with more than 20 members listed this as something that went right during development. Conversely, 25% of both small and large teams listed the team as having gone wrong during development. This may indicate that smaller teams also produce a better game design.

When looking at gameplay and different team sizes, we found that 42% of large teams listed gameplay as something that went right during development, while only 27% of small teams listed gameplay as having gone right. Supporting this, 16% of small teams listed gameplay as something that went wrong, and only 9% of large teams listed this as something that went wrong. This indicates that having a larger team may produce a game with better gameplay.

Lastly, we found that 50% of small teams listed obstacles as something that went wrong, while only 26% of large teams listed obstacles as something that went wrong during development. This indicates that small teams may be much more likely to encounter obstacles during development than large teams.

7.2 Publisher vs. No Publisher

When looking at what went right and wrong in developers who had publishers versus developers who published their own game, we identified a few categories in which one group performed better than the other.

First, 30% of the developers without publishers listed testing as something that went right during development, while only 19% of those with publishers listed this as having gone right. The two groups were comparable in the number of times they listed testing as something that went wrong, with 11% of those without publishers and 13% of those with publishers listing it as having gone wrong. This shows that those developers who published their own games performed better testing than the developers who had publishers.

Second, we found that while only 17% of developers who published their ow game listed tools as something that went right, 40% of developers with publishers listed tools as something that went right. Both groups listed tools as having gone wrong in only 17% of cases. This indicates that developers who have publishers often have better tools for development than developers who do not have publishers.

Lastly, we found that 43% of the developers that published their own game listed obstacles as something that went wrong, while only 35% of the developers with publishers ran into obstacles. This suggests that developers who publish their own games may be affected more severely by obstacles than developers who utilize publishers.

7.3 Single Platform vs. Multiplatform

When looking at games developed for one platform versus games developed for multiple platforms, we found a few advantages and disadvantages to choosing one over the other. First, 46% of developers that used multiple platforms listed art as something that went right, while only 33% of developers that used a single platform listed this. Additionally, 15% of the developers that used a single platform listed this as something that went wrong, while only 17% of developers that used multiple platforms listed art as having gone wrong. Therefore, the evidence shows us that games developed for multiple platforms may be more likely to have better game art.

We also found that 20% of developers that worked on multiple platforms listed marketing as having gone right, and only 12% of developers that worked on a single platform listed this as having gone right. There is not much variation between the two groups when looking at marketing as something that went wrong. In fact, 11% of the developers that used multiple platforms and 10% of the developers that used single platforms listed marketing as going wrong. This evidence suggests that marketing is often better in games developed for multiple platforms.

8. THREATS TO VALIDITY

The three threats to validity of this research are: (1) We could have made mistakes in qualitatively analyzing the postmortems. We tried to address this issue by having two authors discuss their qualitative tagging frequently. (2) The results were synthesized from self reported postmortems, which could be written by one particular set of game developers. However, we find that the games we analyzed were very diverse. (3) The authors of these postmortems may not report what actually happened during development or hide certain failures. However, on reading these postmortems, we often find that the authors are very candid. Also in order to make our experiments more transparent and reproducible, we have made available all the data, including the actual raw postmortem reports, in our online appendix [6].

9. CONCLUSIONS

We find that we were able to identify both best practices and pitfalls in game development using the information present in the postmortems. Such information on the development of all kinds of software would be highly useful too. Therefore we urge the research community to provide a forum where postmortems on general software development can be presented, and practitioners to report their retrospective thoughts in a postmortem.

Finally, based on our analysis of the data we collected, we make a few recommendations to game developers. First, be sure to practice good risk management techniques. This will help avoid some of the adverse effects of obstacles that you may encounter during development. Second, prescribe to an iterative development process, and utilize prototypes as a method of proving features and concepts before committing them to your design. Third, don't be overly ambitious in your design. Be reasonable, and take into account your schedule and budget before adding something to your design. Building off of that, don't be overly optimistic with your scheduling. If you make an estimate that initially feels optimistic to you, don't give that estimate to your stakeholders. Revisit and reassess your design to form a better estimation.

10. REFERENCES

[1] Gamasutra. http://www.gamasutra.com/. 2015-09-30.

[2] Erik Bethke. *Game development and production.* Wordware Publishing, 2003.

