Analyzing Black-box Testing from Classic Game Development Post-mortems

Miguel Garcia-Ruiz, Simon Xu, John Console
Department of Mathematics and Computer Science
Algoma University
Sault Ste. Marie, Canada

Abstract—Testing is a very important activity in game development projects because it uncovers game user interface, gameplay and programming problems, including game “bugs” (flaws or errors in the game software producing an unexpected or incorrect result). Post-mortems (structured documents describing what went well and wrong in game development projects) are widely read in the gaming industry, covering testing and other game development aspects. This paper summarizes game testing described in post-mortems published by the specialized game development magazine gamasutra.com, analyzing black-box testing (testers analyze game inputs and outputs). In our post-mortem analysis methodology, we selected and classified testing aspects from 10 “classic” post-mortems, conducting both qualitative and quantitative data analysis. We found best testing practices described in the post-mortems and useful advice for future gaming development projects.

Keywords—Black-box Testing; Video Game; development; post-mortems; best practices

I. INTRODUCTION

Game testing is an essential game development activity, and is of paramount importance because it allows developers to uncover game design, user interface (UI) and player interaction problems [1] and “bugs” (game glitches or software errors that produce an undesirable and incorrect result when the game is played). User testing methods are run to see if a video game can be played with ease, and whether it is satisfactory and fun to play [1]. A game that contains glitches, bad mechanics and user interfaces may yield to a poor gaming experience such as player’s frustration and anxiety, accepting the acceptance and long-term usage of the video game [2,3]. These problems will also affect sales of commercial video games. In this context, the global video game industry is huge and is growing every year. Worldwide sales of games is billions of U.S. Dollars, and is increasing every year [3]. This makes pressure to game developers to make games faster and with more features, compromising games quality [1]. However, there are no specific and standard methods for testing games while they are developed, and not all the testing methods from software engineering are suitable or directly applicable to video game testing. Obtaining, analyzing and testing requirements for developing games are special and highly subjective (e.g. analyzing how fun a game is), and it has been argued that video games are a very complex type of software. This paper will address game testing issues described in post-mortems. A post-mortem is a summary of a past game development project that mainly describes what went right and what went wrong (or could have done better) in the project, written by game developers [4].

This paper shows a systematic analysis and review of game development post-mortems focusing on how software and user testing were carried out in game development projects.

The objective of this paper is to highlight game testing problems and testing best practices described in post-mortems, summarizing useful advice for future gaming development projects.

II. POST-MORTEMS IN GAME DEVELOPMENT PROJECTS

Software engineering projects such as game development projects involve processing a large quantity of information distributed over different knowledge domains such as project management, resource management, software requirement, software quality, software configuration etc. Since software engineering projects often last for years and need many resources, the project success is very important.

Post-mortems commonly refer to an improvement-oriented analysis by the project team, which may include various users, customers, and stakeholders, to evaluate the past experience and to develop “lessons learned” for the benefit for future projects [4,5].

Game development post-mortems are analyzed in the video game industry. A game post-mortem is a structured document (they have been published in the form of book chapters, articles and web pages) written by a game development team or individual that describes important issues and a retrospective analysis of a finished (or cancelled!) game development project, highlighting what went right and what went wrong during its process [6]. One objective of post-mortems is to avoid past projects’ mistakes in current and future game development projects, as well as learning best practices from them.

III. TESTING IN VIDEO GAME DEVELOPMENT PROJECTS

Game testing can be defined as an investigation and activities performed by game development team members, testers, potential players and other project stakeholders to check whether the actual results of a developed game match the expected results and to identify errors and missing design requirements [2]. In the software engineering context, we analyze two main branches of testing: black-box testing, where the video game is seen as a black box and people only revise and review its inputs and outputs [1]. An example of black-box
testing is quality assurance (QA) testing, where testers look for bugs (game glitches) and report them. Another type of black-box testing is user testing (e.g., usability testing and playtesting), which strives to obtain valuable user feedback that will serve to fix eventual problems from the game’s user interface, mechanics, and gameplay [3]. There are no standard methods for conducting game testing, hence the importance of analyzing them from post-mortems. This paper analyzes from the post-mortems four types of testing that should happen during a typical lifecycle of game development at different stages.

In this paper, we are analyzing black-box testing issues used to obtain valuable feedback to improve the quality and player experience of video games.

