Video Game Project Management Anti-patterns

Gabriel C. Ullmann
g_cavalh@live.concordia.ca
Concordia University
Montreal, Quebec, Canada

Cristiano Politowski
c_polito@encs.concordia.ca
Concordia University
Montreal, Quebec, Canada

Yann-Gaël Guéhéneuc
yann-gael.gueheneuc@concordia.ca
Concordia University
Montreal, Quebec, Canada

Fabio Petrillo
fabio@petrillo.com
Université du Québec à Chicoutimi
Chicoutimi, Quebec, Canada

João Eduardo Montandon
joao.montandon@dcc.ufmg.br
Universidade Federal de Minas Gerais
Belo Horizonte, Minas Gerais, Brazil

ABSTRACT

Project Management anti-patterns are well-documented in the software-engineering literature, and studying them allows understanding their impacts on teams and projects. The video game development industry is known for its mismanagement practices, and therefore applying this knowledge would help improve game developers' productivity and well-being. In this paper, we map project management anti-patterns to anti-patterns reported by game developers in the gray literature. We read 440 postmortem problems, identified anti-pattern candidates, and related them with definitions from the software-engineering literature. We discovered that most anti-pattern candidates could be mapped to anti-patterns in the software-engineering literature, except for Feature Creep, Feature Cuts, Working on Multiple Projects, and Absent or Inadequate Tools. We discussed the impact of the unmapped candidates on the development process while also drawing a parallel between video games and traditional software development. Future works include validating the definitions of the candidates via survey with practitioners and also considering development anti-patterns.

KEYWORDS

software, video games, software development, anti-patterns, project management

ACM Reference Format:
Gabriel C. Ullmann, Cristiano Politowski, Yann-Gaël Guéhéneuc, Fabio Petrillo, and João Eduardo Montandon. 2022. Video Game Project Management Anti-patterns. In *6th International ICSE Workshop on Games and Software Engineering: Engineering fun, inspiration, and motivation (GAS'22), May 20, 2022, Pittsburgh, PA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3524494.3527623

1 INTRODUCTION

Context: A software anti-pattern is a solution to a problem that yields negative consequences [3]. They exist for *Development*, *Architecture*, and *Project Management*. The latter are related to software processes, human resources, and communication. As defined by Brown et al. [3], "[project] management anti-patterns describe how software projects are impaired by people issues, processes, resources, and external relationships". They describe scenarios where human communication and relationship issues affect software development processes [9].

Problem: Over the past years, many project management issues have been reported in the video game development industry[1,2,3]. These were not isolated issues: 45% of problems documented in postmortems are related to management practices [14]. However, while some studies present data related to video game practices [11] and development anti-patterns [6], currently there is no discussion about video game project management anti-patterns.

Objective & Research Questions: Our main goal is to find out how project management anti-patterns in the video game industry relate to traditional software-engineering anti-patterns. Specifically, we answer the following research question: *how do video game project management anti-patterns fit into the software-engineering anti-patterns literature?*

Method: We read 440 video game postmortem problems and, based on their description, grouped them into anti-pattern candidates, which are concise anti-pattern proposals. We then mapped these proposals to project management anti-patterns documented in the software-engineering literature.

Results: Among the most common project management anti-patterns found in video game development, the most frequent was *Project Mismanagement* [3], followed by *Death March* [15], *Shoeless Children* [9], *Cover Your Assets* [3], and *False Surrogate Endpoint* [5]. *Feature Creep* and *Feature Cut* are the most common unmapped anti-pattern candidates, followed by *Absent or Inadequate Tools* and *Working on Multiple Projects.* While some mappings highlight similarities between video game and traditional software development,

[1]https://www.bloomberg.com/news/articles/2021-11-17/activision-atvi-ceo-bobby-kotick-is-under-pressure-to-resign
[2]https://www.businessinsider.com/video-game-development-problems-crunch-culture-ea-rockstar-epic-explained-2019-5
[3]https://www.bloomberg.com/news/articles/2022-01-03/bioshock-creator-s-next-game-and-its-narrative-legos-in-turmoil

the absence of others shows the need for studying anti-patterns in the context of video game development.

