Nuutti Heinäsmäki

THE USE OF POST MORTEM ANALYSIS IN GAME DEVELOPMENT

Faculty of Information Technology and Communication Sciences
M. Sc. Thesis
June 2022

ABSTRACT

Nuutti Heinäsmäki: The use of post mortem analysis in game development
M.Sc. Thesis
Tampere University
Master’s Degree Programme in Computer Science
June 2022

Post mortem analysis (PMA) is a method of development retrospection that has found its way into software development. PMA was the topic of a number of research papers in the 90s and early 2000s, but the research has since moved on to other subjects, despite leaving the discussion on some areas of PMA unfinished. Notably, the unsatisfactory rate of PMA adoption in the industry was identified but not addressed, while the new lightweight method of PMA was developed but not revisited with experience from the industry. PMA research is also very limited on the subject of game development, despite its interesting and unorthodox ways of utilizing PMA reports.

The thesis aims to study the adoption of PMA in the game industry, with a focus on the game industry’s PMA adoption rate and the PMA methods currently being used. Software development has trended towards more agile methodologies in the last decades and game development industry in particular is often noted to only use very lightweight or even ad-hoc methodologies during development, so the game industry offers a good viewpoint for studying if the traditional PMA methods are still in use and how they may have changed over the years.

Besides examining PMA adoption and methods in modern game development, this thesis also goes through the uses of PMA reports in game development. Game developers have publicly released hundreds of PMA reports, which is not a common practice in traditional PMA. The goals that the game developers have for the public reports also differ from the traditional ones. This thesis will focus particularly on public PMA report usage in game development research and the thesis will include a literature analysis on several game development research papers. The analysis shows that the game development research on PMA reports is consistent with other research and that it can also be complementary to other research, though limited in the discussed topics.

The study also features a questionnaire survey aimed at Finnish game industry professionals. The survey helps to answer the research questions of this thesis as it shows that PMA is a common practice and that the PMA method in modern game development has some similarities with traditional methods though it has adopted new lightweight practices in some aspects. The survey also brings to light that even though public PMA reports are well known in the games industry, the common uses for PMA reports in the industry have not changed from the orthodox uses presented in the prior research.

Key words and terms: Game development, Software development, Post mortem analysis

Contents

1. Introduction 1

2. Post mortem analysis (PMA) 3

2.1 Shift towards more lightweight PMA 4

2.2 General critique of PMA 6

2.3 Game development as an environment for PMA 8

3. Comparison of PMA methods 9

3.1 Collier method 10

3.2 Kerth method 11

3.3 Lightweight method 13

3.4 PMA in game development 14

4. Competing retrospective practices 16

4.1 Comparison of PMA and agile retrospective 18

5. Potential use of PMA reports 20

5.1 Game PMA reports 22

5.2 Studies on game PMA reports 26

5.3 Examining commonalities between the research papers 30

5.4 Discussion on PMA report research 35

6. Survey 37

6.1 Background information 37

6.2 Experiences with PMA 40

6.3 Experiences with other retrospective methods 44

6.4 Perception of retrospective methods 47

7. Discussion and conclusions 49

References 52

Appendix – survey questions and results

1. Introduction

Post mortem analysis (PMA) is the practice of analyzing a project after its completion. PMA in the context of software development has received a decent amount of research in the 90s and early 2000s. The specific strengths of PMA have generally been considered to be organizational learning and finding out why a project failed. There are also other potential benefits to PMA, that are rarely discussed in research: allowing developers to "vent" about the issues they faced, providing research data, and helping developers learn more from their projects as well as from projects they didn't participate in.

Game development has become a notable subset of software development and game developers have adopted many of their practices from software development, though game development has its differences that make game development measurably different from other software development [Murphy-Hill *et al.* 2014]. While improving practices and understanding why a project might fail are as important in game development as they are in software development, the laxer methodologies of game development bring to question how well PMA and other development practices have been adapted. Critique on lax game development methodology includes, for example, Lehtonen *et al.* [2020] who studied issues in game development and suggested that developers would benefit from using more requirements engineering methods as is done in traditional software development. Furthermore, it has been suggested that, in game development, a lot more focus is put on the individual developers compared to traditional software development [Murphy-Hill *et al.* 2014], which implies less focus is put on methodologies.

Game development offers an interesting application area for PMA as a subset of software development, but there is a lack of understanding regarding the use of the PMA methods in practice. Furthermore, game developers have set themselves apart from software developers by releasing some PMA reports publicly, as opposed to keeping them private, which is the standard in software development. The most famous publication with written PMA reports is the website *www.gamedeveloper.com*, which continues the legacy of the previously running magazine Game Developer, which also published PMA reports. The publicized data has been used for marketing, spreading knowledge among developers and also in various studies about game development [Petrillo *et al.* 2009, Washburn *et al.* 2016, Lehtonen *et al.* 2020]. Seemingly contradictory to some researchers' opinions on lax game development practices, at least some game developers have long since adopted PMA as is apparent from over two decades of PMA reports. PMA in game development is shrouded in mystery as there is limited research on the topic and only a few other publications that discuss it.

In this thesis the possibilities of PMA will be explored from a more modern perspective and particularly in the context of game development. Much of the research happened in the early 2000s and put a lot of emphasis on large software development organizations, but with the ever-growing and changing software and game industries, the amount and the significance of small development teams and studios is increasing. The different environments may change how PMA can be utilized and is worthy of research. This thesis aims at answering the questions of what is the current state of PMA adoption and what kind of PMA methods are used in the video game industry. Furthermore, this thesis also seeks to explore how PMA reports are used in game development and what kind of possibilities the reports have, that haven't been considered in earlier research.

This thesis will feature a questionnaire survey to gather information about the current state of PMA usage in the Finnish game development industry. The lacking research on PMA and game development leaves many questions unanswered so this thesis will attempt to shed some light on those questions through the survey. The survey aims at gaining some insight on whether PMA is commonly used in the industry, what kind of practices are employed in these methods currently and how the different uses of PMA reports are viewed. The survey will also gather some background data on the respondents to evaluate the validity of the survey. The survey will provide the real-life data that, together with data from literature, will answer the research questions about the adaption of PMA in modern game development.

This thesis will also include a literature analysis to help answer the third research question on the uses of PMA reports. The third question, on the possible uses of PMA reports will mainly focus on the use of PMA reports in research, which will be explored with the help of three research papers, which feature data that is based on collections of public PMA reports. The three papers were selected from a group of 15 papers that were found on *www.scopus.com*, with the following search: ("game development" AND "postmortem"), which targeted titles, abstracts and keywords. Two of the papers are the two most cited papers while the third paper is a newer one, which featured a different kind of data. Additionally, a fourth paper with more traditional forms of research was also selected. The fourth paper is the most cited result of the following search: ("software engineering" AND "game development"). The results of the four papers were reviewed together to determine if research using PMA reports was consistent with other research.

Section 2 starts with going through PMA in depth and Section 3 continues with examining several methods from PMA literature. Then, Section 4 addresses the other modern retrospective practices, particularly agile retrospectives, and compares them with PMA. In Section 5, the uses of PMA reports are examined with particular focus given to their use in research. Afterwards, Section 6 analyzes and discusses the survey data. Finally, Section 7 features the conclusions about the current state of PMA in the game development industry and the use of PMA reports.

2. Post mortem analysis (PMA)

Post mortem analysis (PMA) is known by many names, which vary from study to study. Post mortem may appear written separately, together or with a hyphen, and sometimes the whole term is replaced by a different one, such as project retrospective. At times an alternative term referring to PMA in one source might not refer to it in another. The research can get confusing to read, though the core idea of PMA is easy to identify. The "post mortem" part of the name is borrowed from "post-mortem examination", which is a term referring to autopsy [Wikipedia, 2022], a procedure in which a corpse is examined. Using "post mortem" to refer to a project after completion could be seen as dark humor [Glass, 2002], which may be one reason why the naming has been so inconsistent. This thesis will be using the short and concise acronym PMA, though it may not always be what the source papers have used. Furthermore, the written documentation of PMA will be referred to as PMA reports, though they are also referred to with various names in the literature.

Performing analysis after the completion, or the death, of the project is the core of PMA. In particular, the object of analysis is the development team and their experiences during the development. Finding out and analyzing the successes and failures of the development practices, as well as what issues impacted the development and how those issues were dealt with, can lead to better practices and improved ability to avoid and handle the issues in the future. The PMA activity usually includes some sort of a meeting or meetings between developers and potentially other relevant people as well as writing a report of the findings.

PMA is not specifically, or originally, a software development method. It is a general practice that can be done after any project to facilitate learning from the project. Prior to the software development focused research, PMA was already brought up a project management method, but as PMA has been adopted in software development, the practice has seen various changes and research that specifically examines PMA in this context. This thesis will focus on the developments and methods of PMA as it was adopted within software development as there is no research specific to its use in game development. The rise of the popularity of PMA withing software development research could be traced back to large software companies trying to improve their software development practices through an early version of PMA or similar practices. The reports on the successes of these methods, for example a paper from Collier *et al.* [1996], are some of the earliest and most referenced papers on the subject.

Another early paper on PMA style methods was by Nolan [1999], who focused on "Learning from Success". Later papers in PMA tend to follow the report structure mentioned by Myllyaho *et al.* [2004], which brings up successes and failures as the two different types of things to analyze in a project, but Nolan focused on the importance of the successes, which were being neglected. Nolan does not mention PMA or refer to it by any other name as he does not focus on any methods themselves, but the paper highlights the importance of recognizing the success factors from successful projects and applying the lesson that can be learned in future projects, which is a near perfect match to PMA. While failure often receives more attention in PMA, many researchers keep highlighting the importance of recognizing success as well, either for learning purposes [Nolan 1999], or motivating the project team during the PMA [Kerth and Weinberg 2013].

2.1. Shift towards more lightweight PMA

Some papers from the 2000s, such as Dingsøyr *et al.* [2001], Birk *et al.* [2002], Stålhane *et al.* [2003], and Bjørnson *et al.* [2009] show a new direction in PMA research, in which a new, more lightweight, method of PMA is considered. Specifically, many of the researchers tried to form lightweight PMA methods that would take very little time compared to the other PMA methods such as the method from Collier *et al.* [1996]. Interestingly the Agile Manifesto [Beck *et al.* 2001], which would pave way for significant changes in software development methods dates back to these times as well, which may have played a part in guiding researchers to aim for more lightweight methods. Agile Manifesto also includes one principle which arguably promotes the use PMA as a practice:

“At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly” [Beck *et al.* 2001]

The suggestion “At regular intervals” might allude to a practice that is done more often than only at the end of a project, but the rest of the principle matches PMA. Reflection and trying to improve are at the core of PMA. This Agile Manifesto principle may be the last principle in list of twelve and the main values from the Agile Manifesto might not favor “processes” or “following a plan” but reflecting on previous work is not a forgotten part of agile thinking. However, PMA methods are not the only methods of reflection, so despite fitting within agile thinking, PMA might not be commonly promoted within agile thinking. There are other methodologies like agile retrospectives that also fit the principle in the Agile Manifesto, which will be discussed later in the paper.

While the spread of agile thinking coinciding with the new PMA research angle and it similarly pushed for more lightweight methodologies, the Agile Manifesto might not deserve the credit for lightweight PMA methods. Collier *et al.* [1996] had made their process for projects with over a hundred employees, which is a very different environment from what smaller development teams have. The different kinds of software development environments may be contributing to a divide in methodologies, with more lightweight methods being needed by smaller teams for whom, the complex methods are not suited for. Large software companies with management staff may have been supportive of the earlier methodologies with more elaborate processes and benefits to the whole company or management staff, such as Collier *et al.* [1996] or Kerth and Weinberg [2013]. Lightweight methods PMA methods were, from the beginning, suggested by researchers to be a good fit for small or medium-sized teams [Dingsøyr *et al.* 2001, Stålhane *et al.* 2003]. Earliest papers on developing more lightweight PMA methods were also perhaps too early to have been greatly influenced by the Agile Manifesto as the papers advocating for PMA methods in smaller projects already start showing up in 2001.

One such early paper on lightweight PMA is by Dingsøyr *et al.* [2001], where the team activity and written report of PMA are both investigated separately. Specifically, the paper claims to investigate two separate methods for capturing experience: first, an experience report to document the project experience, and second, a postmortem review which is described as an investigative group process. The experiences brought forth by the two methods are found to have some differences in topics, which are likely caused by the methods used in the activities. The experience report was written by the project manager, so it focuses on their experiences as opposed to the other developers' experiences which were investigated by the researchers in a group process.

The PMA methods tend to place a large focus on the team activity, where the goal is to discover issues or successes from the project. Some papers [Dingsøyr *et al.* 2001; Stålhane *et al.* 2003] suggest using KJ method, named after its developer Jiro Kawakita, to graphically map the discoveries in a logical, easy to understand way during the meeting. The KJ method in PMA is based on "KJ method" described by Scuping [1997], who condenses the method into four steps:

1. **Label making**, people write down their observations on the subject individually
2. **Label grouping**, the group arranges their combined labels into "families"
3. **Chart making**, connections between "families" are marked down
4. **Written or verbal explanation**, examining and explaining the produced chart and potentially developing new ideas

The early recommenders of the KJ method for PMA are Dingsøyr *et al.* [2001] and Stålhane *et al.* [2003]. Most of their PMA method's time consumption comes from a single meeting where the KJ method is used as the primary activity. The purpose of the KJ method was to discover project experiences and analyze them quickly. Their meetings consist of a facilitator or facilitators, who guided the meeting and a number of project members who write the "labels" on post-it notes. The labels were any positive or negative experiences regarding the development. Afterwards the group continues to the grouping and charting steps, where the notes are reorganized. The explanation step is also an activity for the whole group, though the focus can be more on analyzing and planning for future changes. The end results of the KJ meetings can also include a fishbone diagram to illustrate an issue and its causes.

The use of KJ method has been popular in lightweight PMA papers and since its introduction, the KJ method has remained as the recommended way for handling the team activity in papers for lightweight methods. Later research for this type of PMA method has also suggested skipping some of the steps to combat time constraints [Myllyaho *et al.* 2004] and replacing fishbone diagrams with causal maps [Bjørnson *et al.* 2009].

2.2. General critique of PMA

While PMA literature doesn't feature much self-criticism, some less formal complaints and warnings are sometimes made in the sidelines. The complaints are often brought up and ridiculed by writers, such as Kerth and Weinberg [2013], who feel that issues with PMA stem from improper usage of PMA methods and not methods themselves. One critic, Hamann [2003] wrote in July 2003 article of Game Developer magazine, that while it's better than nothing, PMA reports always show the same problems, meaning that they aren't getting fixed. Hamann [2003] based his claims partly on observations of public PMA reports and partly on discussions with participants of Game Developer Conference on the topic of PMA. According to him, game developers had more negative than positive things to say about PMA. Though this is from quite some time ago, it may suggest poor adoption or difficulties in utilization of PMA in game development. Or perhaps as Hamann [2003] claims, PMA methods really have flaws.

While there appears to be no common issues brought up by different researchers about the idea of PMA itself, issues with the actual use and implementation of PMA have been brought up. One consensus among researchers seems to be that PMA is not used often enough [Glass, 2002], and some studies have backed this common claim up by numbers. One such study is by Verner and Evanco [2005], who studied data from 122 software projects and noted that only 29% of the projects had a postmortem review. The study reasoned that the managers didn't understand the benefits of PMA which was partly to blame for the low usage. Verner and Evanco [2005] insist that PMA should be done more:

> **"Postmortem reviews are important for process improvement, but companies seldom perform them. As a result, they tend to repeat the same mistakes."** [Verner and Evanco, 2005]

While researchers always seem to call for more PMA in their papers, even to the point of every project needing one, they sometimes stop to contemplate on what situations PMA is suited for [Birk *et al*. 2002]. The possibility of PMA not being suited for a project would imply that not every project needs PMA, but the way to make that decision receives little discussion. Perhaps because many of the researchers are experienced as facilitators, they do not have much to say about how to choose whether a project will receive one, since that decision is made by the project management and not by the facilitator.

The discussions on the lack of PMA usage is often accompanied with the idea that people don't always understand the benefits of PMA [Verner and Evanco, 2005]. The lack of value seen in PMA leads to disinterest in allocating time for PMA methods and the inability to utilize PMA reports. Some researchers discuss these shortcomings. For one, Collier *et al.* [1996] try to highlight the importance of management to properly commit to making the changes proposed in a PMA report.

"Long-term success requires that the organization now go that extra mile to turn what "everybody knows" into what "everybody does." [Collier *et al.* 1996]

Another warning from Collier *et al.* [1996] is that PMA meetings can become unproductive if not handled properly. A similar note is made by Kerth and Weinberg [2013], who write based on their experiences. One of the issues they mention is that PMA sessions can become very negative and delve into hostile finger pointing. Kerth and Weinberg also mention that the lack of time allocated for the retrospective contributes to a lack of proper analysis and the inability to reach and implement real solutions. Furthermore, PMA reports are often left unused and forgotten. Kerth even shares a personal experience of finding out that a collection of PMA reports had been discarded in a company he worked at in the past. He had come back to recover 86 PMA reports, which he had worked with before, but someone had thrown them away to save space in a filing cabinet. While the uses of PMA reports are discussed later in the paper, it suffices to say that they are not so worthless that they should be thrown away to save cabinet space.

Both Collier *et al.* [1996] as well as Kerth and Weinberg [2013] are proponents of more heavyweight, time-consuming PMA methods in software development and they share plenty of warnings in their works. While not all proponents of heavyweight PMA do the same, the papers on lightweight methods, such as Dingsøyr *et al.* [2001] and Stålhane *et al.* [2003], hardly mention a negative experiences or give out a warning. This is one area of research where lightweight PMA papers are lacking. It may be because the heavyweight PMA papers were written after a lot of experience had been built up in using PMA methods in the industry, so the writers knew about potential issues in implementing their methods. The lightweight papers were bringing up an entirely new, and largely untested, approach that smaller software development teams could use in the future. Unfortunately, there isn't much research on PMA methods and their use going into 2010s, so lightweight PMA never received the same kind of thorough research backed by long-term experience.

2.3. Game development as an environment for PMA

Game development offers an interesting environment for PMA for a few reasons. One of the reasons is the lack of research on the subject and another is the potentially differences that PMA may have in game development. Researchers often note how game developers have a tendency to not follow industry standard practices. There is not much research on PMA usage in game development, but other research on game development methods suggests lacking or ad-hoc practices. For example, Koutonen and Leppänen [2013] found that the results of their survey showed less frequent use of agile methods and practices among game developers when compared to another study on software development, while similarly, Murphy-Hill *et al.* [2014] found lack of software engineering process to be a trend in game development in their interviews. Furthermore, it has also been suggested that even when game developers adopt development practices, they overrate their own commitment. One such note was made by McKenzie *et al.* [2019] who compared game developers' self-ratings on following agile practices to a professional made review.

Murphy-Hill *et al.* [2014] suggests the lacking use of development practices may stem from the game development environment, which is unpredictable and focuses more on creativity. On the other hand, game developers may tailor development practices to suit their needs separate from the software development community as is also noted by Murphy-Hill *et al.* [2014]. Furthermore, Kultima and Alha [2010] studied innovation among game developers and noted "The lack of tools and education in the field is present in the opinions of how to invoke innovation in practice. 75% of our interviewees had no formal education in game development and most of their practices had been formed through their personal experiences". The lack of research and documentation on game development practices may also contribute to game developers having more personalized development practices.

3. Comparison of PMA methods

Next, we will go over the tasks in three PMA methods to elaborate how PMA can be done in practice. First, however, a method of examining PMA will be introduced based on a paper by Birk *et al.* [2002]. There are four steps identifiable in the paper: **'Preparation', 'Data collection', 'Analysis', 'Results and experience'**. The goal of the paper was not to define a single method but to generalize and suggest some possibilities for how to conduct different steps of PMA in the first three steps. Using these steps to investigate the different methods will help in drawing comparisons, so they have been included in the summary Figures 1-3, that will come up later.

Birk *et al.* [2002] mention reviewing available documents as the way of walking through project history and determining how to progress with the PMA and also that for external facilitators the preparation step will be especially important, as they would be especially lacking in background information about the project and the company. Birk *et al.* [2002] leave interviews for the data collection phase, that will follow after initial preparations, but interviews, surveys, or other methods of collecting data could also serve the preparation phase instead or in addition to being part of the data collection for the analysis. This will be a common occurrence in the methods that will be discussed, as this thesis will try to separate the data collection in preparation step and data collection for the analysis step into two different sections, somewhat contradictory to Birk *et al.* [2002]. Ultimately, a clear line is hard to draw, as the preparation work will invariably contribute to the analysis in one way or another. Generally, data collection does involve group discussions as a core component, though it would still be possible to employ other methods of collecting data. In the third phase, analysis, the previously collected data is analyzed to form an understanding of the project, that will help make improvements or bring notable experiences to light. Despite relying on the previous step, data collection, analysis can be performed in the same meetings, even as part of the same activity as is done in the KJ method.

The fourth step, results and experience, refers to the report written based on the previous stages of PMA. It has been included, despite not being considered a PMA step in the same way as the other steps in the paper [Birk *et al.* 2002]. The PMA reports are important, but different researchers have different ideas about how important the reports are and how the reports can be used.

Furthermore, some other key parts of PMA methods deserve their own mentions so a few more points will be included in this examination. First, it's worth nothing who the methods suggest as the **'Facilitator'**, the person overseeing the PMA method from start to finish. Secondly, the **'Goal'** of every PMA method, will be examined based on what the referred works showcases as their method's accomplishment. Thirdly, '**Time required'** will also be featured as it varies a lot due to the differences in the methods.