IV. RELATED WORK

Washburn et al. [7] conducted an extensive analysis of 155 post-mortems published in the e-magazine Gamasutra.com, categorizing aspects such as product, development, resources and customer support from the post-mortems, and summarizing findings on what went right and what went wrong in each of them and their sub-categories. Testing was briefly analyzed in their paper as a sub-category of “Development”, and software testing was briefly mentioned in other categories. Interestingly, some of the analyzed post-mortems reported a lack of testing in the game development projects as some aspect of what went wrong in those projects. Grossman’s book [6] compiled 25 post-mortems of popular video games (when the book was published), showing that testing was a common and important activity in all of them. Some of the post-mortems point out that the time and resources devoted to testing was inadequate, and the project managers wished to have more time allocated for that. McAvoy’s paper [8] analyzed and discussed attitudes and beliefs that project members have towards their own game post-mortems. In it, McAvoy found that in a post-mortem from a game called “The Pygmalion Effect”, project testers’ expectations affect how the rest of the team view software testing, including overly positive beliefs about the game code that may hinder quality assurance testing. This confirms a common effect in software engineering where software testers cannot identify some bugs in their own software. Tschang [9] reviewed a number of post-mortems published in the magazine Game Developer, where found that both positive and negative development features were often described in them, concluding that testing is a very critical activity in video game development projects.

At the time of writing this paper, to our knowledge, there is no academic publication focusing on testing analysis conducted on game post-mortems.

V. METHODOLOGY

To analyze black-box testing aspects from posted post-mortems, we selected and classified testing types from them and conducted qualitative analysis. We then compared and summarized the testing-related information from the post-mortems.

A. Post-mortem Selection

We selected ten “classic” post-mortems that describe development projects of popular games (they are important and unmissable post-mortems according to [10] and written by accomplished game developers), published in the online magazine gamasutra.com. Gamasutra is one of the main sources of game post-mortems, and is widely read in the gaming industry. We selected the post-mortems listed by [10] to compare testing issues and to obtain best testing practices from them. We analyzed the development projects of the following games, described in those post-mortems:

- Ion Storm’s Deus Ex
- Irrational Games’ System Shock 2
- Blizzard’s Diablo II
- Looking Glass Studios’ Thief: The Dark Project
- BioWare’s Baldur’s Gate II
- Bungie’s Myth: The Fallen Lords
- Epic Games’ Unreal Tournament
- Ensemble Studios’ Age of Empires II: The Age of Kings
- Red Storm’s Rainbow Six
- Lionhead Studios’ Black & White Postmortem

B. Category Identification

We defined seven categories for classifying testing aspects and issues in the analyzed post-mortems, namely: prototyping and testing, QA and beta testing, playtesting, and miscellaneous. This category classification was based on the structured categories analysis done by [7]. We also added the following categories: what went right, what went wrong, and best practices. We corroborated and expanded the list of categories by browsing the selected post-mortems in two iterations.

C. Post-mortem Qualitative Analysis

After we selected the post-mortems and identified the categories, we conducted a qualitative analysis of the selected post-mortems. We wrote a document with phrases taken from the post-mortems, about testing information and issues (e.g., comments that project members made about their QA testing) for future data analysis. We conducted word searches on the post-mortems using keywords such as “test”, “testing”, “tester”, “evaluation”, “evaluate”, “QA”, “quality assurance”, “asses”, “requirements analysis”, “gameplay”, “playtesting”, “tried”, “played”, and “feedback”. This greatly facilitated the post-mortem analysis. Then, we compiled the findings from the analyzed post-mortems.

We analyzed the selected post-mortems by following Miles and Huberman’s [11] stages for qualitative data analysis, as follows:
(i) preparation (review and organization)
(ii) classification (data reduction and coding levels)
(iii) interpretation (data display and pattern identification)
(iv) conclusion drawing (verification and presentation).

D. Post-mortem Quantitative Analysis

At the end of the post-mortem revisions, we operationalized the testing comments by calculating the number of comments from each of the category, and counting the number of projects that conducted QA, automated testing, playtesting and software testing.

VI. RESULTS AND DISCUSSION

The development time of the projects described in the post-mortems were 2.2 years, where the shortest one was 1.5 and the longest one was just above 3 years, with a standard deviation of 0.5, and most of them employed about 10 to 30 people. We deemed the 10 projects as large. Nevertheless, we believe that the testing issues analyzed and the best practices obtained from those post-mortems could apply to smaller projects, because most of them are in line with testing recommendations and best practices published in software and game testing publications such as [1,12,13].

A. What Went Well and what Went Wrong?

Table 1 shows a summary of what went well and what went bad from the game development projects analyzed in the post-mortems, written by their game developers. We marked the testing issues in black to put them into context. More testing issues are also described in the rest of the post-mortems.