[3] Jonathan Blow. Game development: Harder than you think. *Queue*, 1(10):28–37, February 2004.

[4] Thierry Burger-Helmchen and Patrick Cohendet. User communities and social software in the video game industry. *Long Range Planning*, 44(5–6):317 – 343, 2011. Social Software: Strategy, Technology, and Community.

[5] Heather Maxwell Chandler. *The Game Production Handbook.* Jones & Bartlett Learning, 2008.

[6] Michael Washburn Jr., Pavithra Sathiyanarayanan, Meiyappan Nagappan, Thomas Zimmermann, and Christian Bird. Appendix to "what went right and what went wrong": An analysis of 155 postmortems from game development. Technical Report MSR-TR-2016-6, February 2016. research.microsoft.com/apps/pubs/?id=262289.

[7] Jussi Kasurinen, Jukka-Pekka Strandén, and Kari Smolander. What do game developers expect from development and design tools? In *Proceedings of the 17th International Conference on Evaluation and Assessment in Software Engineering*, pages 36–41. ACM, 2013.

[8] Annakaisa Kultima and Kati Alha. Hopefully everything i'm doing has to do with innovation: Games industry professionals on innovation in 2009. In *Games Innovations Conference (ICE-GIC), 2010 International IEEE Consumer Electronics Society's*, pages 1–8, Dec 2010.

[9] Chris Lewis, Jim Whitehead, and Noah Wardrip-Fruin. What went wrong: a taxonomy of video game bugs. In *Proceedings of the fifth international conference on the foundations of digital games*, pages 108–115. ACM, 2010.

[10] Morgan McGuire and Odest Chadwicke Jenkins. *Creating games: Mechanics, content, and technology.* CRC Press, 2008.

[11] Mike McShaffry and David Graham. *Game Coding Complete.* 4 edition.

[12] Emerson Murphy-Hill, Thomas Zimmermann, and Nachiappan Nagappan. Cowboys, ankle sprains, and keepers of quality: How is video game development different from software development? In *Proceedings of the 36th International Conference on Software Engineering*, ICSE 2014, pages 1–11, 2014.

[13] Juergen Musil, Angelika Schweda, Dietmar Winkler, and Stefan Biffl. Improving video game development: Facilitating heterogeneous team collaboration through flexible software processes. In *Systems, Software and Services Process Improvement*, pages 83–94. Springer, 2010.

[14] Juergen Musil, Angelika Schweda, Dietmar Winkler, and Stefan Biffl. A survey on the state of the practice in video game software development. Technical report, Technical report, QSE-IFS-10/04, TU Wien, 2010.

[15] Meiyappan Nagappan, Thomas Zimmermann, and Christian Bird. Diversity in software engineering research. In *Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering*, ESEC/FSE 2013, pages 466–476, 2013.

[16] Randy J. Pagulayan, Kevin Keeker, Dennis Wixon, Ramon L. Romero, and Thomas Fuller. The human-computer interaction handbook. chapter User-centered Design in Games, pages 883–906. L. Erlbaum Associates Inc., Hillsdale, NJ, USA, 2003.

[17] Fábio Petrillo, Marcelo Pimenta, Francisco Trindade, and Carlos Dietrich. What went wrong&#63; a survey of problems in game development. *Comput. Entertain.*, 7(1):13:1–13:22, February 2009.

[18] Robert Purchese. Games are arguably the most sophisticated and complex forms of software out there these days. http://bit.ly/eurogamer-games-more-complex. 2015-10-22.

[19] Ara Shirinian. Dissecting the postmortem. http://gamasutra.com/view/feature/134679/ dissecting_the_postmortem_lessons_.php. 2015-09-30.

[20] Patrick Stacey and Joe Nandhakumar. A temporal perspective of the computer game development process. *Information Systems Journal*, 19(5):479–497, 2009.

[21] Rob van der Meulen and Janessa Rivera. Gartner Says Worldwide Video Game Market to Total $93 Billion in 2013. http://www.gartner.com/newsroom/id/2614915. 2015-10-22.

[22] Michael Thornton Wyman. *Making Great Games: An Insider's Guide to designing and developing the World's Greatest Games.* CRC Press, 2012.