3.1. Collier Method

The first method, named Collier method, is based on the method described by Collier *et al.* [1996], who made the method for the large company they worked in (Apple). The method was designed with hundreds of project members in mind, so it differs a lot from methods proposed by later researchers who focused on smaller projects. This method was also designed with a lot of time in mind. Preparation phase would include holding a large survey and going through data regarding the project. The included instruction for the survey suggested an online questionnaire for all of the project participants that would take between 30 minutes and an hour to complete with about a hundred questions in total. It is also mentioned that metrics such as *lines of codes changed* and *defect finding and closing* should be collected, which means that tools for collecting that data must already be active during the project.

In the data collection phase, a series of coordinated meetings are to be held to give everyone in the project team a chance to voice their thoughts. It is recommended to fill three management roles for these meetings: a chairman, a member of project management; a coordinator, who helps to organize the meetings and write down what is discussed; and a facilitator, an outsider of the project who leads the meeting and protects the participants from the project's own management. Collier believes that a lot of skill is required from the facilitator to make sure that the meetings produce the proper data needed for the later analysis. A therapeutic benefit to participants in form of venting is also mentioned. The later analysis step is performed in a separate meeting with a workday being fully dedicated to it and with only the relevant participants invited.

The reporting aspect was considered very important and the report itself was meant to be directed to project leaders within the organization as well as upper management to facilitate improvement. There was arguably an even further step beyond writing and distributing the report, which was assigning the responsibility to investigate the issue found in the report and implement a solution to a single person. The main goal of the method is making changes in the organization. The suggestions and warnings at the end of the paper, such as "Ultimately, no changes will occur in the organization unless someone is held responsible", are all about turning PMA reports into real changes. The paper does not, however, make it clear who is in overall charge of the PMA. Often methods have a designated person investigating the project, running the meetings, and writing a report, though they may do these things with the help of other people. Generally, this person is the facilitator. This method does use facilitators in the data collection and analysis meetings, but at other times, project leads or other management staff are handling the work of the PMA.

Collier method [Collier *et al.* 1996]

Preparation	**Survey project participants and gather project data**
Data collection	**Multiple coordinated team meetings with project participants**
Analysis	**Another meeting with key people to analyze data**
Results	**Write a publication for the results and implementing a solution**
Facilitator	**Various, including project lead, company management and facilitator for meetings**
Goal	**“Therapeutic” benefits to developers; Company management learns from projects and implements changes**
Time required	**Not explicitly stated, but presumably several weeks or months**

Figure 1, Collier method.

3.2. Kerth Method

The second method, named Kerth method, is from a book by Kerth and Weinberg [2013]. Kerth worked as a facilitator for retrospective meetings in software development companies and wrote a book about his experiences, which was originally published in 2001, though this thesis will refer to the later digital edition from 2013. In the book, Kerth and Weinberg refer to their method as project retrospective team review and they consider post mortems as an even more intense version of project retrospectives. However, their idea of a project retrospective team review is already in line with other PMA methods. Within the book, the PMA version of project retrospective focuses more on learning from project’s failure and needs more time and effort but is otherwise same. Compared to other methods of retrospection, Kerth method is still much closer to PMA than something else, like an agile retrospective.

Similar to the Collier method, the preparation phase in Kerth method is very important. The facilitator must acquire a solid understanding of not only the project, but also about the project participants feeling towards the project. The meetings in the Kerth method require planning to decide how they are conducted, and this depends on the project members existing knowledge and understanding of the project. The preparatory data gathering is done before the meetings, through interviewing each participant, which will help the facilitator to figure out how they can help the participants and manage the meetings.

The meetings, which are the main form of data collection, are the bulk of the work in this method. Generally, three days are suggested to be reserved just for meetings and other activities, though this is only the minimum presented in the book. The possibility of taking even more days for the meetings is discussed, while the possibility of taking fewer is strongly opposed. There are mentions of trying meetings that only last only half an hour, an hour or half a day, but it is claimed that such meetings did not result in any actual changes and thus, that not using enough time would simply make the whole method a waste of time.

The meetings in this method require a few days, as there are many activities to be held during them. Some of the activities are about going through the project experiences and learning from them, as would be expected of PMA, however there are also many activities that are made to help the developers grow a more mature perspective towards their experiences or to learn to cope with their failures. The analysis is largely done during the meetings as well. The book suggests that the activities during the meetings to offer an ideal method of learning for the participants. The facilitator's job is to help the meetings to move forward and lead the participants into making conclusions themselves.

The learning in the Kerth method relies on the meetings instead of the final report, so the importance of the reports is lessened. The reports aren't considered a mandatory part of the PMA. It is mentioned that the report will likely just get filed away, but some value is still given to the reports, with mentions that the reports become more valuable over time. The reports are useful once the meeting isn't in fresh memory anymore, or if people who didn't participate in the meeting need to review the experiences from the project. Furthermore, the value of a collection of reports is talked about in the book, both in regard to a company with reports of their own projects as well as to an outsider, like a professional facilitator, with their clients' reports.

The goal of the method is the learning during the meetings through the many activities, which is different from the Collier method. However, since the project lead, and other relevant management staff, should be part of the meetings as participants or because they were called to participate in some specific meeting activity them, they may already learn with the other participants and plan future changes without needing the report to do it. In that sense, the goals between the Collier and Kerth methods are quite similar. While the changes to be done are already considered during the meetings, ultimately it is left for the participants and their management to handle in the future, although Kerth and Weinberg [2013] mention doing checkups on old clients and potentially offering further services.

The Kerth method is made so that just a single facilitator could lead it from start to finish. This isn't necessarily ideal, however, as having multiple facilitators is mentioned quite a few times by Kerth and Weinberg [2013]. Facilitating the PMA is treated as a job for a professional outside the company, though it's mentioned that a large company may be able to manage a department dedicated to such work internally. Regardless, the facilitator is expected to be a project outsider and anything else would be considered problematic [Kerth and Weinberg, 2013]. Running the various possible activities and keeping the mood of the meeting right takes experience, but not only that. People need to be open about their experience without fearing retaliation from coworkers or management, even when the experiences in question are negative and make others defensive. Because of the many potential issues of interpersonal relationships in a workspace, especially, though not limited to, those between superiors and subordinates, it is important for the facilitator to have no connections.

Kerth method [Kerth and Weinberg, 2013]

Preparation	**Interview project members and make preparations for meetings**
Data collection	**Hold three or more days of meetings, according to premade plan, between relevant team members and potentially valuable outsiders**
Analysis	**Perform analysis during the meetings**
Results	**Meeting participants figure out improvement opportunities from meetings; A report is produced**
Facilitator	**Experienced facilitator from outside the project, potentially outsider to the company**
Goal	**Meeting participants learn from the project and implement changes; Company and the facilitator acquire PMA reports**
Time required	**3-5 days for the meetings and few more days for other work**

Figure 2, Kerth method.

3.3. Lightweight Method

The third method, named Lightweight method, is based on various papers describing a new lightweight PMA method that utilizes the KJ method for data collection and analysis. One of the earliest such papers is by Dingsøyr *et al.* [2001]. This method is commonly used in PMA papers, and it has been described in detail in a few of them, with some minor differences in practices. The explanation of this method is based on two early papers, i.e. [Dingsøyr *et al.* 2001, Stålhane *et al.* 2003], which share some of their writers and describe a fairly identical method.

For the preparation phase, generally, it is recommended that some kind of preliminary data gathering is held to provide the researchers with enough knowledge to manage the rest of the PMA. Interviewing a project manager is the most popular method, while reviewing project documentation is another common one. In contrast to the previous two methods, the papers don’t put a lot of focus on the preparation phase, or in the importance of facilitators understanding the feelings of the participants. The papers feature very little discussion on such preparatory work and the method doesn’t allocate much time for them either. For example, Stålhane *et al.* [2003] suggested 4-5 hours would suffice for the “researchers’ preparations”, though it was not explained how that time was to be spent. This may be largely because a KJ meeting requires less direction and coordination. The Lightweight method is supposed to be quite simple after all and the KJ method is adopted to serve as a simple way of collecting and analyzing data as a group.

At the start of the meetings, where the data collection and analysis occurs, the facilitators briefly explain how the KJ method works and afterwards the meeting proceeds with only minimal directing while the participants apply the KJ method. The meeting, despite being most of the workload for the PMA method, is only expected to last a few hours at most. Size wise, the early papers [Dingsøyr *et al.* 2001, Stålhane *et al.* 2003] appear to have two researchers working with most of the development team during the meetings, though the number of participants doesn't receive much attention.

In the Lightweight method, writing a report appears as the final step that happens after a PMA session. The goal of the report is to help developers and companies to document their knowledge, which would help them learn from their experiences. It is, however, also noted that the KJ meetings themselves already have this kind of an effect. This isn't to suggest that documenting the results of a KJ meeting isn't worthwhile, for one, it is necessary for communicating the results outside of the development team who participated in the meeting, though this would require an effort in knowledge management by the relevant company. In any case, company side issues in utilization of PMA reports are not often considered in the papers. Perhaps researchers felt such considerations to be outside of their scope. The value of the report is left a bit unclear, as there are already benefits from just the PMA session.

	Lightweight method [Dingsøyr et al. 2001, Stålhane et al. 2003]
Preparation	**Interview project manager and gather project data**
Data collection	**Hold a KJ meeting with project participants**
Analysis	**Perform root cause analysis during the KJ meeting**
Results	**Write a report and get feedback for the report from project participants**
Facilitator	**Project outsider**
Goal	**Participants and company management learns from projects**
Time required	**~5 hours of meetings and ~10 hours of other work**

Figure 3, Lightweight method.

3.4. PMA in game development

While the existing literature leaves unclear what kind of PMA methods game developers use, some references to PMA do exist in game development literature. One example, which will be examined here, is a book by Chandler [2013], which gives instructions on how to manage a game project. In the book, PMA is considered to be a part of a game's development process. According to Chandler [2013] the goal of PMA is to learn from mistakes and improve the development methods through a discussion between the entire development team, although Chandler [2013] also mentions that the PMA meeting is a chance for the team to congratulate each other on the job well done.

The description for the preparation step of the PMA differ from the descriptions in the previously described methods. For the preparation step of the PMA, the book instructs to gather all relevant information before the meeting, sharing it with the entire team and then asking them to prepare for discussion on the topics that the PMA will concern, such as what went wrong or right. This is to help the discussion in the meeting.

However, the meeting which encompasses the data gathering and analysis is not described in much detail despite its significance. The PMA team meetings described in the book are just moderated discussions on the topics of the PMA. There are no mentions of KJ method or other specific methods to use in the meeting. This shows that the Chandler [2013] did not intend to challenge more traditional PMA methods by creating a new, better method for game development. The book makes multiple references to other sources for information on different methods for conducting a PMA and while it suggests a unique way to conduct a PMA, it does not assert its methods as the correct one. Instead, Chandler [2013] suggests that the reader should determine how thorough of a method is appropriate for the project in question and proceed accordingly.

In the book, a PMA report is referred to as “Lessons Learned document”, but it is more or less the same as PMA reports from the earlier methods and is treated as such. Though publicly released “postmortems” from Game Developer Magazine are mentioned as good examples in the book, the book uses a different name for PMA reports and gives no consideration for public report releases.

Some connections between Chandler’s [2013] description of PMA and the methods described earlier can be drawn, but there isn’t a perfect match among them. The most thorough methods for conducting PMA, like the Collier method, are appreciated in the book, but the exemplary practices that Chandler describes are instead very lightweight. However, the Lightweight method described earlier differs by using the KJ method during the meeting and having less preparation. Furthermore Chandler’s [2013] description includes a suggestion for holding “minipostmortem” between each different development phase and the proper postmortem at the end of the project. This idea is for longer project so that they can continually improve the development process at faster intervals than only between game projects.

4. Competing retrospective practices

While PMA faces little critique from researchers, there is one inherent factor in PMA that has been contested. As PMA is done in postmortem, after a project has concluded, the lessons are learned for the future and do not help with the project itself. Other similar methods of analyzing a project during the project's lifespan exist, and arguably they have a significant advantage, as the lessons can be put to work immediately. This kind of alternatives have been highlighted even in PMA research, for example Myllyaho *et al.* [2004] describe a case study, in which, a method, which they named post-iteration workshops, used PMA-like meeting was held multiple times during the project, after a major iteration. The method received praise for being able to affect the project before its conclusion instead of the issues only being dealt with after the project is already finished.

Rather than a unique method, post-iteration workshop is more of an instance of agile retrospective, which is a broad term for retrospective methods that were made for agile development. The term "post-iteration workshop" might have been short lived though, as the term "agile retrospective" has been used in other papers and is sometimes done post-iteration. In accordance with the agile manifesto, mentioned previously, Agile development practices promote reflection at regular intervals. While PMA somewhat fits this, the agile world may be more prone to adapting some method of agile retrospectives over PMA. In practice, any form of reflection done at some kind of intervals during a project that follows agile development practices can be called an agile retrospective method. Common intervals are after a sprint or a regular time interval, like for example a month.

Some proponents of agile retrospectives, specifically sprint retrospectives, are Schwaber and Sutherland [2020], who suggest that the retrospective should last a maximum of three hours for one-month sprint. They describe the sprint retrospective as the last step of the sprint. In the retrospective, the experiences from the sprint are discussed, particularly what went well, what didn't go well and what kind of solutions were made or not made. This description for the topics is quite identical to some descriptions in PMA literature. Other sprint related discussion events include a daily scrum, for communication over the progress of the sprint and sprint review, which is a discussion with stakeholders over the outcome of the sprint, but these discussions do not cover the reflection and learning of a retrospective, and on the other hand, in the sprint retrospective there is no need for other discussion than the retrospection.

Further proponents of retrospectives, though not necessarily agile retrospectives, are Kerth and Weinberg [2013], who, as mentioned previously, supported PMA like project retrospective team reviews. Kerth and Weinberg's book on the matter makes little distinction between PMA and project retrospectives but compared to some of the ideas presented by proponents of agile retrospectives, the project retrospectives are much more on the side of PMA, especially with final written reports being held in high regard and the focus on examining the whole project, which can't be done until the project is done. On the methodologies side, the project retrospective does appear to match both other PMA methods as well as agile retrospective to a large degree, though Kerth and Weinberg are more in favor of longer sessions after a project rather than short meetings during.

The "agile" in agile retrospectives isn't necessarily important. Instead of agile retrospectives, Derby and Larsen [2006] talk about iteration retrospectives in their book. Their idea of iteration retrospectives would work naturally in iterative agile development, as they suggest the interval between meetings to be between a week and a month, but they suggest it for any kind of development, agile or not. The retrospective only needs one meeting between the developers and could be as short as fifteen minutes, though many variances are presented for how much time should be prepared. Even half a day or multiple days are mentioned, though an hour or two is presented as most common. The meeting of the retrospective is led by a facilitator, though the potential for the facilitator to be a team member is not shunned, like it is in PMA literature. Furthermore, a written report of the retrospective is hardly mentioned, besides writing out the future ideas for the development during the next iteration.

Besides the few differences presented here, the ideas in the book [Derby and Larsen, 2006] are actually very similar to those of Kerth and Weinberg [2013] and the writers do talk about Kerth and Weinberg's book on retrospective as well. Derby and Larsen consider their lightweight iteration retrospective approach good for learning small things during the project but consider Kerth and Weinberg's "project retrospective" to be good for learning bigger things from the whole project. Regarding the historical development of retrospective practices however, the writers note a movement from "project retrospectives" to "iteration retrospectives", which may be signaling the fall of PMA in favor of agile retrospectives.

Some idea, about how widespread agile retrospectives are, can be found in the 15th Annual State of Agile Report [Digital.ai, 2021], which claims that 83% of organizations use "Retrospectives". It is not specified how extensive the retrospectives are, which leaves a lot to imagination, especially since non-agile retrospectives also exist. The difference between "retrospective" according to Kerth and Weinberg [2013] and "retrospective" according to Derby and Larsen [2006] is quite notable as they are essentially talking about PMA and agile retrospectives, though their methods involve similar practices. Looking at the definition from the organization that made the report however, term "retrospective" appears to be more in line with other definitions of agile retrospective rather than PMA:

"Retrospective: a session where the team and scrum master reflect on the process and make commitments to improve" [Digital.ai, no date]

4.1. Comparison of PMA and agile retrospective

A simple comparison between PMA and agile retrospectives can be seen in Figure 4, which somewhat follows the format of previous Figures 1-3. The most notable difference between PMA and agile retrospectives is the frequency that the method is used. While looking at the whole project during PMA has its benefits, it is not possible to improve the development methods during the project with this approach. Agile retrospectives on the other hand focus primarily on improving development methods and fixing issues as they come up.

	Generalized PMA	Agile retrospective [Derby and Larsen, 2006]
Method usage	**After a project is completed**	**During a project with intervals between a week and a month, or at project milestones**
Preparation	**Preliminary interviews and gathering project data**	**Bring data about the recent development**
Meetings	**Gather and analyze project data from project members**	**Gather and analyze recent data from project members**
Meeting time consumption	**Between several hours and several days**	**Between an hour and a day**
Facilitator	**Project outsider**	**Project lead or other participant**
Report	**Document meeting results, get feedback for the report from participants**	**Document experiences and write down the plan for the next development cycle**
Goal	**Participants and company management learn from projects**	**Improve development practices and tackle issues as they come up**

Figure 4, PMA and Agile retrospectives compared.

Frequent meetings may also be more fit for modern software and game development as projects rely more on long term support and further development after launch. The change in game development can be seen in the rise of subscription and free-to-play games [Flunger *et al.* 2017]. Another example of long-term game development comes from the business model of Paradox Interactive [GDC, 2018], who plan to keep expanding their games for years after release, citing an "indefinite life cycle". The policy of Paradox Interactive also highlights an issue with PMA as they have released a PMA report of one of their games, Stellaris [Game Developer, 2016]. The game was released on the same year as its PMA report, so the PMA couldn't have examined the game's whole lifetime as even now in 2022, the game receives new updates and paid content [Valve, 2022].

The issues with covering long projects using PMA in game development may be significant and may call for multiple different attempts of analysis during different times of production. One such proposal is Chandler's [2013] suggestion of "minipostmortems" between the development phases such as preproduction, alpha, beta, and release. While this would be similar to agile retrospectives between iterations, development phases could make for much longer iterations than what agile retrospective proponents say as Schwaber and Sutherland [2020] use a sprint as interval while Derby and Larsen [2006] suggest intervals to be between a week and a month. Multiple instances of PMA during a project is not a well-covered topic in literature and it is rather contradictory to the name “post mortem”, but modern development may benefit from this kind of a change in the method.

The other difference between PMA and agile retrospectives is the use of reports. In PMA, reports are often talked about as a valuable way of recording the results [Dingsøyr *et al.* 2001; Stålhane *et al.* 2003; Keith and Weinberg, 2013], though the reports aren’t seen as mandatory. Other times, the reports may be considered to be a significant part of the method [Collier *et al.* 1996]. In agile retrospectives, reports do not appear to have much significance and they might not be mentioned at all [Schwaber and Sutherland 2020]. Derby and Larsen [2006] mention documenting experiences and making a plan for the next development cycle before the next retrospective, but the focus is on bringing the changes to practice in the development team and not on making a report to someone. Arguably the goals for agile retrospective are different as well, since the focus is on making immediate changes and not on deeper analysis that may be done in PMA.

Otherwise, PMA and agile retrospectives are rather similar. The key idea of retrospectives is to examine past development and learn from it. This can be conveniently done in a meeting between developers. The popularity agile retrospectives may be a threat to PMA as they share a lot and agile retrospectives are, as the name suggest, more in tune with agile development practices. However, agile retrospectives do not promote producing reports and neither does it promote reviewing the entire project in a larger scale. Also, the potential for analyzing is greater after the whole project or a large section of it is finished rather in the midst of it after a short iteration. Some of the benefits of PMA reports are clearly not found within agile retrospectives, so PMA can still be considered a separate practice with its own benefits.

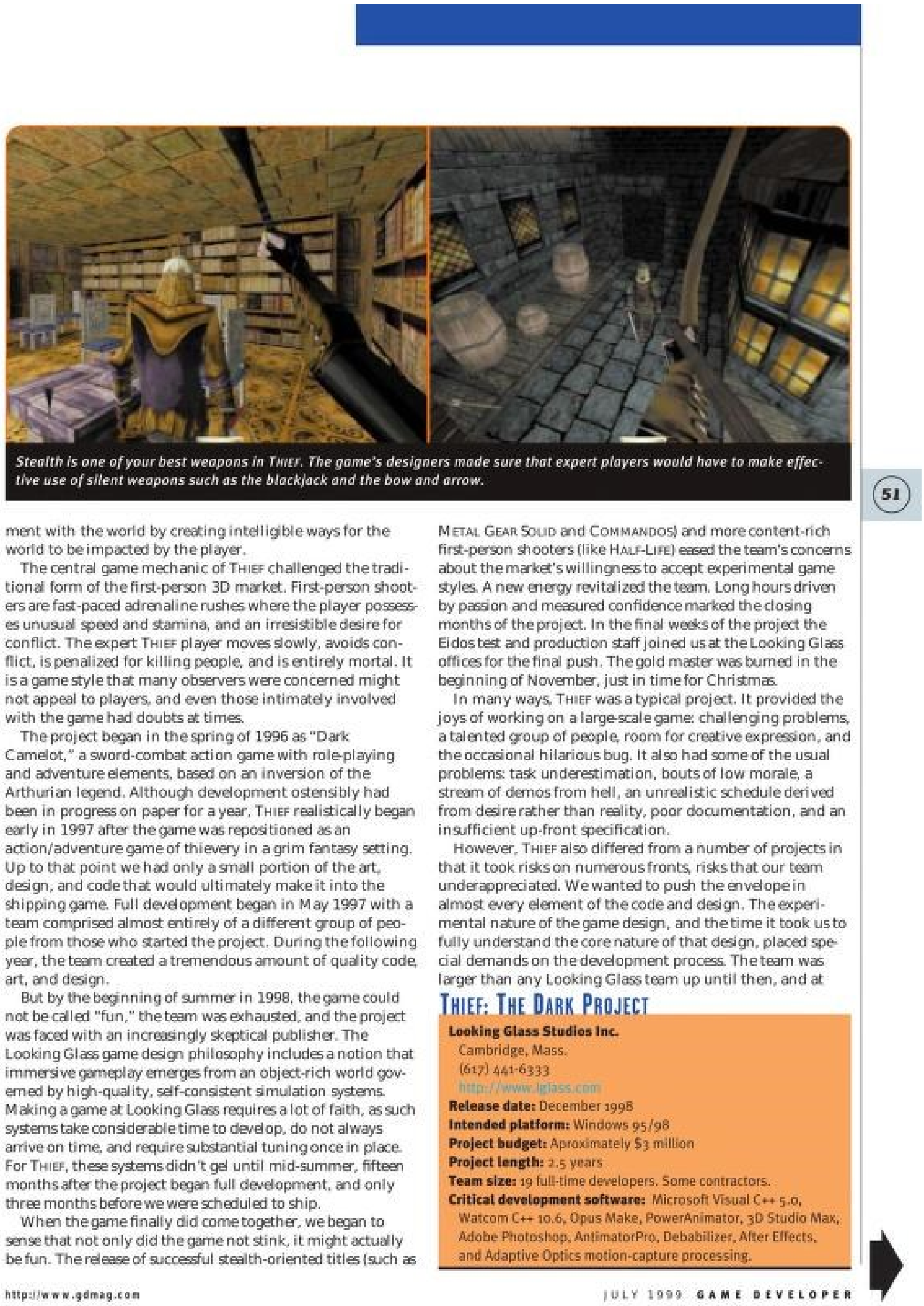
5. The potential uses of PMA reports

In the research on PMA, the main goals of PMA are commonly to educate managers about how to conduct projects better and to help organizations as a whole to avoid repeating mistakes. This was the goal of the PMA method by Collier *et al.* [1996] and similar thinking gets repeated by others, for example by Birk *et al.* [2002]. PMA reports usually play a significant role in achieving this goal.