Table 1. Summary of what went well and what went bad described in each post-mortem.

It seems that, according to what went well and bad in the analyzed post-mortems, game development projects must take into account black-box testing seriously, with a testing plan before the game development starts. In addition, the project should include gameplay, usability, QA and software testing and conduct them in a timely manner.

B. Best Practices

We inferred the following best practices from the ten analyzed post-mortems:

Prototyping and Testing:
- Test game prototypes early and often. If this does not happen, there will not be enough time at the end of the project to conduct proper testing. Especially, conduct testing of a game level prototype as early as possible, when it is still easy to modify. Prototype testing is good for seeing potential in them.
- Do not underestimate wireframe prototypes. They can work as a useful test bed and provide valuable feedback in early development stages.
- In addition to wireframe prototypes, test the AI from a prototype as soon as it is implemented in the game. AI can be tricky to code and fix and this can take considerable time in a development project.

QA and Beta Testing:
- Conduct QA testing early, as soon as the game has most of the functionality implemented in it. Do not leave QA testing just until the end of the project, because in some projects testers may uncover many critical bugs that will need to be fixed before the game is released.
- Allow many potential players to play a final version of your game by performing beta testing. It is invaluable for getting plenty of varied feedback from potential players.
- If you are developing a multiplayer game, you must test the server’s load capacity and stability by releasing a public beta version of the game and allow many people play it online. Testers can submit errors (bugs) or comments via email or use a dedicated online chat/message platform for that.

Playtesting:
- Playtesting sessions should be well organized and focused enough. Define whether it will be a QA (testing for bugs) or playtesting (testing for fun) session, but you should not mix them both.
- Constantly test and reevaluate the gameplay to make the game better. If there are big player tasks, consider redoing them and break them down into smaller ones.
• Project milestones (e.g. completion of level 1) should be testable, which should show visible progress whenever possible. Conduct playtesting of the milestones.

• Allow adequate time for tuning up and balancing the game according to playtesting feedback.

• Consider the “Mom test” as part of playtesting, asking yourself this question: “Could mom figure this out without reading the game instructions or tutorial?” Also, allow parents or grandparents to play the game and specifically let them test the game tutorial.

Miscellaneous:

• Strive for getting a solid game design done up front and a solid testing done on the back end.

• Take into account the possible hardware that potential players will use for running your game. Test the game on different hardware specifications, such as slower and faster computers.

• Sometimes, non-digital tools such as a whiteboard can be useful for keeping track of the parts of the game that are tested and the list of critical bugs that testers uncover, and for marking all the bugs that have been fixed. This way all the team developers can see and be up to date with this bug list and testing activities.

VII. DATA VALIDITY THREATS

Our qualitative analysis and results could be affected by our personal opinions and biases, and missing testing information from the post-mortems. In addition, the games that were analyzed in the post-mortems belonged to different genres, having different project requirements and development needs. This could affect the types of testing methodologies and testing results from the post-mortems. We also believe that the post-mortems’ authors could hide or not disclose some testing details from their game development projects. To make our post-mortem analysis reproducible and more transparent, we posted all the raw data available online in [].

VIII. CONCLUSIONS

This paper presented an overview of important game testing aspects described in recent game development post-mortems. Our methodology was practical for reviewing testing issues from the post-mortems, conducting both qualitative and quantitative data analysis from them. We recommend that stakeholders from a game development project should read many game post-mortems, and also read overview papers on post-mortems, since they provide with a number of benefits that may yield game development improvement, including an analysis of an overall reflection on what went right and what went wrong in game development projects, and useful best practices on black-box testing of video games, among other game software engineering aspects. For example, one important best practice that we found from the analyzed post-mortems is that the development team should test their game prototype early and often. Another one is that a team must dedicate enough testing time.

Future work will include a more extensive post-mortems analysis covering recent post-mortems, applying software tools for downloading the post-mortems’ web pages automatically (web data extraction) and thus saving time on this activity. We also plan to use a qualitative data analysis software package that will allow us to obtain a more comprehensive testing analysis from the post-mortems.

REFERENCES

The post-mortems in this book are the next best thing to actual game development experience. They follow projects from start to finish, talking about mistakes as well as good decisions, giving candid accounts, rather than just trying to abstract general guidelines. They record the experiences of average game developers as well as high-ranking producers. Each article is written in the same simple format. A member of the development team writes down how the game got made, starting from the initial vision and the starting goals, what kind of company and project team was involved, what tools were used.