"The summary of the findings is published and presented in a way that enables future projects to know what processes or tools are important to continue, and to turn problems into improvement activities" [Myllyaho *et al.* 2004]

This use of PMA reports relies on people to utilize the reports in the future, which is not something that will necessarily happen. A different approach to learning is taken by Kerth and Weinberg [2013], who wanted to make the learning happen during the PMA meetings. For Kerth and Weinberg [2013], the reports aren't a necessarily part of PMA, but the book still features various potential uses for the reports, such as: recordings of project development, reminders of what was discussed in PMA, a way to convey the discoveries of the PMA to people who didn't participate, and forming data for making return-on-investment calculations for improvement measures. Despite these many uses, it's mentioned that the reports are not always used, even when the client specifically asked for one.

The previous uses of PMA reports do not require the reports to be released publicly and PMA reports are not generally made public. The public game development PMA reports that have been released, were made public with different goals in mind. Being posted on a magazine, such as in Figures 5 and 7, or online, they were made to reach a large audience of people interested in game development. Public game related content promotes the game and thus works as a method of marketing. However, as PMA reports are much more on the technical side than marketing content usually is, the experience sharing may have been the main goal rather than trying to market the game. The benefits of spreading knowledge and experience from one developer to another through PMA reports is an interesting possibility. It is quite similar to the traditional goal of PMA reports, but rather than being limited the one's own organization it can reach the whole community.



Stealth is one of your best weapons in THIEF. The game's designers made sure that expert players would have to make effective use of silent weapons such as the blackjack and the bow and arrow.

51

ment with the world by creating intelligible ways for the world to be impacted by the player.

The central game mechanic of THIEF challenged the traditional form of the first-person 3D market. First-person shooters are fast-paced adrenaline rushes where the player possesses unusual speed and stamina, and an irresistible desire for conflict. The expert THIEF player moves slowly, avoids conflict, is penalized for killing people, and is entirely mortal. It is a game style that many observers were concerned might not appeal to players, and even those intimately involved with the game had doubts at times.

The project began in the spring of 1996 as "Dark Camelot," a sword-combat action game with role-playing and adventure elements, based on an inversion of the Arthurian legend. Although development ostensibly had been in progress on paper for a year, THIEF realistically began early in 1997 after the game was repositioned as an action/adventure game of thievery in a grim fantasy setting. Up to that point we had only a small portion of the art, design, and code that would ultimately make it into the shipping game. Full development began in May 1997 with a team comprised almost entirely of a different group of people from those who started the project. During the following year, the team created a tremendous amount of quality code, art, and design.

But by the beginning of summer in 1998, the game could not be called "fun," the team was exhausted, and the project was faced with an increasingly skeptical publisher. The Looking Glass game design philosophy includes a notion that immersive gameplay emerges from an object-rich world governed by high-quality, self-consistent simulation systems. Making a game at Looking Glass requires a lot of faith, as such systems take considerable time to develop, do not always arrive on time, and require substantial tuning once in place. For THIEF, these systems didn't gel until mid-summer, fifteen months after the project began full development, and only three months before we were scheduled to ship.

When the game finally did come together, we began to sense that not only did the game not stink, it might actually be fun. The release of successful stealth-oriented titles (such as METAL GEAR SOLID and COMMANDOS) and more content-rich first-person shooters (like HALF-LIFE) eased the team's concerns about the market's willingness to accept experimental game styles. A new energy revitalized the team. Long hours driven by passion and measured confidence marked the closing months of the project. In the final weeks of the project the Eidos test and production staff joined us at the Looking Glass offices for the final push. The gold master was burned in the beginning of November, just in time for Christmas.

In many ways, THIEF was a typical project. It provided the joys of working on a large-scale game: challenging problems, a talented group of people, room for creative expression, and the occasional hilarious bug. It also had some of the usual problems: task underestimation, bouts of low morale, a stream of demos from hell, an unrealistic schedule derived from desire rather than reality, poor documentation, and an insufficient up-front specification.

However, THIEF also differed from a number of projects in that it took risks on numerous fronts, risks that our team underappreciated. We wanted to push the envelope in almost every element of the code and design. The experimental nature of the game design, and the time it took us to fully understand the core nature of that design, placed special demands on the development process. The team was larger than any Looking Glass team up until then, and at

THIEF: THE DARK PROJECT

Looking Glass Studios Inc.
Cambridge, Mass.
(617) 441-6333
http://www.lglass.com

Release date: December 1998
Intended platform: Windows 95/98
Project budget: Aproximately $3 million
Project length: 2.5 years
Team size: 19 full-time developers. Some contractors.
Critical development software: Microsoft Visual C++ 5.0, Watcom C++ 10.6, Opus Make, PowerAnimator, 3D Studio Max, Adobe Photoshop, AntimatorPro, Debabilizer, After Effects, and Adaptive Optics motion-capture processing.

http://www.gdmag.com

JULY 1999 GAME DEVELOPER

Figure 5, An excerpt of Thief: The Dark Project postmortem [Leonard 1999] retrieved from www.gdcvault.com/gdmag.

Forming research data sets from PMA reports is another way to use public PMA reports, but it is unique to game development research and was unlikely to be considered by the writers of the public PMA reports. However, forming data sets isn't the only way of using PMA or PMA reports for research. Much of the PMA based studies are like the early research papers on lightweight PMA: the paper features a case study, in which PMA was conducted in a company or multiple companies with the help of the researchers, this is the case for the papers by Dingsøyr *et al.* [2001] and Stålhane *et al.* [2003].

There are also studies that don't focus on PMA itself but use case studies featuring PMA to study a specific subject, like the failure of a software project. PMA can be utilized to help understand what happened in the project in question. One such exemplarily study is by Ahonen and Savolainen [2010]. They wrote about the analysis of five cancelled projects which they analyzed as outsiders. First, they helped assess how the projects were doing before cancellation and then afterwards they analyzed the projects to determine the factors that lead into the projects' unrecoverable failures. Though in essence a PMA, it wasn't possible for the researchers to follow any established PMA method properly, as the projects were disbanded after cancellation and the previous members weren't allowed to partake in the study to the necessary extent. The researchers relied mostly on documentation, their knowledge, and few interviews, which sets their methodology somewhat apart from usual PMA. It does show however, that PMA methods can be used flexibly and outside their original use case. Changes and adaptions are common even when PMA is used in the original way as can be surmised from the many different existing methods. Beyond these kinds of studies there aren't many publications of PMA results, though these studies aren't necessarily publicizing the PMA results either, only mentioning that the researchers in the studies came to their conclusions with the information they obtained from doing PMA.

5.1. Game PMA reports

As brought up earlier, game developers have made an exception and publicized PMA reports for anyone to read. The reports share information about the project with the focus on describing successes, failures, and how the project went from the developers' perspectives. Unfortunately, the reports don't explain how the analysis was conducted or how the report was drawn. Since the goal of PMA is to help in understanding what happened during the project and why things happened the way they did, one might imagine proper PMA was conducted to some degree at least. However, some of the oldest PMA reports predate the research on lightweight PMA methods, which also brings to question what methodologies game developers have been practicing all this time.

As noted by Myllyaho *et al.* [2004], game development PMA reports generally follow the Open Letter Template structure, which is described in their paper as well as in a paper by Collier *et al.* [1996]. The structure appears to be a good fit for reporting the results of PMA, as identifying good and bad experiences as well as their causes is generally part of the goal. The template suggests identifying the good, the bad, and the ugly. The "good" summarizes things that went well or were surprisingly beneficial. The "bad" indicates issues that came up, practices that failed and what was identified to have led the development team into the problems. The "ugly" refers to the most critical issues or things that were recognized as improvement opportunities to tackle. The addition of "ugly" is in a sense just an extension of the "bad". The reports are more skewered towards learning from mistakes, though this may be a good fit for game development which is plagued by various issues. However, it is common for reports to drop the “ugly” section or to discuss the future improvement in a conclusions part of the report instead while the “good” and the “bad” exist as lists of good and bad experiences. The two lists themselves can also contemplate on the matters that would originally be left for the “ugly” part, so as a section it hasn’t been as popular as the “good” and the “bad” are.

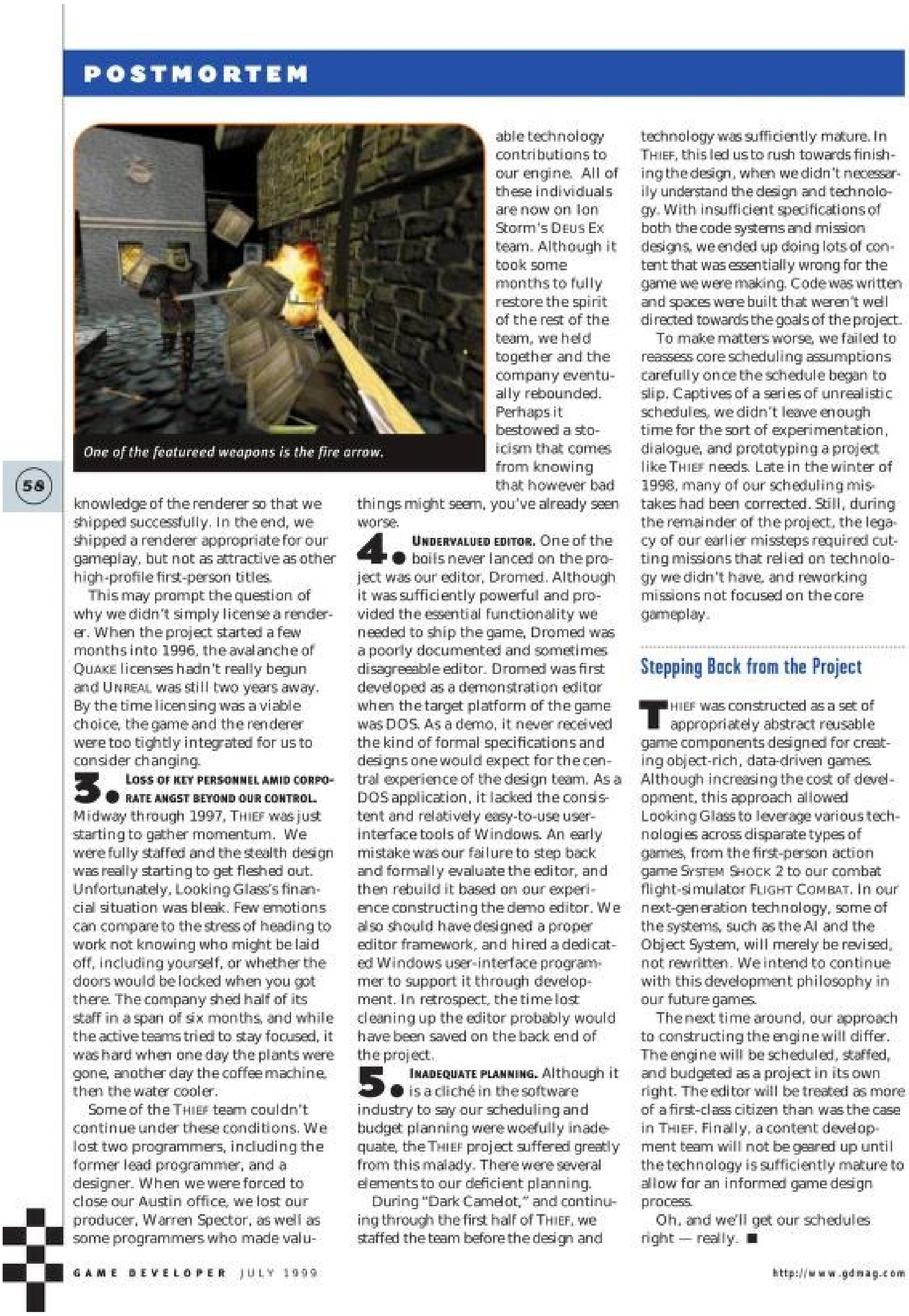
While the previous format is generally used in the publicized PMA reports, there may be other formats used in game development PMA as well. For example, Chandler’s [2013] vision for the PMA report is a more concise one. She suggests writing a short report based on the notes from PMA, but the focus for Chandler is the lessons learned. There is no mention of repeating the goods and the bads that were discovered during the analysis, so the report is much more focused, and furthermore, it’s suggested to limit those lessons to a maximum of five to avoid making the new changes too much of a hassle to implement. Chandler [2013] also suggests including some of the experiences that led to the lesson learned to help convey the value of the lesson.

To better elaborate on what a PMA report could look like here is a breakdown of one for the development of the game “Thief: The Dark Project” [Leonard, 1999], which originally appeared in The July issue of Game Developer magazine in 1999. The report starts with some introductory thoughts about how the project went. In this case the beginning is rather strongly worded: “Thief: The Dark Project is one of those games that almost wasn’t.” While this and the rest of the introduction may be an attempt to draw the reader in and the whole report may be more designed to be a game related content piece rather than a traditional PMA report, the report does continue with a description of some of the issues the project faced. Afterwards, there is a longer section to explain the basics of the game, developer team and the project’s history.

Majority of the report, however, is reserved for the next two sections: “What Went Right” and “What Went Wrong”. Figure 6 will feature the lists for what is discussed in these sections in the discussed PMA report to show how the topics look in practice, though they can be hard to fully comprehend without the further explanations that the articles themselves have. For more details on how an issue might be discussed in these kinds of PMA reports, refer to Figure 7.

What Went Right	What Went Wrong
1. Designing data-driven tools	1. Trouble with the AI
2. Sound as a game design focus	2. An uncertain renderer
3. Focus, focus, focus	3. Loss of key personnel amid corporate angst beyond our control
4. Objectives and difficulty	4. Undervalued editor
5. Multiple narrow-purpose scripting solutions	5. Inadequate planning

Figure 6, what went right and wrong in Thief: The Dark Project postmortem [Leonard, 1999].



POSTMORTEM

One of the featureed weapons is the fire arrow.

58

knowledge of the renderer so that we shipped successfully. In the end, we shipped a renderer appropriate for our gameplay, but not as attractive as other high-profile first-person titles.

This may prompt the question of why we didn't simply license a renderer. When the project started a few months into 1996, the avalanche of QUAKE licenses hadn't really begun and UNREAL was still two years away. By the time licensing was a viable choice, the game and the renderer were too tightly integrated for us to consider changing.

3. LOSS OF KEY PERSONNEL AMID CORPORATE ANGST BEYOND OUR CONTROL. Midway through 1997, THIEF was just starting to gather momentum. We were fully staffed and the stealth design was really starting to get fleshed out. Unfortunately, Looking Glass's financial situation was bleak. Few emotions can compare to the stress of heading to work not knowing who might be laid off, including yourself, or whether the doors would be locked when you got there. The company shed half of its staff in a span of six months, and while the active teams tried to stay focused, it was hard when one day the plants were gone, another day the coffee machine, then the water cooler.

Some of the THIEF team couldn't continue under these conditions. We lost two programmers, including the former lead programmer, and a designer. When we were forced to close our Austin office, we lost our producer, Warren Spector, as well as some programmers who made valuable technology contributions to our engine. All of these individuals are now on Ion Storm's DEUS EX team. Although it took some months to fully restore the spirit of the rest of the team, we held together and the company eventually rebounded. Perhaps it bestowed a stoicism that comes from knowing that however bad things might seem, you've already seen worse.

4. UNDERVALUED EDITOR. One of the boils never lanced on the project was our editor, Dromed. Although it was sufficiently powerful and provided the essential functionality we needed to ship the game, Dromed was a poorly documented and sometimes disagreeable editor. Dromed was first developed as a demonstration editor when the target platform of the game was DOS. As a demo, it never received the kind of formal specifications and designs one would expect for the central experience of the design team. As a DOS application, it lacked the consistent and relatively easy-to-use user-interface tools of Windows. An early mistake was our failure to step back and formally evaluate the editor, and then rebuild it based on our experience constructing the demo editor. We also should have designed a proper editor framework, and hired a dedicated Windows user-interface programmer to support it through development. In retrospect, the time lost cleaning up the editor probably would have been saved on the back end of the project.

5. INADEQUATE PLANNING. Although it is a cliché in the software industry to say our scheduling and budget planning were woefully inadequate, the THIEF project suffered greatly from this malady. There were several elements to our deficient planning.

During "Dark Camelot," and continuing through the first half of THIEF, we staffed the team before the design and technology was sufficiently mature. In THIEF, this led us to rush towards finishing the design, when we didn't necessarily understand the design and technology. With insufficient specifications of both the code systems and mission designs, we ended up doing lots of content that was essentially wrong for the game we were making. Code was written and spaces were built that weren't well directed towards the goals of the project.

To make matters worse, we failed to reassess core scheduling assumptions carefully once the schedule began to slip. Captives of a series of unrealistic schedules, we didn't leave enough time for the sort of experimentation, dialogue, and prototyping a project like THIEF needs. Late in the winter of 1998, many of our scheduling mistakes had been corrected. Still, during the remainder of the project, the legacy of our earlier missteps required cutting missions that relied on technology we didn't have, and reworking missions not focused on the core gameplay.

Stepping Back from the Project

THIEF was constructed as a set of appropriately abstract reusable game components designed for creating object-rich, data-driven games. Although increasing the cost of development, this approach allowed Looking Glass to leverage various technologies across disparate types of games, from the first-person action game SYSTEM SHOCK 2 to our combat flight-simulator FLIGHT COMBAT. In our next-generation technology, some of the systems, such as the AI and the Object System, will merely be revised, not rewritten. We intend to continue with this development philosophy in our future games.

The next time around, our approach to constructing the engine will differ. The engine will be scheduled, staffed, and budgeted as a project in its own right. The editor will be treated as more of a first-class citizen than was the case in THIEF. Finally, a content development team will not be geared up until the technology is sufficiently mature to allow for an informed game design process.

Oh, and we'll get our schedules right — really. ■

GAME DEVELOPER JULY 1999

http://www.gdmag.com

Figure 7, An excerpt of the last page of the Thief: The Dark Project postmortem [Leonard 1999] retrieved from www.gdcvault.com/gdmag.

Furthermore, there is a short final section, also seen in Figure 7, for summarizing the report and having a final bit of discussion. The potential future adaptions that could mitigate some of the issues that the project faced may also be elaborated in the final section if they weren't already discussed.

The previous report shows a common format for the reports available at Game Developer, though there is some amount of variety in the PMA reports found on Game Developer. A number of "postmortems" use a video or a live performance rather than a written report. One such postmortem was shown for the game "Death Stranding" in the Game Developers Conference event in 2021 [GDC, 2021]. However, the PMA report in question is fundamentally different from traditional PMA reports as the report is focused on the aspect of AI (artificial intelligence) agents in game, which was a significant challenge in the game's development. This means the report is about how a game feature was developed, which may interest game developers, but it means that the report doesn't examine the development as a whole. While the idea of performing a PMA focused on a specific aspect of development has been mentioned in literature[Stålhane *et al.* 2003], some game developers have taken the concept further and turned "postmortems" into opportunities to discuss game features and their development.

The number of publicly released PMA reports that focus on analyzing the project's development is not easy to count. Some of the difficulties come from the unconventional formats, such as publicly presented reports mentioned previously, that can deviate too much from traditional PMA. The latest papers using PMA reports, such as Lehtonen *et al.* [2020], who had 347 reports, tend to use a dataset that features hundreds of reports. Of the 347 reports, 218 were from *www.gamedeveloper.com*, which was called Gamasutra at the time of the study, and 129 from *www.gamecareerguide.com*. Currently in the latter site, there appears to be 153 articles marked as "postmortem" [Game Career Guide, 2022], though there is no guarantee that each one would fulfill the standards for research. Game Developer makes things more difficult by not having a separate category for PMA reports. The website gives 2090 results for searching "postmortem" [Game Developer, 2022], but a notable amount of these are links to presentations from Game Developers Conference, reports on the development of specific game features or other content. The number of posts that are reports usable for research is only a fraction, likely close to 218, which was mentioned by Lehtonen *et al.* [2020]. Based on how many reports researchers use and the data on these sites, which are the commonly used in research, it's likely that the number of readily available written PMA reports that researchers can easily access in 2022 is close to 400.

5.2. Studies on game PMA reports

PMA reports have been used as research data in game development research for about a decade and it has produced papers, such as Petrillo *et al.* [2009] and Washburn *et al.* [2016], which have contributed to game development research. Despite the research on PMA reports, there is limited understanding about how the findings on PMA reports are consistent with the game development related research findings based on other sources of information. In this section, the use of PMA reports in research will be analyzed and discussed.

We use the work by Murphy-Hill *et al.* [2014], which uses surveys and interviews for data gathering, as a basis for discussing findings of research on game development. Murphy-Hill *et al.* [2014] examine how game development is different from other software development and the results can be referred to when examining game development research. The discussion will also feature several other research papers on game development, which have made datasets from PMA reports. Including every paper was not feasible or necessary for this discussion, so papers with different kinds of data sets from different times were selected to showcase the available research.

Murphy-Hill *et al.* [2014] focus on finding out how game development differs from software development which is a common topic in game development research in general and the extensive paper shares many subtopics from other research. The research subjects include both game and non-game software developers from Microsoft so the comparisons should show real differences within otherwise similar environment. Murphy-Hill *et al.* [2014] discusses various topics with the help of the data they collected, from which the list in Figure 8 was compiled from.

List A	Noted difference	Explanation
1.	**Fun requirement**	The only real requirement of games is to be “fun”, so requirements specification is more subjective and unclear than in non-game software development.
2.	**Design and planning resistance**	Game developers feel that design and planning effort is wasted due to point 1.
3.	**Code reuse in game development**	The culture of reusing code is different as pieces of code often do not work in a different game that is programmed differently, while on the other hand, tools and engines tend to be reused a lot.
4.	**Testing games**	Testing games focuses on play testing and automated testing is often skipped as both gameplay and game development are too chaotic.
5.	**Technical requirements**	Technical requirements of games cause complications because programmers don’t necessarily understand the game’s art assets and likewise artists don’t always understand the technical requirements.
6.	**Unskilled management**	Game programmers can suffer if the project management doesn’t understand the technical side of software engineering and people in game management positions come from more varied backgrounds.
7.	**Cowboy programmers**	Individual “cowboy” programmers that can “save the day” when problems arise are important in game development due to points 1, 5 and 11, though reliance on skilled individuals can hide other problems.
8.	**Software engineering process resistance**	Game developers oppose software engineering processes, because they are not believed to work for game development.
9.	**Agile game development**	Despite point 8, agile practices are somewhat popular within game development, though sometimes “agile” is only a cover up for lack of practices.
10.	**Inflexible deadlines**	Game developers feel that their deadlines are very inflexible, more so that non-game software development.
11.	**Technical knowledge requirements**	Game programmers may need very technical knowledge for the sake of optimization and developing complex game mechanics.
12.	**Interdisciplinary teams**	Game development is interdisciplinary, which increases the need for conflict resolution skills.
13.	**Prestige**	Being a game developer can be prestigious, especially for popular and prize-winning games.
14.	**Perception of games**	While game development was generally felt to be meaningful, the individual developer’s views on games can affect how they feel about their job.
15.	**Demanding work**	Game development is sometimes mentioned to be more physically demanding, either because of longer work days or for otherwise more intensive work environment.

Figure 8, list A – differences in game development, summarized based on the study by Murphy-Hill et al. [2014].

One notable author who has used PMA reports in research is Petrillo, who co-authored one of the earliest research papers utilizing PMA report data [Petrillo *et al.* 2009] as well as several other papers that have benefitted from PMA reports wrote such as Politowski *et al.* [2016] and Politowski *et al.* [2021]. The goal of the paper [Petrillo *et al.* 2009] was to identify common problems in game development from analyzing 20 different PMA reports from *www.gamasutra.com*, which was the predecessor of *www.gamedeveloper.com*, referred to elsewhere in this thesis. The identified problems and their occurrence rate in the paper [Petrillo *et al.* 2009] are listed in the Figure 9.

List B	Identified problem	Occurrence rate
1.	Unrealistic scope	75%
2.	Feature creep	75%
3.	Cutting features	70%
4.	Design problems	65%
5.	Delays	65%
6.	Technological problems	60%
7.	Crunch time	45%
8.	Lack of documentation	40%
9.	Communication problems	35%
10.	Tool problems	35%
11.	Test problems	35%
12.	Team building	35%
13.	Number of defects	30%
14.	Loss of professionals	25%
15.	Over budget	25%

Figure 9, list B – problems in game development, adapted from Petrillo et al. [2009].

Petrillo *et al.* [2009] note that many of the issues they found were similar to issues in software development, however, comparing these problems to list A from Murphy-Hill *et al.* [2014], shows that many of the issues relate to differences that were highlighted.

Another study analyzing game development was made by Wasburn *et al.* [2016], who focused on compiling what went well and what went wrong in 155 different PMA reports, which were also from *www.gamasutra.com*. The identified categories, with the separate occurrence rates for right and wrong in order of most total occurrences can be seen in Figure 10.

List C	Category	Right	Wrong	Total
1.	Game design	50%	22%	72%
2.	Development process	43%	24%	67%
3.	Team	40%	13%	53%
4.	Art	39%	14%	53%
5.	Tools	34%	16%	50%
6.	Schedule	19%	25%	44%
7.	Features	25%	16%	41%
8.	Obstacles	3%	37%	40%
9.	Gameplay	27%	12%	39%
10.	Testing	25%	14%	39%
11.	Scope	17%	12%	29%
12.	Marketing	15%	11%	26%
13.	Budget	10%	8%	18%
14.	Feedback	14%	4%	18%
15.	Creativity	14%	2%	16%
16.	Documentation	4%	7%	11%
17.	Hardware	5%	5%	10%
18.	Community support	8%	1%	9%
19.	Piracy/licensing	5%	4%	9%
20.	Publisher relations	5%	2%	7%
21.	Product evolution	2%	1%	3%

Figure 10, list C – what went right and wrong in game development, adapted from Washburn et al. [2016].

An interesting set of data has also been made by Lehtonen *et al.* [2020], who investigated requirements engineering in game development. One of the things they did in their study was mining occurrence rates for specific keywords from 347 PMA reports from *www.gamasutra.com* and *www.gamecareerguide.com*, which can be used to show common or uncommon some topics are. The study is a bit different in focus compared to the earlier three, but the keyword list in particular shows some interesting things and the utilization of best practices in software development does still link to the other studies. The list of keywords that were examined in the study and that had over 1% occurrence can be seen in Figure 11.

List D	Keyword	Occurrence	#	Keyword	Occurrence
1.	Development	94.41%	15.	Discipline	13.82%
2.	Feature	81.76%	16.	Transition	13.24%
3.	Process	79.41%	17.	Emotional	11.47%
4.	Document	67.06%	18.	Formal	10.29%
5.	Schedule	60.59%	19.	Feature-creep	9.71%
6.	Production	60.29%	20.	Scrum	9.71%
7.	Communication	42.94%	21.	Agile	7.94%
8.	Management	42.06%	22.	Overtime	7.35%
9.	Scope	39.41%	23.	Specification	6.47%
10.	Method	31.76%	24.	Game-design-document	5.59%
11.	Requirement	27.65%	25.	Creep	5.59%
12.	Engineer	27.06%	26.	Engineering	4.41%
13.	Crunch	25.88%	27.	Estimation	2.35%
14.	Pre-production	16.47%	28.	Backlog	1.76%

Figure 11, list D – keywords and their occurrence rate in PMA reports from Lehtonen et al. [2020].

5.3. Examining the commonalities between the research papers

One huge intersection of problems is **tough schedules** for game development, which shows up in Murphy-Hill *et al.* [2014] as A-10 (inflexible deadlines) and A-15(demanding work). In the study by Petrillo *et al.* [2009], potential causes for scheduling problems appear in the forms of B-1(unrealistic scope) and B-2(feature creep), while many other problems that may be the results of this problem also exist, such as B-3(cutting features), B-5(delays), B-7(crunch time) and B-15(over budget). Furthermore Washburn *et al.* [2016] calculate C-6(schedule) to have 25% occurrence rate as something that went wrong, making it the second most common thing to be wrong, while they also note 19% occurrence for schedule as something that went right, perhaps showing that a well working schedule is something game developers feel worth noting in a PMA report. Additionally, C-11(scope) and C-13(budget) shows up in the paper as well, though they have fairly balanced occurrence rates that slightly favor success. Word occurrences from Lehtonen *et al.* [2020] also show schedules as common topics with D-5(schedule) occurring in 60.59% of the papers while the related D-9(scope), D-13(crunch), D-19(feature-creep) and D-22(overtime) also have 39.41%, 25.88%, 9.71% and 7.35% occurrence rates.

Tough schedules are generally noted to be a common issue in software development as well, but it may be even more of an issue within game development as many methods used to alleviate it in software development are not as popular in game development. There are also many other potential reasons such as more inflexible deadlines, caused by, for example, targeting holiday sales, and inexperience in estimating required time for development. Murphy-Hill *et al.* [2014] highlight the inflexible deadlines as the big reason for tough schedules and “crunch time”, which refers to extra work beyond the normal working hours. On the other hand, PMA report studies show that the writers of the reports blame inability to estimate the workload much more than the inflexibility of the schedule [Petrillo *et al.* 2009, Wasburn *et al.* 2016, Lehtonen *et al.* 2020]. These are to some degree, different sides of the same coin as the initial schedule would be made based on the unfortunately inaccurate estimate, but the focus ends up on different notes between the research papers. All in all, the research consistently suggests that there are multiple reasons causing scheduling problems and excessive work times in game development.

One thing that gets brought up time after time is the fundamentally different approach to **design practices** that is inherent to game development. Instead of focusing on making a functional piece of software, developers need to create a piece of entertainment. This makes the creative part of development much more subjective than in traditional software development. While Murphy-Hill *et al.* [2014] talk about software design in game development, they bring up A-1(fun requirement) as the main reason for poor design practices. As “fun” is the basis of design, they believe that developers perceive game requirements as arbitrary and chaotic, which makes developers feel that their design effort is wasted, leading them into poor practices.

Game design is mentioned by Petrillo *et al.* [2009], though only as B-4(design problems), which refers to bad design practices and choices. With 65% occurrence it is one of the more common problems. Game design gets highlighted by Wasburn *et al.* [2016] as well, though in a different light, as they have a separate C-2(development process) category, which leaves the C-1(game design) category with more game related things. The game design category is the most common reported category according to their research, with 72% of PMA reports featuring it. It had 50% occurrence as a good thing, which according to the researchers was because developers felt that they made good design choices. On the other hand, 22% reported it as something that went wrong as they regretted some choices in design later. Furthermore, Wasburn *et al.* [2016] also mention other similar categories C-7(features), C-9(gameplay) and C-15(creativity), all of which show up more often as something that went right, with C-15(creativity) in particular, having 14% to 2% split in occurrences in favor of positive experiences.

It may be that PMA reports are quite likely to feature this kind of positive references to game design, as the developers want to discuss how their game ended up as it did. Design choices and creativity have a more direct link to the features in the final product than other parts of the development process and thus makes for a more interesting report. In the context of studying game development however, it might not be that important to hear about individual successes in game design.

The actual paper on **requirements engineering**, Lehtonen *et al.* [2020] does not make much note of the requirements for game development besides acknowledging that games are made to be as fun as possible. Regarding the PMA reports, it is noted that terms related to requirements engineering process hardly appear at all, indicating that game developers do not discuss them in the reports. It is unclear how much this lack is caused by the actual lack of requirements engineering and how much it is that the developers just don't mention it in the reports, but Lehtonen is convinced that this backs up the general belief among researchers that game developers are not following the accepted best practices to a large extent. This belief is also shown in the paper by Murphy-Hill *et al.* [2014] in the points A-2(design and planning resistance) and A-8(software engineering process resistance).

The study by Lehtonen *et al.* [2020] might also suggest that developers are hesitant to discuss development practices in detail during PMA reports. Many of the top keywords would suggest that development practices are discussed a lot during reports, such as D-1(development), D-3(process), D-4(document), D-6(production). D-7(communication), D-8(management) and D-10(method), which have occurrence rates in the range 32-94 percent. However, as mentioned before, some of the keywords that they were looking for did not show up at all. Furthermore, some of the more popular terms, relating to development details, such as D-14(pre-production), D-20(scrum), D-21(agile) and D-24(game-design-document), had low occurrence rates of only 16.47%, 9.71%, 7.94% and 5.59%. The actual usage rates for these practices are likely much higher than these numbers, though how much is a hard to estimate. With only general terms being popular, it may be difficult to gain deeper knowledge about development practices from PMA reports.

Furthermore, the commonality of some keywords doesn't seem to translate into the topic being common. The D-4(document) had 67.06% occurrence rate, but it doesn't see similar rates as topic in other studies. Documentation still somewhat stands up with B-8(lack of documentation) having 40% rate, but on the other hand, C-16(documentation) only has 11% total rate, suggesting perhaps that documentation and documents get mentioned on the side of other topics rather than being a popular topic itself. Murphy-Hill *et al.* [2014] did not note documentation directly as a topic in their study either, which might suggest it either doesn't differ much between game and non-game software development or that they simply didn't consider it noteworthy. Considering that documentation is related to the discussions of design, planning and software engineering processes, the topic of documentation may get overshadowed by other related discussion.

There are some other common points between research as well. For one **testing** is mentioned in A-4(testing games), B-11(test problems) and C-10(testing). Murphy-Hill *et al.* [2014] notes that high level testing, such as play testing, is favored over low-level testing like unit testing in game development, which does go against best practices in software development. In PMA report studies testing also comes up fairly often, but it is nearly always about play testing and properly utilizing the data from such tests, with low-level testing hardly being mentioned. Washburn *et al.* [2016] even give C-14(feedback) its own category despite C-10(testing) being on the same list. The category is specifically for developers utilizing feedback they receive, though both the examples from testing and feedback categories appear to refer to beta testing only. Unit testing and other more traditional software testing rarely if ever appear in game development literature, they are also entirely omitted by Chandler [2013], despite her book on game production having a whole section dedicated to testing.

Interestingly another point that would traditionally be linked to testing, defects, do get a mention in only one study: B-13(number of defects) shows up with 30% occurrence. The paper by Petrillo *et al.* [2009] has the most coverage about defects and bugs, while Murphy-Hill *et al.* [2014] also have a few mentions of bugs but no real discussion about them. Many of the references are quotes from developer interviews or PMA reports, so one might imagine it's a common point across the papers, but it receives little attention compared to other issues. Potentially Murphy-Hill *et al.* [2014] didn't feel that it was different from usual software development. On the other hand, "too many defects" could be squashed into some other category in PMA report analysis, for example into C-8(obstacles).

Another common point is talking **development tools**. Tools get touched in A-3(code reuse in game development) with the mention that tools are important and get reused a lot in game development. In PMA report studies tools also frequently get their own popular category, such as B-10(tool problems) with 35% occurrence, and C-5(tools) with 50% total occurrence rate. PMA report studies suggest that tool usage can have a big impact on development, while A-3(code reuse in game development) showed that developers consider tool reuse to be more common and important in game development.

The **interdisciplinary teams,** that game development requires, also receive attention from researchers. The focus is usually on communication issues between different professionals. The following points are at least partly about such issues: A-5(technical requirements) A-12(interdisciplinary team) B-9(communication problems) and B-12(team building). The art side of the game plays a big role, so issues with utilizing the art side can also play a big role, although interdisciplinary teams involve more than just two professions. Usually, it is the artist that get picked up by researchers as the alternative profession to programmers to showcase interdisciplinary teams. The significance of art side can also be seen from C-4(art), which has 39% and 14% rates for right and wrong making it a common topic, though generally positive. Despite the fears of issues with interdisciplinary teams, in general, it does appear that teams are much more commonly a positive aspect of game development rather than an issue as C-3(team) is one of the most common PMA report points with 40% and 13% rates for right and wrong placing nearly identically with C-4(art). Instead of fearing issues of interdisciplinary teams, Washburn *et al.* [2016]

appears to highlight the newly created team as a significant danger, as such a scenario is the given example for the category C-8(obstacles). The large variety of different “obstacles” makes it hard to estimate how significant the troubles of team forming are, though the category also manages to be the most common one to go wrong on the list with 37% occurrence rate.

Another interesting point about teams is the significance of **skilled individual programmers**. This shows up as A-7(cowboy programmers) and A-11(technical knowledge requirements) but it can also be seen in the study by Petrillo *et al.* [2009], where B-6(technological problems), B-12(team building) and B-14(loss of professionals) all touch the subject of programmers with necessary skills being a critical need. Issues with new technology, finding programmers with the right skills set and losing hard to replace individuals all tell a tale of problematic dependencies, similar to what was described by Murphy-Hill *et al.* [2014] as “cowboy programmers”. Washburn *et al.* [2016] on the other hand tells a much more positive tale of team building, as C-3(team) is three times as likely to show up in positive way. Still, “team” does contribute another 13% occurrence in things to go wrong, with the example given being team members not having the necessary experience to implement the intended features. The reliance on skilled individuals is very high and consistently shows up in game development literature, often in the form of difficulties regarding lack of skills or loss of professionals. While the mentioned “cowboy programmers” that show up and “save the day” [Murphy-Hill *et al.* 2014] might have sounded fairly romanticized or plain silly, it may be a common occurrence in some game development teams.

There are some common points with fairly small occurrence rates such as **agile development** showing up in A-9(agile game development) and in D-21(agile) with 7.94% occurrence rate. Neither Petrillo *et al.* [2009] or Washburn *et al.* [2016] bring agile up as its own issue, likely due to low occurrence. As noted before, developers appear hesitant to discuss development practices in PMA reports. An even less common point is **marketing** which only shows up as C-12(marketing) with a 26% total occurrence. Despite this occurrence rate suggesting that marketing could be a somewhat common topic, it is not mentioned by either Murphy-Hill *et al.* [2014] or Petrillo *et al.* [2009].

Not all points made by Murphy-Hill *et al.* [2014] appear in the other papers. Potentially because interviewing developers can bring up different kinds of things and because PMA reports tend to focus on the most significant things. The points of A-13(prestige) and A-14(perception of games) are interesting in their own right, but they do not seem to appear in PMA reports. The developers’ feelings about being game developers are either not brought up in PMA reports or the researchers do not note them. Also, from the A-3(code reuse in game development), only the tool reuse is talked about in other papers while the reuse of actual code is not, which is in line with the point itself, as the developers suggested that actual code is not reused much. On the other hand, A-6(unskilled management) is perhaps too specific to come out from general analysis done to PMA reports. Alternatively, such complaints could get hidden behind other categories or potentially be hidden from PMA reports all together, if said management is who gets to write the report.

Another large intersection is issues regarding **development practices**, which shows up in Murphy-Hill *et al.* [2014] as A-2 (design and planning resistance) and A-8 (software engineering process resistance). Petrillo *et al.* [2009] similarly highlight B-4 (design problems). There isn't a direct reference to development processes in list B, but many of the listed problems are likely related to the lack of development practices, particularly B-8(lack of documentation). In list C, C-2(development process) is the second most common category with 43% occurrence in what went right and 24% in what went wrong, which are interesting numbers as one might think they suggest game developers have good practices, although with 24% development process places third in most common thing to go wrong. Washburn *et al.* [2016] make further notes about development process category by listing three common things from PMA reports that felt their development process was something that went right. The success appears to stem from more time spent planning, particularly in making prototypes, as well as using an iterative development process.

While game developers didn't talk about their processes as failures very often, it is generally noted by researchers that many of the issues game developers face can be mitigated by software engineering methods that are much more adopted in non-game development. Therefore, we could say that many of the other issues are issues with development practices. While various game development related reasons cause extra issues for game developers and differentiate their work from other software development, better application of software development practices should alleviate many of the issues game developers face.

5.4. Discussion on PMA report research

Studying PMA reports appears to be a novel method of research, as the results from these research papers are consistent in their findings with each other and other studies. The papers can offer a different perspective from other kinds of research and are good for discovering interesting points from developers, though they can be limited in their topics. While there doesn't appear to be significant inconsistencies or contradictions, some minor inconsistencies could be pointed out, such as that marketing only came up in one of the papers despite it being a common topic in PMA reports according to the data in that paper. It could be guessed that marketing was not considered to be within the scope of the other studies as it is another field of study from software development. The studies have their own focuses, which makes them different from one another, but the image they form of game development appears to be uniform.

The different perspective that PMA report studies can have may be quite useful at times. Murphy-Hill *et al.* [2014] interviewed developers after conducting a preliminary survey which allowed them to collect the right information for making their points about how game development differs from non-game software development. Re-examining the points with PMA report data can show how impactful some of the points really are. For example, Murphy-Hill *et al.* [2014] suggested "rather than using automated, low-level testing, testing in games tends to be run more often at a high level, either by a human playing through the game, or as a script simulating what a human would do", however PMA reports made a much clearer point of human testing being the norm and also showed that it is a significant factor in making a successful game. On the other hand, there were no mentions of low-level testing in any of the PMA studies. Murphy-Hill *et al.* [2014] was able to find out about underlying differences of game development and how developers felt about them specifically, but the impacts of those differences can be better seen in the PMA report utilizing studies.

There are also some issues with PMA reports which may restrict their usefulness. One potential issues with the public game PMA reports is that they may focus too much on game feature related topics and avoid deep discussion on development practices. The issues may lead to parts of the PMA reports not containing useful information for researchers. The large number of reports, and topics that each report discusses, allows the reports to provide sufficient data for research despite this, though researchers may struggle to find discussion on specific topics as was the case with Lehtonen *et al.* [2020].

PMA reports offer a way to gauge how common some issues or practices are, which is how the studies presented in this thesis used the reports. However, this benefit is also an issue since this means that the same things are being repeated across different reports. Common things and trends may be easy to observe through the reports but there might not be new insights to be found. This kind of repetition is sometimes noted as a bad thing even among PMA supporters, though there appears to be different views of whether this is an issue of PMA not being done right [Kerth and Weinberg, 2013] or software industry being fundamentally bad at learning activities [Glass, 2002].

6. Survey

The main goal of the survey is to find out if PMA is a common practice in the game industry in 2022, how does PMA of today compare with the PMA methods proposed in early 2000s, and how PMA reports are used in the game industry. The survey was conducted as an online questionnaire, which was posted in social platforms used by Finnish game development professionals.

Finnish game industry was chosen over trying to target global game industry, as it was considered easier to reach more local developers and organizations, such as Neogames, which is an umbrella organization for the Finnish game industry and gave advice on conducting this survey. Neogames [2021] also publishes its own reports on Finnish game industry, which show that the Finnish game industry is quite sizable with 200 studios and 3600 people employed in 2020.

The online questionnaire was directed towards professionals from the game industry through a link posted in the Facebook groups Play Finland and IDGA Finland, as was recommended by Neogames. These groups are the largest Finnish Facebook groups for the game industry and game developers with ~8800 and ~6300 members at the time of the study. As the groups are open for anyone to join, it's impossible to estimate what percentage of members are game industry professional. The survey was conducted between 7.4.2022 and 28.4.2022, with 16 responses, which were all considered valid. The full list of survey questions and results can be found in the appendix.

6.1. Background information

While most of the questions about background information aren't essential to the goals of the survey, they help with evaluating the validity of the survey, so in this section the focus is on the demographic information on the survey participants.

As seen in Figure 12, the survey shows 25% of the developers to have 5 or less years of experience, while the survey from Neogames [2021] suggests that between 2016 and 2020, the number of employees in the Finnish game industry increased from 2750 to 3600, which would mean that about 24% of the employees had 0 to 4 years of experience. The comparison is a little off, since this doesn't account for people leaving the industry and is off by one year, but the number appears a fairly close match to the survey in this thesis, so the spread of experience among the survey participants is within reason.

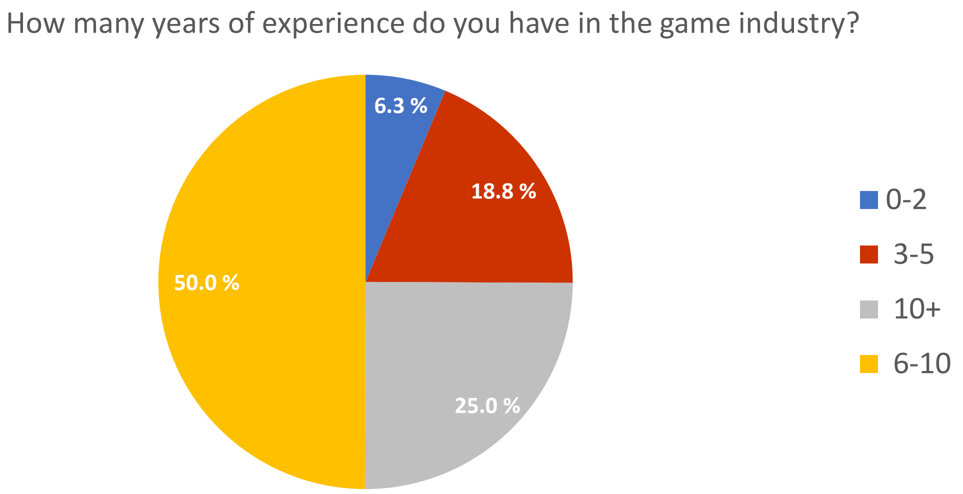


Figure 12, a survey question on game industry experience.

On the other hand, the spread of team roles appears skewered. The most popular role in Figure 13 is project manager, which 50% of the respondents picked as one of their roles. It should be safe to assume that game development teams have a different kind of spread for the roles than what is seen on Figure 13. The reason for the overrepresentation of project managers is not entirely clear, though there are a few potential reasons. First, managers are likely more knowledgeable about PMA and thus more likely to complete the survey. Another potential reason could be a biased demographic in the Facebook groups. These potential issues could not be examined further as there is no data available to determine their significance.

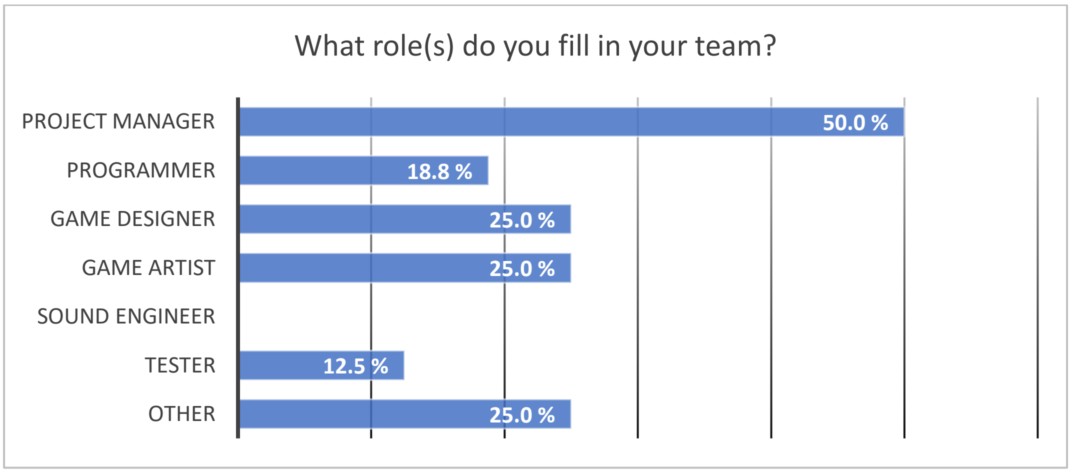


Figure 13, a survey question on development roles.

The data also shows a high representation of large companies as seen in Figure 14 where almost half the participants claimed their company to have 50 or more people. If we were to calculate an average employee count using the minimum number of each category the result would an average of 53, which, while high, is not unreasonable. The Neogames [2021] survey claimed that the number of employees in Finnish game studios was 25 on average and 8 as a median. The big difference is that Neogames surveyed companies and not employees as was done for the survey in this thesis, so the representation is not necessarily inaccurate, since companies with more employees would be more likely to appear multiple times on a survey targeting employees, making it more skewered. It is also noteworthy that the Neogames [2021] survey had the average employee count as only 20 in 2018, while the 25 number is for 2020. The trend appears to be for Finnish game companies to be growing bigger without the number of companies growing at a similar pace, which may also partly explain why the reported company sizes are on the larger side.

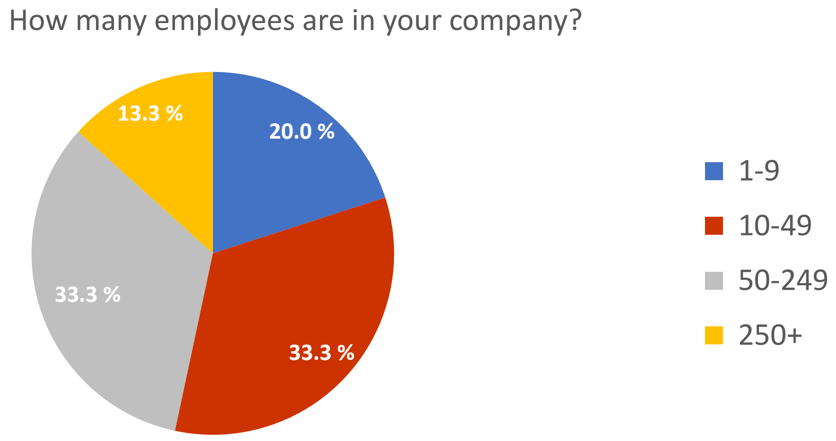


Figure 14, a survey question on company size.

Of the survey respondents, 81% have read a PMA report (Figure 15), which is a notable number considering how some researchers have been worried about reports being forgotten. Furthermore, as seen in Figure 16 on PMA report sources, 76.9% of the PMA report readers have read a publicly released report, making it the second most popular source for a report after a report from a PMA the respondent had participated in, which had a rate of 92.3%. At least among the respondents, PMA reports seem to be well utilized. While similar numbers might have come up from surveying non-game software development, it is unlikely that publicly released reports would have scored high, or have been selected at all, since public PMA reports are a game development phenomenon.

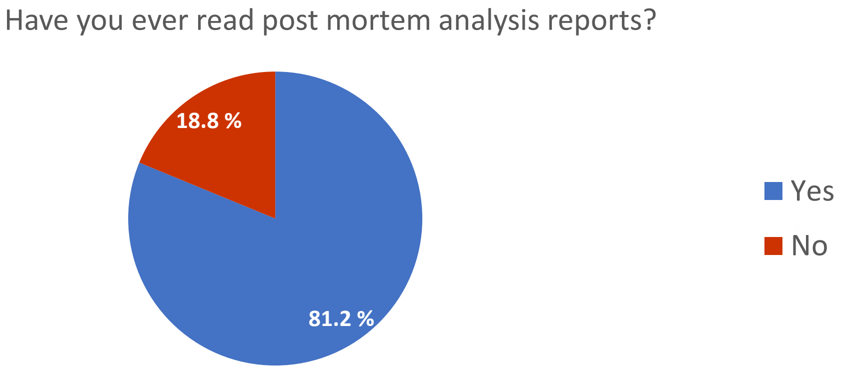


Figure 15, a survey question on reading PMA reports.

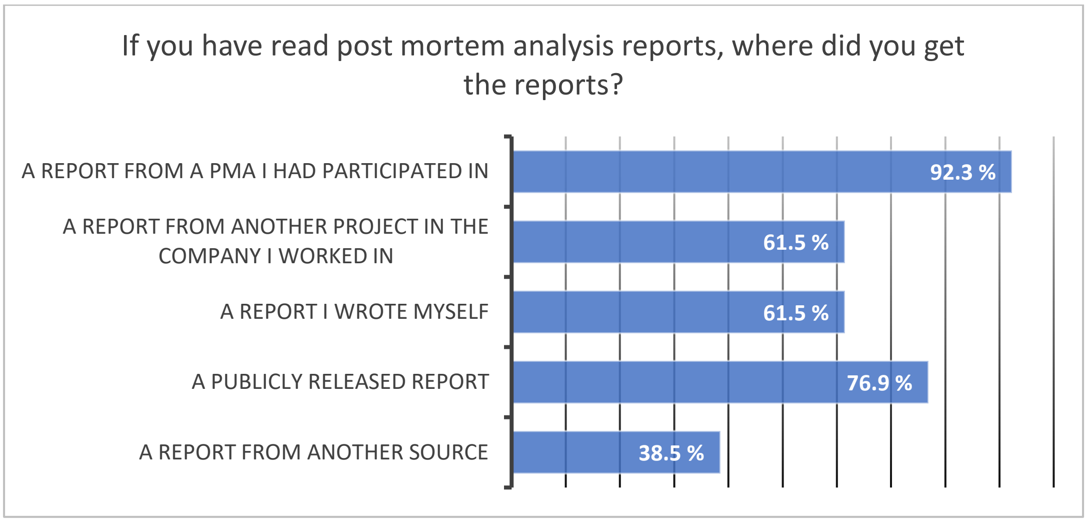


Figure 16, a survey question on PMA report sources.

Interestingly a 61.5% share of readers had apparently written a report themselves as they marked themselves as a source for a PMA report (Figure 16), though this may be linked to the high amount of project managers answering the survey. In a separate examination of the people who identified themselves as project managers 87.5% had written a report themselves, so they make up the clear majority of that response.

In many PMA methods, writing the report would fall on the facilitator of the PMA, who would be a project outsider from outside the company or a department outside the development team. The project management may help the facilitator but is not generally placed as either the facilitator or the report writer, though involving project management in PMA isn't entirely alien in PMA literature. For example, the method of Collier *et al.* [1996], gives management personal many responsibilities during the PMA process, and only uses facilitators to help with the meetings.

6.2. Experiences with PMA

Answers to the questionnaire indicate that role of facilitator falls upon the project manager as Figure 17 shows that 83.3% of the respondents say that a person from the project team leads the meeting. It was not explicitly stated who in project team does it, but it would be reasonable to assume it is the project manager or equivalent person. The previously mentioned high percentage of project managers among survey respondents, among who a high percentage had written a PMA report, also supports the possibility of project managers being in charge of the PMA.

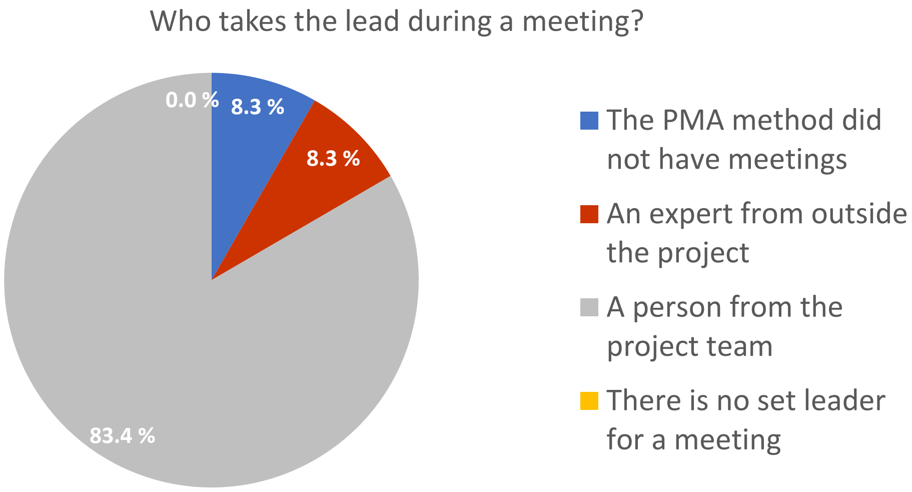


Figure 17, a survey question on who acts as the PMA meeting's facilitator.

In Figure 18, the responses show that 75% had taken part in PMA, which is fairly large number, though without data on PMA usage from other software developers, it's not possible to make comparisons.

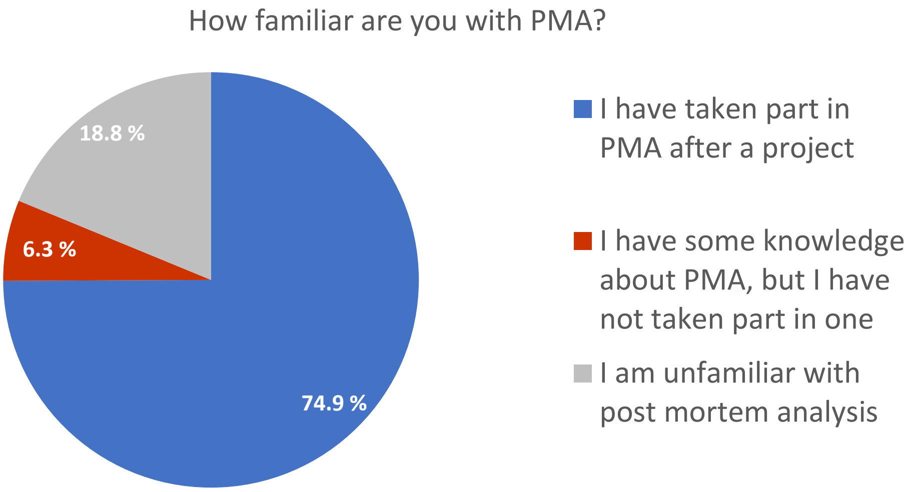


Figure 18, a survey question on PMA participation.

PMA does not appear to be a practice that every project receives. Figure 19 shows 41.6% of the respondents estimating that less than a third of projects had one and 16.7% estimating over two thirds had a PMA, there is a fair amount of variance, but the trend seems to be to only analyze a selection of projects. As mentioned previously in this thesis, there is no well-established belief on how often PMA should be done but more often is usually what researchers seem to want. Limiting PMA to selected cases seems to be how it's done in the game industry.

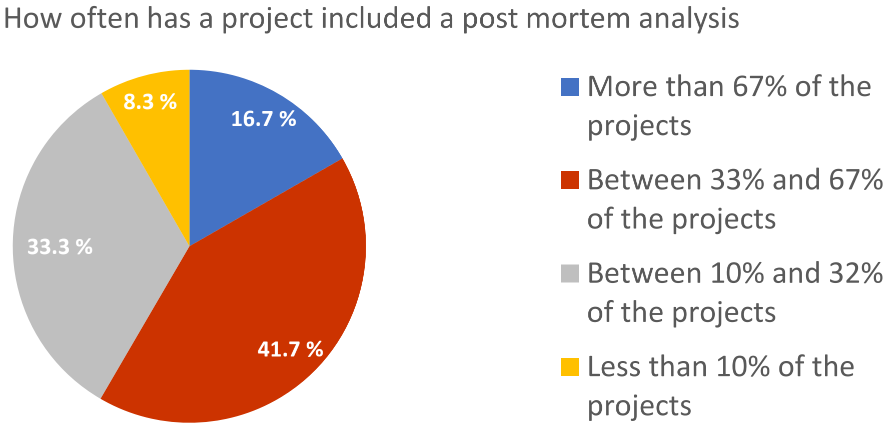


Figure 19, a survey question on PMA frequency.

How much time is spent in meetings is another noteworthy data point on how PMA is used is. Meetings tend to play a crucial role in PMA as many methods expect most of the work to happen during them. The time spent in meetings according to the survey is not on the high end, rather both 0-3 hours and 4-9 hours options received 41.7% of the vote each as seen in Figure 20. This time consumption suggests that a single meeting could be sufficient, though perhaps a whole day of meetings or a few shorter meetings during a few days would also fit in the 4-9 hour option.

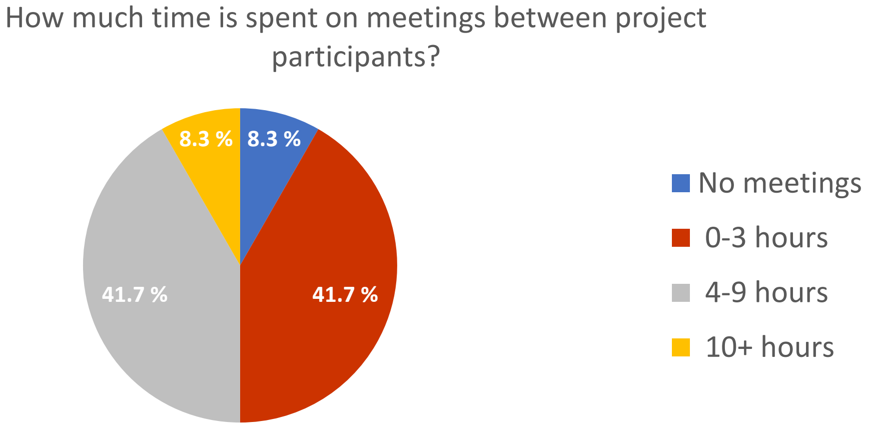


Figure 20, a survey question on PMA meeting time usage.

The popular time numbers are close to the Lightweight method as Stålhane *et al.* [2003] gave an estimate of 4 hours for the PMA session and another hour-long meeting for review. The 0-3 hours could be seen to undercut that suggestion by a bit but going lower isn't uncommon in later literature for the lightweight PMA method. Notably, while Chandler [2013], who wrote about PMA in game development projects, did not specify how long PMA should last. Instead, Chandler [2013] was supportive of all kinds of methods of different intensities, even including PMA meetings that would last under 2 hours, which would make her book match the results of this survey question. The 4-9 hours category on the other hand may include other kinds of arrangements than a single PMA session, but it would be rather short for some of the other PMA methods such as the ones from Collier *et al.* [1996] and Kerth and Weinberg [2013], which would fall in the 10+ category if the instructions were accurately followed. Timewise PMA reports in game development appear to be very lightweight.

The survey questions about PMA reports did not produce very surprising answers. Writing reports was a popular part of PMA with 75% support (Figure 21) and the most popular target of the reports in Figure 22 is the management staff or other department in the company with a 100% vote. Participants of the PMA also received a 90% in the survey, so they may also be a noteworthy target. Both are common targets for PMA reports in literature, though management is perhaps the more common one to focus on. There were other options for this question: outsiders and public release, but neither was picked a single time. These do not appear in literature much, so it's not too surprising that they are not in favor, but it shows that survey participants treated PMA reports as they have traditionally been treated and not in the exceptional ways that some game developers have treated them. There was a further question on how the respondents felt about the idea of releasing PMA reports publicly, which received a 50-50 (Figure 23) split on opinions on whether there were issues doing so or not, so that alone doesn't explain the lack of interest in doing a public PMA report release.

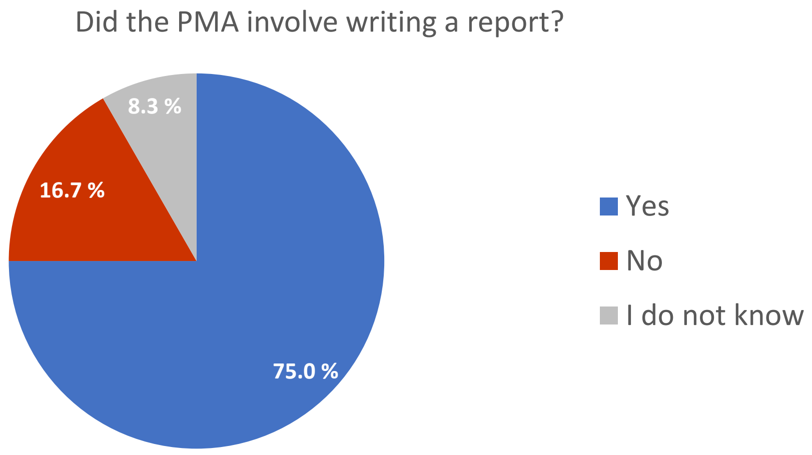


Figure 21, a survey question on whether PMA includes a written report.

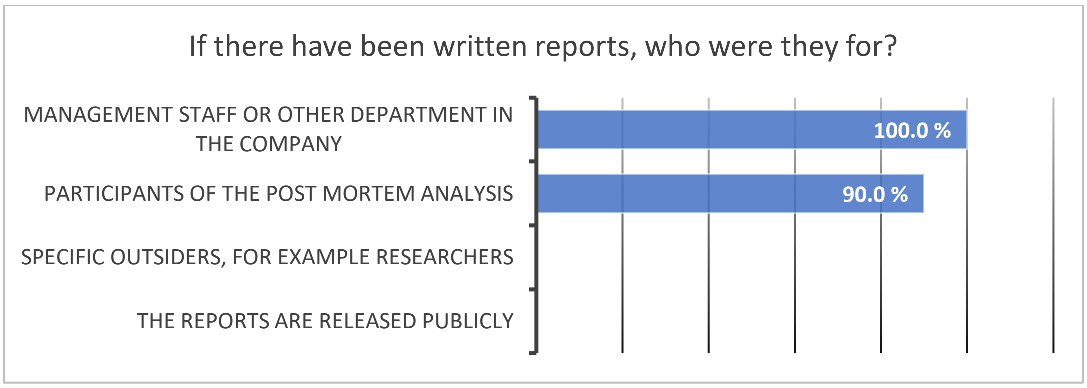


Figure 22, a survey question on PMA reports' target audience.

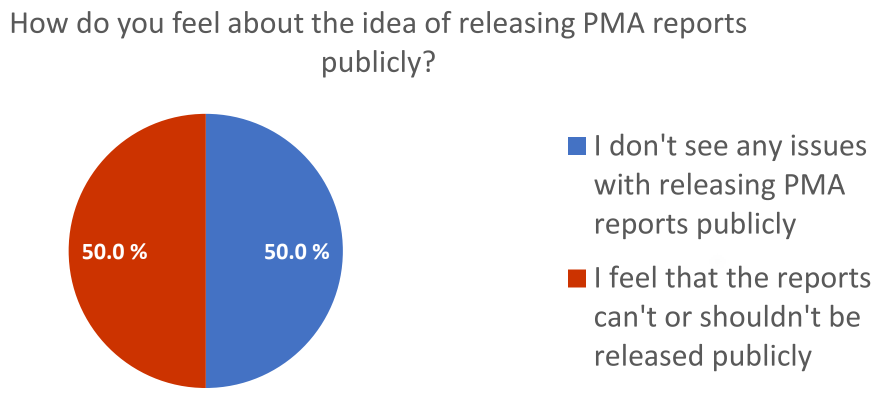


Figure 23, a survey question on releasing PMA reports publicly.

Interestingly, there are some points suggesting that PMA should be done multiple times rather than the traditional one time. It was mentioned by a survey participant ("…Retrospectives and yearly "post-mortems" arranged mid-production are also very useful…") and also by Chandler [2013]. Long game development cycles also may require a different approach since the post-release development could continue building up the game for years meaning there is nothing postmortem about doing analysis after the initial release. Such an approach might be the beginning of a new method between PMA and agile retrospectives or a fundamental change to PMA.

6.3. Experiences with other retrospective methods

A section of the survey was dedicated to other retrospective practices, as they have many things in common. Retrospection has become widely adopted practice in software development and it appears to have been carried over to game development quite well. Every respondent said that they had participated in such practices. PMA and other retrospective method sections had largely the same questions, though the answers were less surprising. Figure 24 shows that 50% of the people estimating that over two thirds of the project had retrospective methods applied making it more common than PMA.

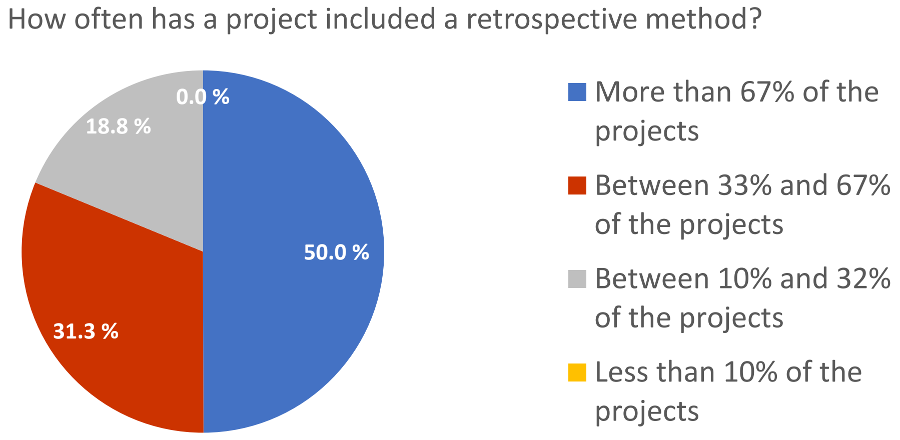


Figure 24, a survey question on how common retrospective practices are.

While the method of doing PMA in game development seemed to at times differ notably from what is described in PMA literature, the method of retrospection seems much more in line with its literature. Figure 25 shows retrospective methods to be practiced about once a month according to 43.8% and less often by 37.5%. There isn't an exact interval that should be followed, but a month is sometimes given as an example, so game developers don't seem to be too off with this, though they do not show a very clear preference.

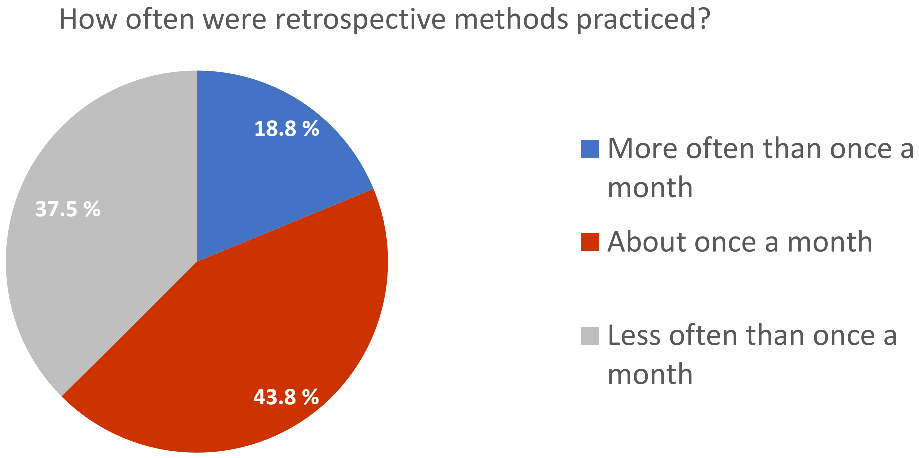


Figure 25, a survey question on the frequency of retrospective practices.

On the other hand, Figure 26 shows a clear answer as 93.8% of the respondents said that a single retrospective meeting would take 0-3 hours. Figure 27 also shows a favored answer as 87.5% said that a person from the project team lead the meeting, with the other 12.5% saying that there is not set leader.

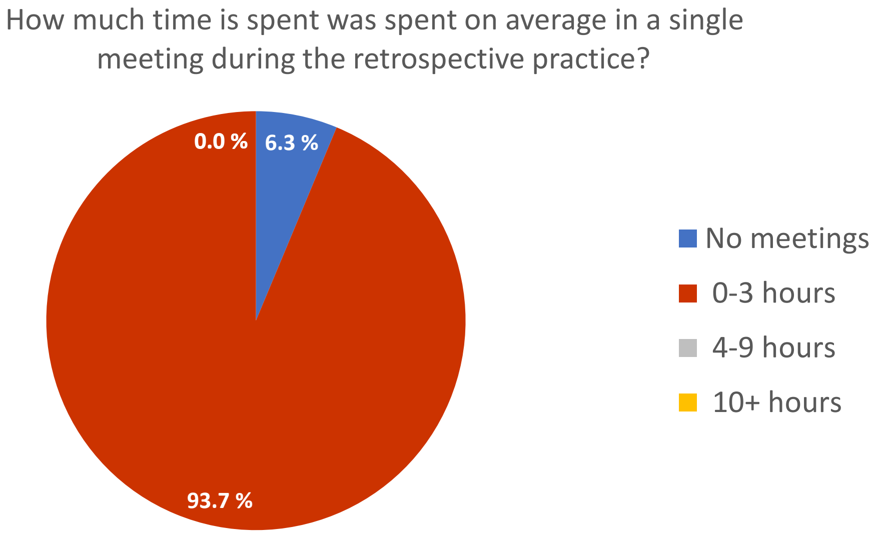


Figure 26, a survey question on the retrospective practice meeting time usage.

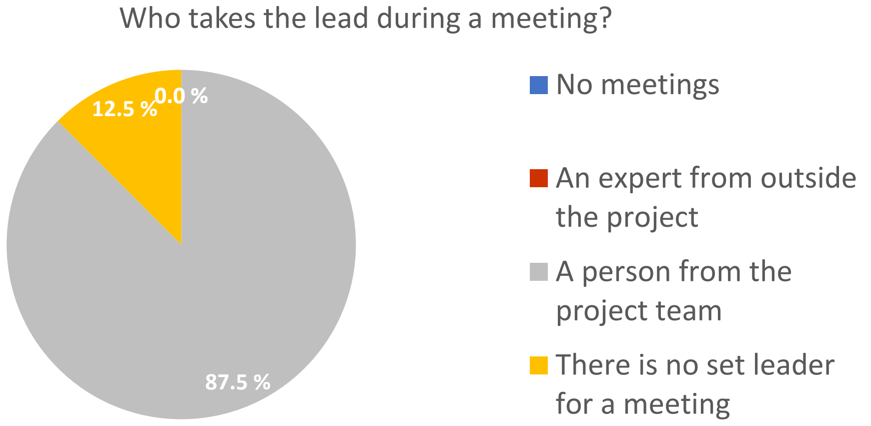


Figure 27, a survey question on who acts as the retrospective meeting's facilitator.

The focus on retrospective methods in generally on the development team improving their development methods, so writing a document for outsiders, which is common in PMA is not of great importance. Still, Figure 28 shows that for 50% of the respondents' reports had been made from the results of the retrospective methods. This could perhaps refer to an internal document for the participants to help with assessing or changing their development methods.

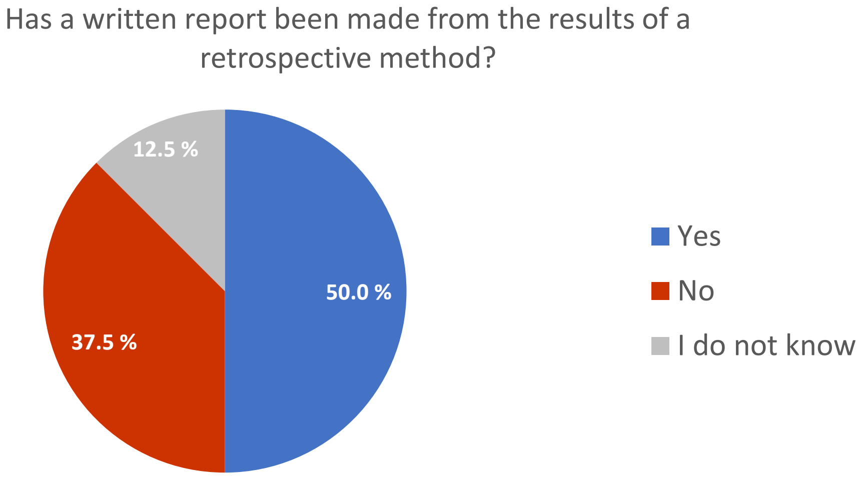


Figure 28, a survey question on whether retrospective method included a written report.

The following question on who the report was written for (Figure 29) would seem to support the previous assumption as well. The most voted answer is participants of the retrospective method with 88.9% but the choice of company management or other department also received 55.6% vote. Reports are less common than in PMA, but the numbers still suggest that similar report-based communications may occur in other retrospective methods as well. Of course, the content of such reports could be very different, they are likely more centered on small changes the development team wants to make rather than analyzing the project.

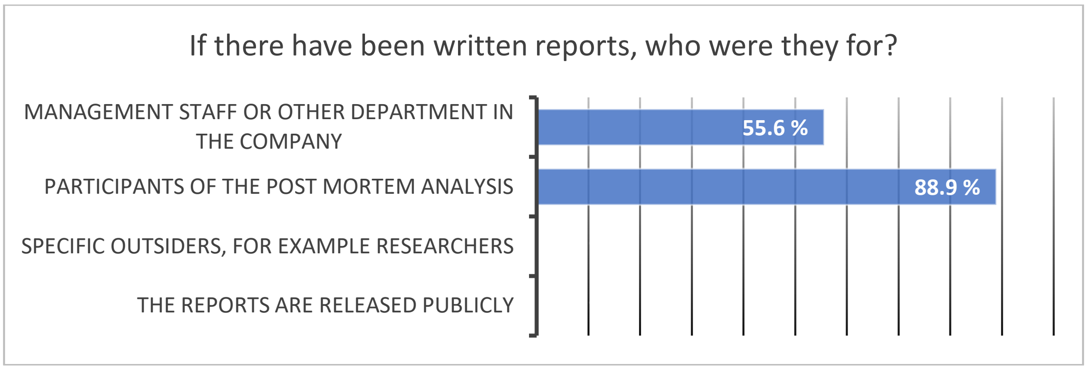


Figure 29, a survey question on retrospective reports' target audience.

6.4. Perception of retrospective methods

The perception of PMA (Figure 30) and retrospective practices (Figure 31) in the survey were both very positive, over 50% scored both of them a 4 or 5 on the scale of 1 to 5, and PMA didn't receive a score under three. Despite the seemingly good scores, some doubts and worries were expressed in the optional written section for PMA. Comments included "Varies a lot how they're done and how useful they are" and "Personally, post mortems usually just end up listing quite obvious things that seldom lead to any concrete benefit in the future", which may indicate some issues with how PMA is used.

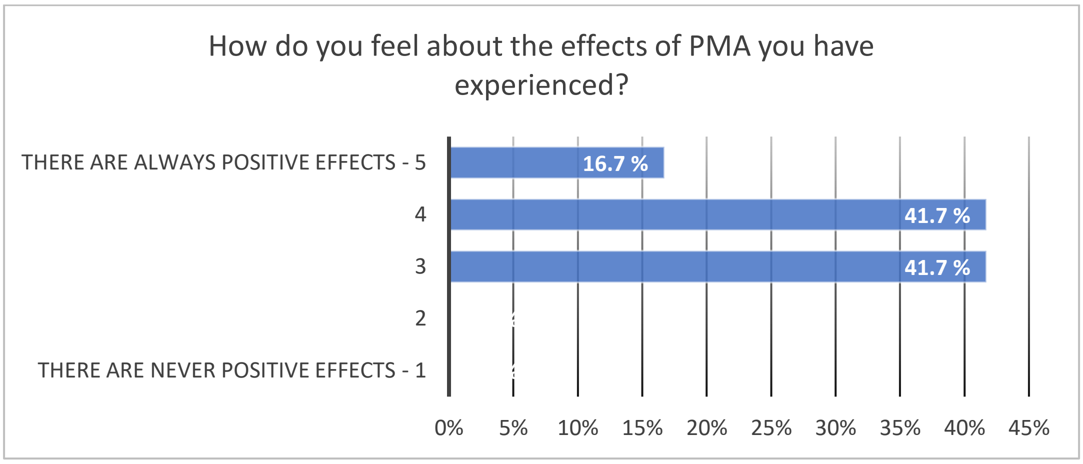


Figure 30, a survey question about the effects of PMA.

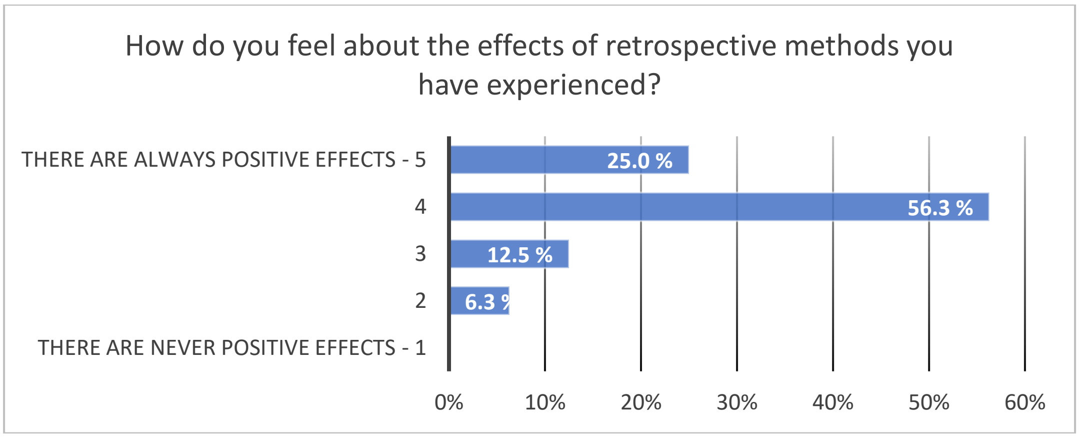


Figure 31, a survey question about the effects of other retrospective methods.

Turning short PMA sessions into useful results may be a notable struggle in the game development industry given how lightweight their methodology appears to be. The feasibility of reducing the amount of time PMA uses has been doubted in the past [Kerth and Weinberg, 2013] yet such adaptions have been made in the form of the Lightweight method [Dingsøyr et al. 2001, Stålhane et al. 2003], Chandler [2013] and in the methods the survey respondents have used. The survey ratings for usefulness of PMA did not indicate a negative perception of the results, so it would appear that less time-consuming methods can be useful, but further studies on their benefits could be done to ascertain how well the benefits of PMA can be obtained with less time-consuming methods.

7. Discussion and conclusions

The first research question: “What is the current state of PMA adoption in game development?”, can be partly answered with the survey results, which have shown that PMA is a common practice in the game industry. However, the exact industry adoption rate can’t be established from the survey’s results as the data appears skewered towards larger companies and more management focused individuals. The high rate of PMA practice experience among the respondents may imply that the practice of PMA has spread into game industry and become widely adopted, but the development methods of smaller companies such as start-up and independent game studios could differ from this data. Such studios often get left in the sidelines of research as they may be more difficult to reach [Neogames, 2021] or considered too disorganized to yield interesting data [Koutonen and Leppänen, 2013].

This thesis addressed other retrospective methods, both earlier in Section 4 and in the survey. The data collected on retrospection during projects doesn’t significantly contradict the method presented by Derby and Larsen [2006], which suggests that there aren’t significant changes. Furthermore, retrospectives appear to have become a common practice, even more so than PMA. The survey responses did not seem to show any evidence of the potential competition between PMA and other retrospective practices, as both of the methods had widespread adoption. Game industry professionals appear to see PMA as a different practice from other modern forms of retrospectives, which is also how they are treated in literature.

The second research question: “How is PMA used in modern game development”, was also investigated through the survey, which shows that the ways of using PMA in modern game development deviate in some aspects from the methods described in literature two decades ago. The biggest change is abandoning the idea of an external facilitator, which was a common recommendation in the past but nearly nonexistent in the survey results. Furthermore, neither the time spend on PMA meetings nor the rate of how commonly a project received a PMA were high.

The low amount of time spent by survey participants is within the range of the Lightweight method described earlier in this thesis, which supports the idea that Lightweight method may be somewhat close to the PMA used in modern game development, though some other factors, such as the use of project member as a facilitator still contradict this.

The question of how common PMA should be, meaning if it should be done for every project or only specific ones, has not received a consensus in literature, though more often [Verner and Evanco, 2005] or always [Birk *et al.* 2002] seems to be generally favored by researchers. While the rate in the survey is certainly not high, its significance is up to interpretation. One comparison could be drawn to the study by Verner and Evanco [2005], where 29% of the project received one, which is a similar rate to the results in this study.

Industry usage rates might be far behind the researchers' suggestions, so game developers might not be specifically underusing PMA despite the numbers in the survey appearing to be quite low.

The survey shows that some parts of the PMA still closely follow the older literature. For one, meetings and written reports were commonly used in PMA. Furthermore, the use of PMA reports doesn't appear to have changed, despite some alternative uses for them surfacing. The survey respondents named management and PMA participants as the only targets for the reports. Considering the previously estimated ~400 public PMA reports in 2022 and attempts the calculate the amount of video games having resulted in 43,806 games in 2014 [Polygon, 2014] and 208,048 in 2021 [MobyGames, 2022], it's clear that publicly releasing PMA reports is not at all a common habit among game developers.

The third research question: "What uses do PMA reports have and how are they realized in game industry?", produced interesting results. Despite the releasing of public PMA reports being quite rare in the industry and not even showing up in the survey, reading public reports was shown to be common. Even the relatively small amount of public PMA reports might be sufficient for many purposes. Since many developers are familiar with them, the lessons learned in them are getting spread in the industry to some degree. The number of available public PMA reports is also more than sufficient for research, which, as discussed in Section 4, can provide unique perspectives and complement other research.

The benefits of making public PMA reports might not be very attractive to game developers. The practice of public releases hasn't been widely adapted and rate of public releases has been suggested to be diminishing [Lu *et al.* 2019] even though the game industry itself only keeps growing. The main concerns for game developers who made the public reports are likely connecting with fans and other developers. Lu *et al.* [2019] suggest that lower numbers of new PMA reports may be caused by developers looking into other similar methods such as public discussion on Reddit to gain the same benefits. The openness to discuss game development practices might not be dying out, despite the idea of publicizing PMA reports failing to gain any more popularity as developers instead look at other means of publicizing their experiences.

One major limitation of this work is, as is repeated often in this thesis, the lack of recent research on PMA and the low amount of scientific literature on game development practices. This thesis takes a look on the current state of PMA in game development, but it will take more research to properly shed light on these topics. A second limitation in this thesis is the survey design. While helpful in answering this thesis' research questions, the study could have either looked deeper into PMA methods or tried to collect a more comprehensive group of respondents. Focusing more on either of those goals in the making of the survey could have led to better answers for some of the research questions. Conducting such an investigation could be one approach for future works on this topic.

Another good topic for future works would formulating a new method of PMA for modern game development. There is a clear lack of guidance on adopting PMA in modern game development as the methods differ in practice from the literature. The game industry also faces difficulties in software development method adoption, so research on how to best apply PMA in game development could be very beneficial. Furthermore, smaller companies and especially growing companies could also offer an interesting avenue of research in the context of PMA adoption, as such companies are not well covered in literature. The method of doing PMA has also kept becoming more lightweight over time, which may make it easier to adopt in even smaller projects and which could also be a topic for more research.

References

Jarmo Ahonen and Paula Savolainen. 2010. Software engineering projects may fail before they are started: Post-mortem analysis of five cancelled projects. *The Journal of Systems and Software* Vol. 83, p. 2175-2187.

Kent Beck *et al.* 2001. Manifesto for agile software development. Available as *http://agilemanifesto.org/* Checked 14.2.2022.

Andreas Birk, Torgeir Dingsøyr, and Tor Stålhane. 2002. Postmortem: never leave a project without it. *IEEE Software* Vol. 19 (3), p. 43-35.

Finn Olav Bjørnson, Alf Inge Wan, and Erik Arisholm. 2009. Improving the effectiveness of root cause analysis in post mortem analysis: A controlled experiment. *Information and Software Technology* Vol. 51 (1), p. 150-161.

Heather Maxwell Chandler. 2013. *The game production handbook, 3rd edition*. Jones & Bartlett Learning.

Bonnie Collier, Tom DeMarco, and Peter Fearey. 1996. A defined process for project postmortem review. *IEEE Software* Vol. 13 (4), p. 65.

Esther Derby and Diana Larsen. 2006. *Agile retrospectives: Making good teams great.* The Pragmatic Bookshelf.

Digital.ai. 2021. 15th annual state of agile survey report. Available as *https://digital.ai/resource-center/analyst-reports/state-of-agile-report* Checked 2.5.2022.

Digital.ai. No date. Common agile terms. Available as *https://digital.ai/glossary/agile-scrum-terminology* Checked 2.5.2022.

Torgeir Dingsøyr, Nils Brede Moe, and Øystein Nytrø. 2001. Augmenting experience reports with Lightweight postmortem reviews. In: *International Conference on Product Focused Software Process Improvement*, p. 167-181.

Robert Flunger, Andreas Mladenow, and Christine Strauss. 2017. The free-to-play business model. In: *Proceedings of the 19th International Conference on Information Integration and Web-based Applications & Services*, p. 373-379.

Game Career Guide. 2022. Postmortems. Available as *https://gamecareerguide.com/postmortems/* Checked 22.5.2022.

Game Developer. 2016. Postmortem: Paradox Development Studio's Stellaris. Available as *https://www.gamedeveloper.com/design/postmortem-paradox-development-studio-s-i-stellaris-i- Checked 20.5.2022.*

Game Developer. 2022. Search "postmortem". Available as *https://www.gamedeveloper.com/search?q=postmortem* Checked 22.5.2022.

GDC (Game Developers Conference). 2018. The Paradox dlc model: planning for the long term. Available as *https://www.youtube.com/watch?v=muvZni2DQfk* Checked 20.5.2022.

GDC (Game Developers Conference). 2021. ‘Death Stranding’: an AI postmortem. Available as *https://www.youtube.com/watch?v=yqZE5O8VPAU* Checked 21.5.2022.

Robert L. Glass. 2002. Project retrospectives, and why they never happen. *IEEE Software* Vol. 19 (5), p. 112-111.

Wolfgang Hamann. 2003. Goodbye postmortems, hello critical stage analysis. Available as *https://www.gamedeveloper.com/production/goodbye-postmortems-hello-critical-stage-analysis* Checked 16.5.2022.

Norman Kerth and Gerald Weinberg. 2013. *Project Retrospectives: A Handbook for Team Reviews*. Addison-Wesley Professional.

Jussi Koutonen and Mauri Leppänen. 2013. How are agile methods and practices deployed in video game development? A survey into Finnish game studios. *Agile Processes in Software Engineering and Extreme Programming*, Vol. 149, p. 135-149.

Annakaisa Kultima and Kati Alha. 2010. ”Hopefully everything I’m doing has to do with innovation” games industry professionals on innovation in 2009. *2010 2nd International IEEE Consumer Electronics Society’s Games Innovations Conference*, p. 1-8.

Miikka Lehtonen, Chien Lu, Timo Nummenmaa, and Jaakko Peltonen. 2020. Adoption of requirements engineering methods in game development: A literature and postmortem analysis. In: Anthony Brooks and Eva Irene Brooks (eds.), *Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering*, LNICST Vol. 328, p. 436-457.

Tom Leonard. 1999. Postmortem: Thief: The Dark Project. Available as *https://www.gamedeveloper.com/design/postmortem-i-thief-the-dark-project-i-* Checked 21.1.2022.

Chien Lu, Jaakko Peltonen, and Timo Nummenmaa. 2019. Game postmortems vs. developer Reddit AMAs: computational analysis of developer communication. In: *Proceedings of the 14th International Conference on the Foundations of Digital Games*, p. 1-7.

Timothy McKenzie, Miguel Morales Trujillo, and Simon Hoermann. 2019. Software engineering practices and methods in the game development industry. In: *Extended Abstracts of the Annual Symposium on Computer-Human Interaction in Play*, p. 181-193.

MobyGames. 2022. MobyGames Stats – 2021 edition. Available as *https://www.mobygames.com/forums/dga,2/dgb,3/dgm,257542/* Checked 22.5.2022.

Mauri Myllyaho, Outi Salo, Jukka Kääriäinen, Jarkko Hyysalo, and Juha Koskela. 2004. A review of small and large post-mortem analysis methods. In: *Proceedings of the ICSSEA, Paris 2004.*

Emerson Murphy-Hill, Thomas Zimmermann, and Nachiappan Nagappan. 2014. 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,* p. 1-11.

Neogames. 2021. Finnish game industry report 2020. Available as *https://neogames.fi/finnish-game-industry-report-2020/* Checked 2.5.2022.

Andrew Nolan. 1999. Learning from success. *IEEE Software* Vol. 16 (1), p. 97-105.

Fábio Petrillo, Marcelo Pimenta, Francisco Trindade, and Carlos Dietrich. 2009. What went wrong? A survey of problems in game development. *Computers in Entertainment* Vol. 7 (1), p. 36.

Cristiano Politowski, Lisandra Fontoura, Fábio Petrillo, and Yann-Gaël Guéhéneuc. 2016. Are the old days gone?: A survey on actual software engineering processes in video game industry. In: *Proceedings of the 5th International Workshop on Games and Software Engineering*, p. 22-28.

Cristiano Politowski, Fábio Petrillo, Gabriel C. Ullmann, and Yann-Gaël Guéhéneuc. 2021. Game industry problems: An extensive analysis of the gray literature. *Information and Software Technology* Vol. 134, p. 106538.

Polygon. 2014. A list of every video game ever made: 43,806 names, and counting. Available as *https://www.polygon.com/2014/4/20/5633602/list-of-every-video-game-all-time* Checked 22.5.2022.

Ken Schwaber and Jeff Sutherland. 2020. The Scrum Guide. Available as *https://scrumguides.org/scrum-guide.html* Checked 14.2.2022.

Raymond Scupin. 1997. The KJ method: a technique for analyzing data derived from Japanese ethnology. *Human organization* Vol. 56 (2), p. 233-237.

Tor Stålhane, Torgeir Dingsøyr, Geir Kjetil Hanssen, and Nils Brede Moe. 2003. Post mortem – An assessment of two approaches. *Empirical Methods and Studies in Software Engineering* Vol. 2765, p. 129-141.

Valve. 2022. Stellaris: Overlord. Available as *https://store.steampowered.com/app/1889490/Stellaris_Overlord/* Checked 20.5.2022.

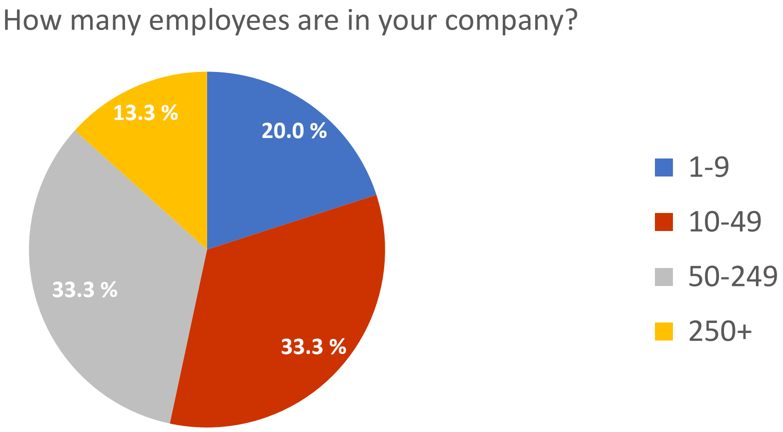
June Verner and William Evanco. 2005. In-house software development: what project management practices lead to success? *IEEE Software* Vol. 22 (1), p. 86-93.

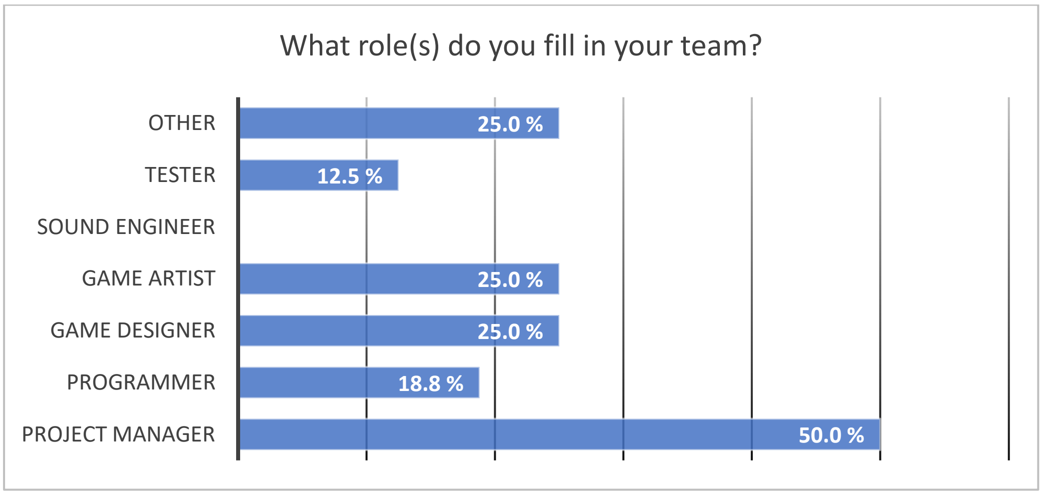
Michael Washburn, Pavithra Sathiyanarayanan, Meiyappan Nagappan, Thomas Zimmermann, and Christian Bird. 2016. What went right and what went wrong: An analysis of 155 postmortems from game development. In: *Proceedings – International Conference on Software Engineering,* p. 280-289.

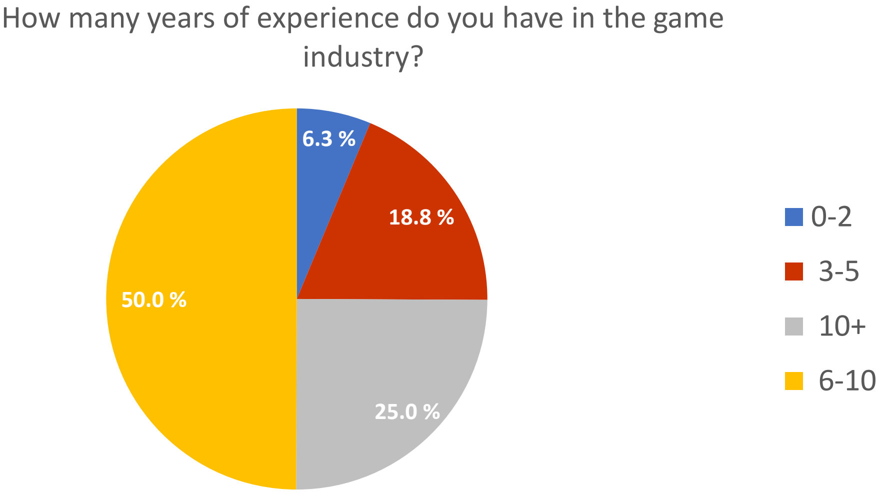
Wikipedia. 2022. Autopsy. Available as *https://en.wikipedia.org/wiki/Autopsy* Checked 16.5.2022.

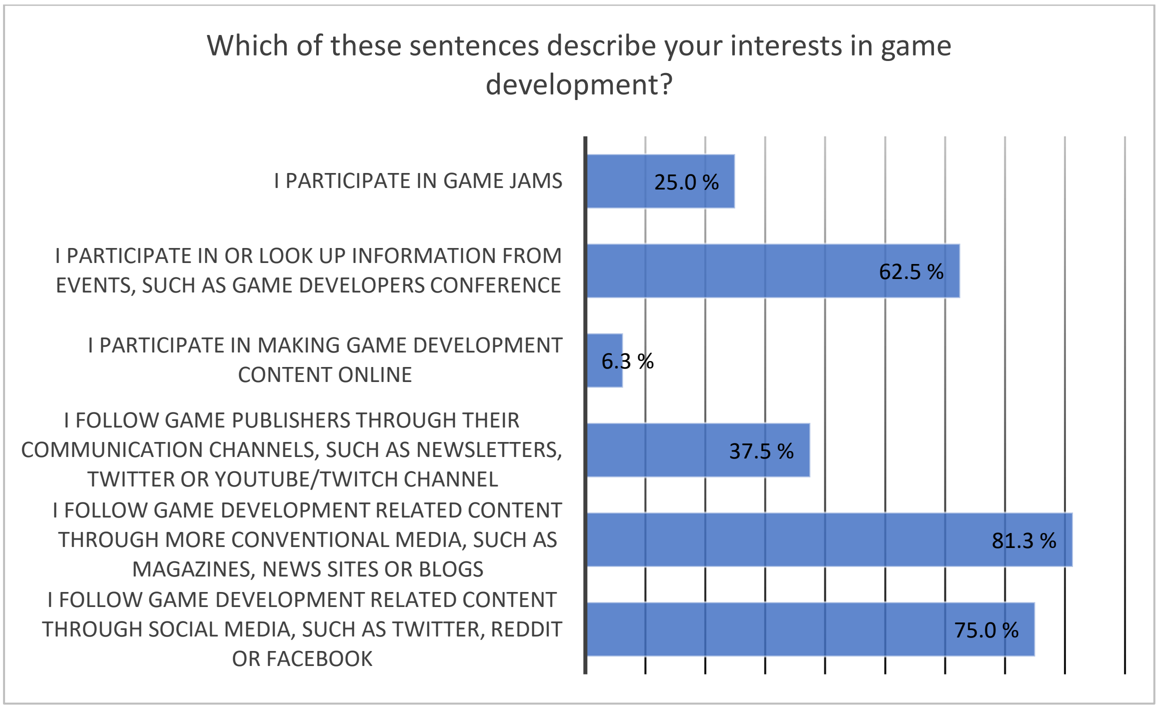
Appendix – Survey questions and graphs

Background questions:

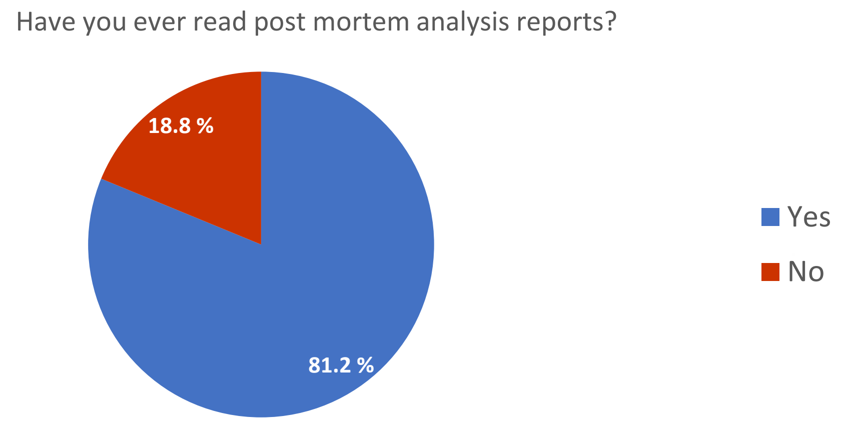


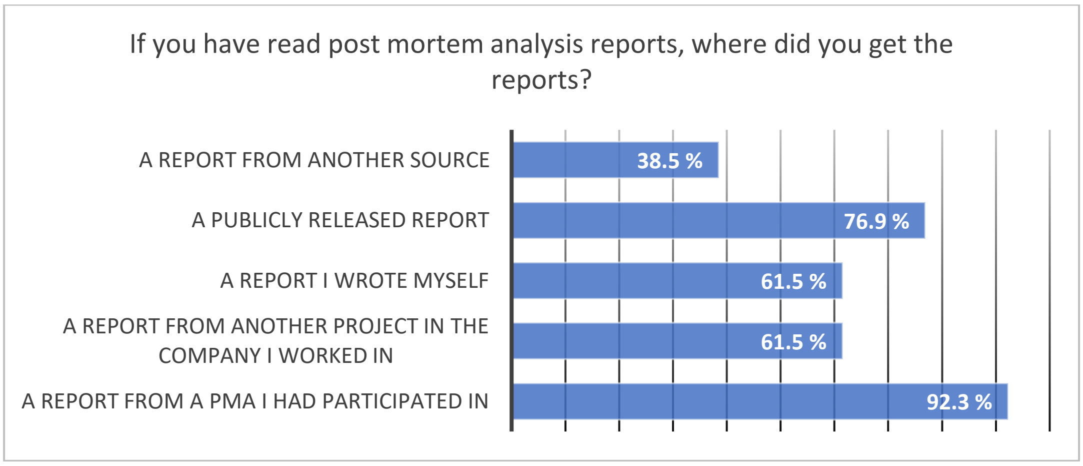


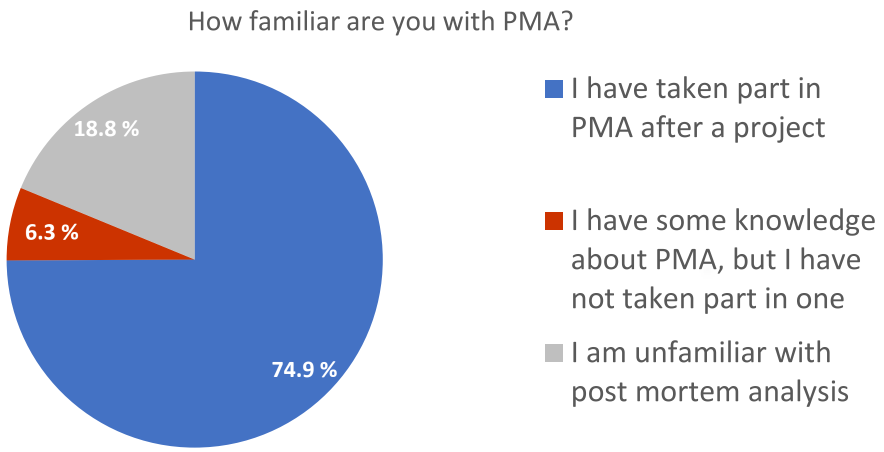




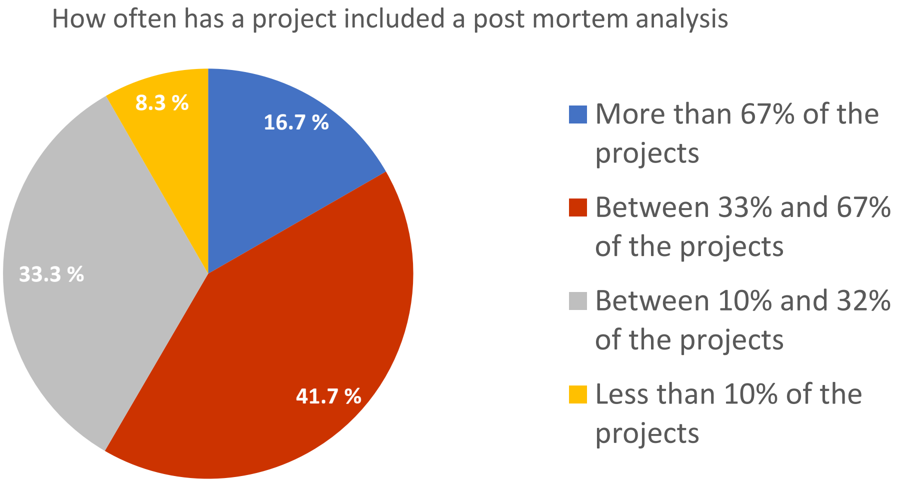
About post mortem analysis:

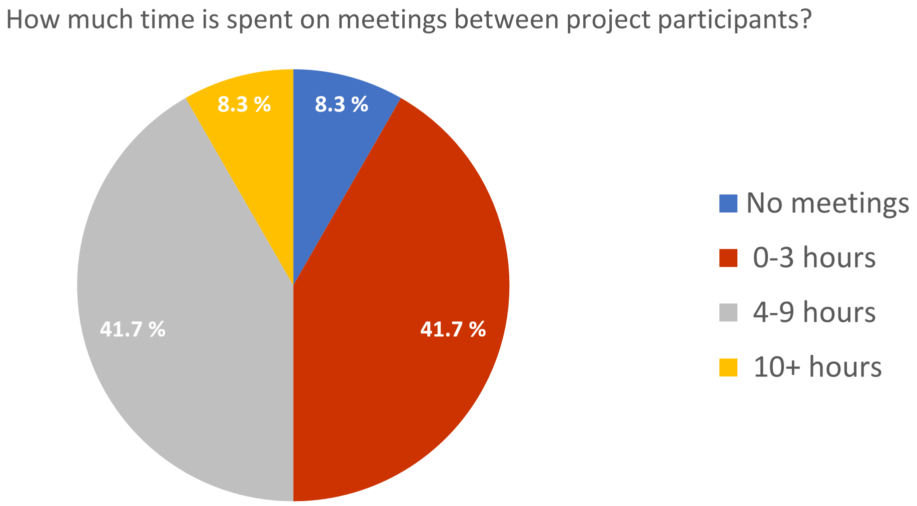


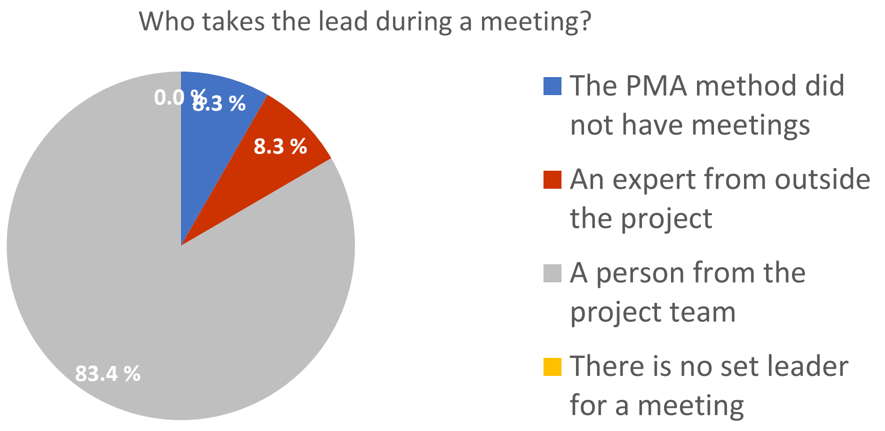


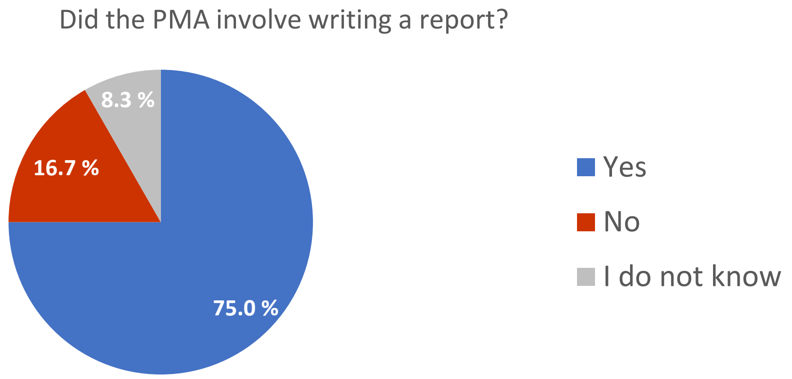


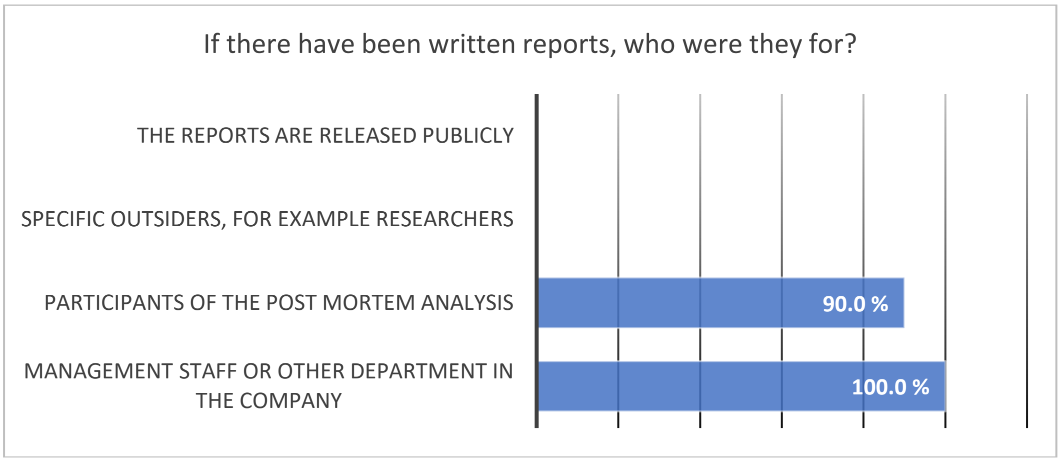
About the use of post mortem analysis:

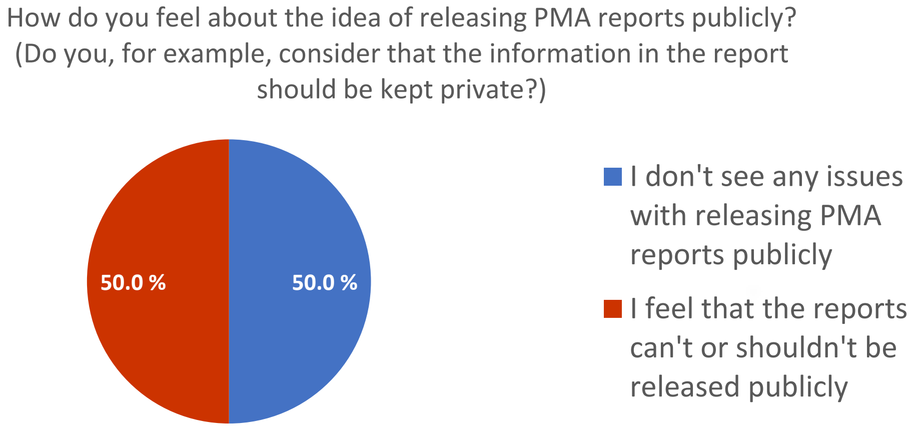


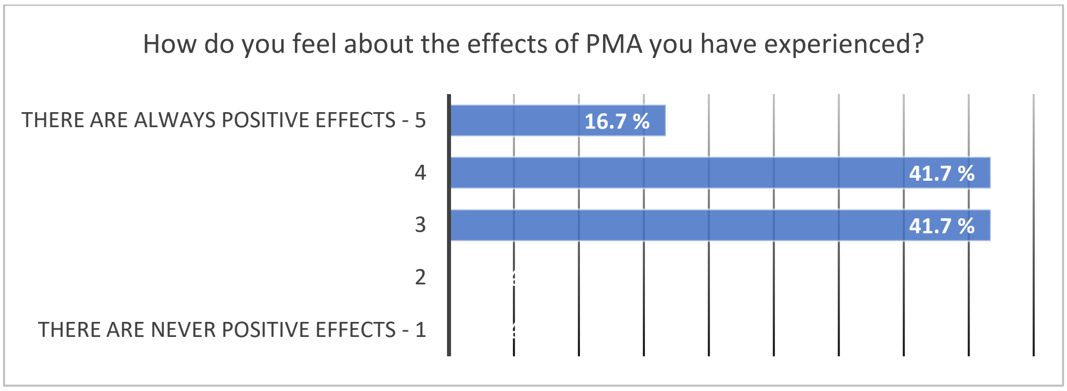



Who takes the lead during a meeting?


Did the PMA involve writing a report?


If there have been written reports, who were they for?


How do you feel about the idea of releasing PMA reports publicly? (Do you, for example, consider that the information in the report should be kept private?)



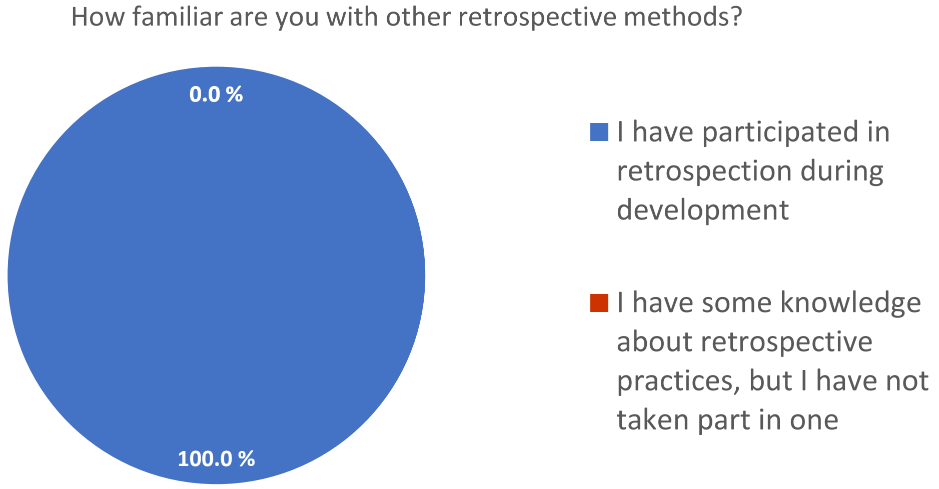
(Optional question) You may elaborate on your experiences with post mortem analysis here.

1. Post-mortem analysis of a game project is a powerful tool for analysing what went right and what went wrong. Internally, it can help the game company spread the word about pitfalls, shortcomings, and other learnings. Externally, it is a great way to market a game company and help educate the industry. Retrospectives and yearly ”post-mortems” arranged mid-production are also very useful. They help with future roadmapping, backlog grooming, sprint planning, and provide insights for the whole company.

2. Personally, post mortems usually just end up listing quite obvious things that seldom lead to any concrete benefit in the future.

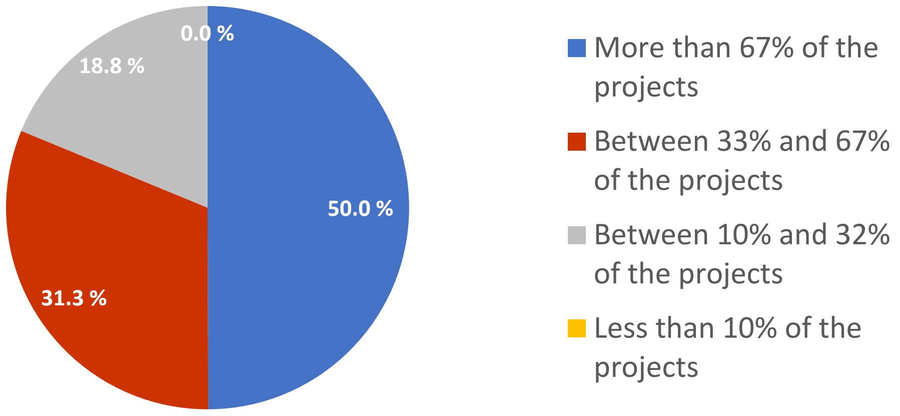
3. Varies a lot how they're done and how useful they are

About other retrospective methods:

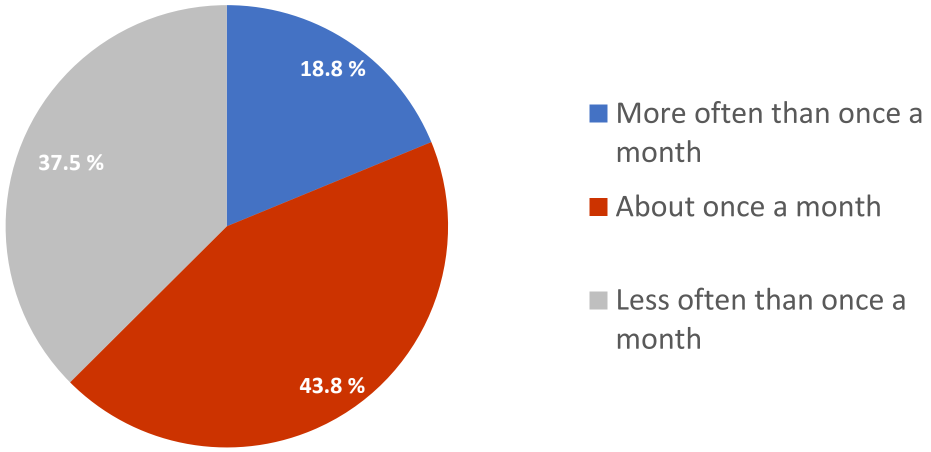


About the use of other retrospective methods:

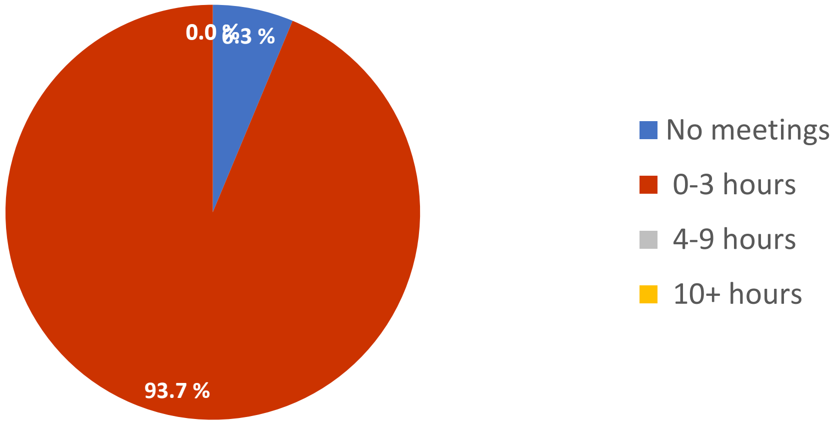
How often has a project included a retrospective method?

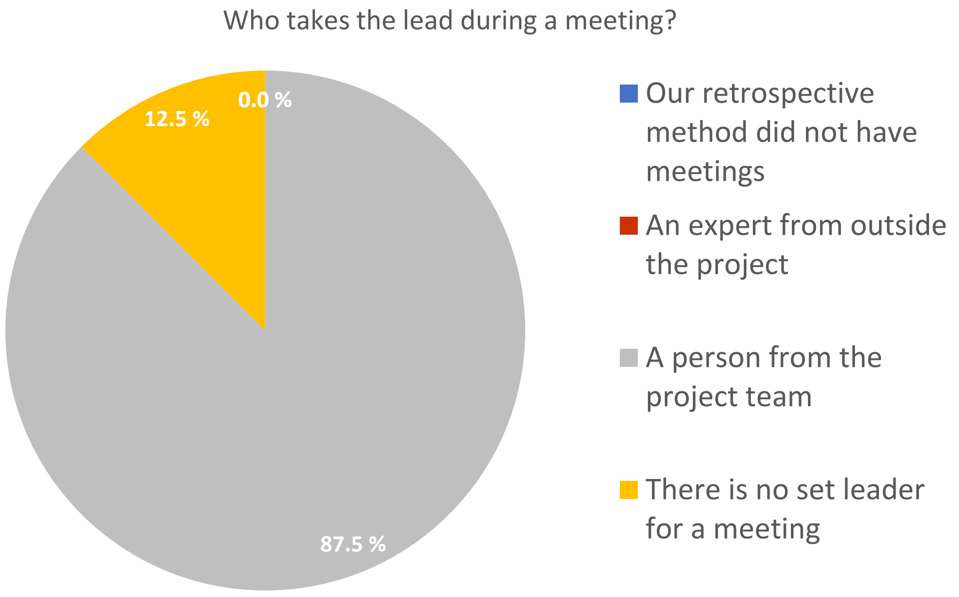


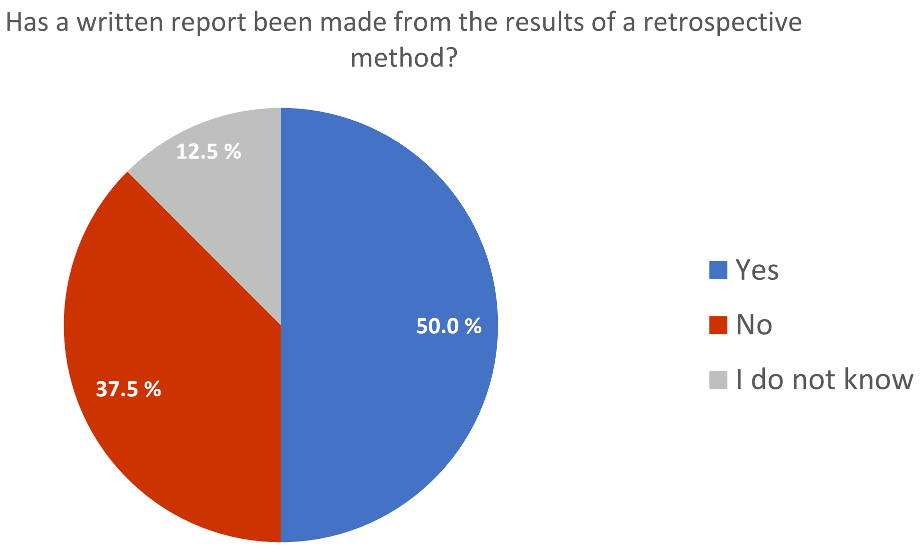
When retrospective methods have been practiced during a project, how often were they practiced?

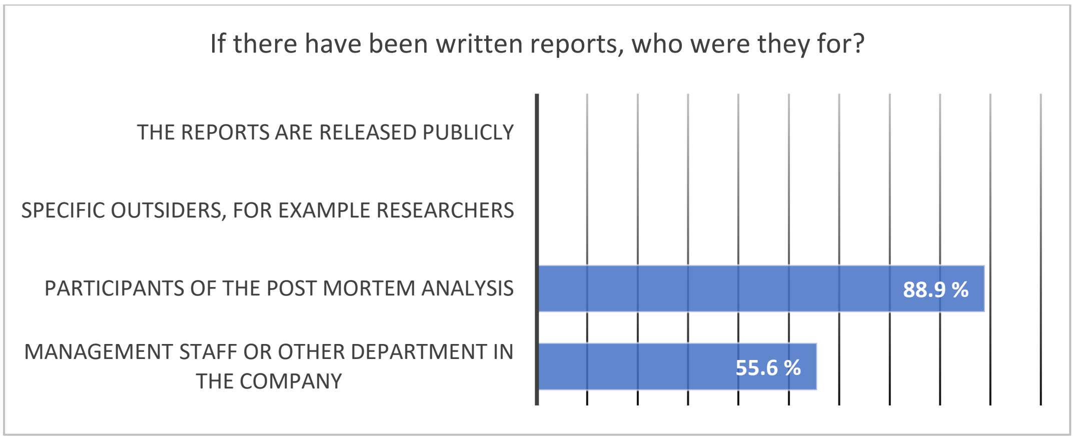


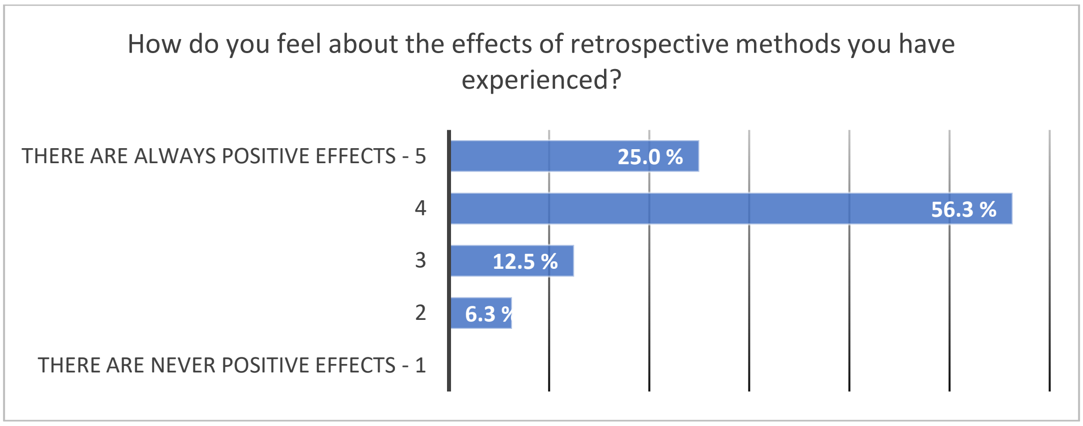
How much time is spent was spent on average in a single meeting between project participants during the retrospective practice?




Who takes the lead during a meeting?
0.0 %
12.5 %
87.5 %
🟦 Our retrospective method did not have meetings
🟥 An expert from outside the project
⬜ A person from the project team
🟨 There is no set leader for a meeting


Has a written report been made from the results of a retrospective method?
12.5 %
50.0 %
37.5 %
🟦 Yes
🟥 No
⬜ I do not know


If there have been written reports, who were they for?
THE REPORTS ARE RELEASED PUBLICLY
SPECIFIC OUTSIDERS, FOR EXAMPLE RESEARCHERS
PARTICIPANTS OF THE POST MORTEM ANALYSIS
88.9 %
MANAGEMENT STAFF OR OTHER DEPARTMENT IN THE COMPANY
55.6 %



(Optional question) You may elaborate on your experiences with retrospective methods here.

1. It is always useful to have a sprint retrospective, but the most useful retrospectives in my experience have been quarterly and yearly retrospectives. They can't be communicated externally for several reasons, but internally they make a huge difference.

2. Likewise, rarely has any concrete benefit come out of the retrospective in my view. Problems-areas are not unknown to begin with, they are usually quite obvious. Practical solutions on the other hand are incredibly hard to solve - if they weren't they would already have been addressed during sprints/production.

3. Crucial in identifying and fixing issues related to the development process

4. I think much of it is also done through casual discussion, talking about projects and lessons learnt, not necessarily through raports or meetings on this topic only. At least this has been my experience with small indie games and game jams.

5. It's an important part to develop your processes and make work smoother and more fun for the team. Good to have some method for it, in minimum the what went well, could improve, continue doing, but there are many ways to keep retros, we had them regularly when in non-games IT jobs. In games had none, some or a bit randomly in different companies though we mean to have them once a month now on. But! They are not very useful if no action points are generated