Conclusion: Our data shows that some project management anti-patterns happen in both fields: traditional software and video games. In future works, we plan to validate the anti-pattern candidate definitions via a survey to understand if they are accurate, considering a broad range of video games and professionals. We also intend to analyze video game development anti-patterns.

Paper summary: Section 2 describes the methods of data collection and inference. Section 3 discusses the analysis of the dataset and the relations between variables. Section 4 discusses the results. Section 5 summarises threats to validity. Section 6 describes related works. Section 7 concludes with future work.

2 METHOD

We divide our method into three steps, which we perform sequentially, as shown in Figure 1. We describe these steps in the remainder of this section.

2.1 Dataset of Video Game Problems

We used a dataset of video game problems [13, 14], which contains a curated list of problems gathered from postmortems. It comprises 200 projects from 1997 to 2019, describing 927 problems. Each problem is composed of the video game name, year of release, and a quote, which is a small excerpt from the postmortem describing an issue that occurred during development.

These problems are grouped in four categories: *Production*, *People Management*, *Feature Management*, and *Business*. We considered only *People/Feature Management* problems, which correspond to 45% of the problems.

- *Production* describes operational problems that happen during the production phase;
- *People Management* describes management problems related to teams and stakeholders;
- *Feature Management* describes management problems related to feature planning, scheduling, and development;
- *Business* relates to marketing and strategies to generate revenue. It is not considered a management problem in this analysis.

2.2 Selecting Project Management Problems

Once we defined the dataset as shown in subsection 2.1, we filtered out the problems to include only those related to Project Management. This choice allowed us to reduce our scope while focusing on the category with the most occurrences in the dataset - 440 distinct problems.

2.3 From Problems to Anti-pattern Candidates

We read the quotes from the filtered dataset, isolating their descriptions of problems and solutions. For problems in which the quote did not give us enough information, we read the whole postmortem. This process was first done by the main author, and subsequently reviewed iteratively by the others in order to make sure descriptions were concise and well defined for each case.

Then, we applied open card sorting iteratively to group the problems. At first, the main author grouped the problems independently. After finishing this step, all authors discussed together, and sorting changes were made in common agreement. The sorting was considered finished when all authors reached an agreement.

We performed the sorting based on the content of the quotes. We grouped quotes when they shared the same problems and solutions. For example, we found several postmortem quotes mentioning financial problems and how developers reacted to the situation through working harder or reducing expenses, which we grouped into a "Underbudgeted Project" anti-pattern candidate.

Thus, we obtained 66 groups or anti-pattern candidates. We structured these candidates using the mini-template for anti-patterns described by Brown et al. [3], who provided three types of anti-pattern templates. These vary according to the amount of information that they provide. For example, the *Pseudo anti-pattern template* shows only the name and the problem while the *Full anti-pattern template* enumerates various attributes. We chose to work with the intermediate *Mini anti-pattern template*, which contains:

- **Name**: What shall this anti-pattern be called by practitioners (pejorative)?
- **Anti-pattern Problem**: What is the recurrent solution that causes negative consequences?
- **Refactored Solution**: How do we avoid, minimize, or refactor the anti-pattern problem?

2.4 From Anti-pattern Candidates to Software Engineering Literature

For each anti-pattern candidate, we manually searched for a fitting project management anti-pattern in the software-engineering literature [2, 3, 9]. For example, we related "Underbudgeted Project" to the Shoeless Children anti-pattern from Brown et al. [3] because they both describe the same problem (deprivation of resources) and solution (strategic budget-cutting that excludes essential infrastructure). Table 2 show more examples of the mapping.

Not all anti-pattern candidates could be mapped to a software-engineering counterpart. We describe them in Section 3 and discuss their causes and consequences in-depth in Section 4.

3 RESULTS

From the 440 management problems in the video game literature, we derived 66 different anti-pattern candidates. Figure 2 shows that we could map 57% of the candidates to software-engineering anti-patterns.

Among the 66 anti-pattern candidates, we found 12 with a number of occurrences equal to or greater than 10 (Table 1). Four candidates were not in the software-engineering literature: *Feature Creep*, *Feature Cuts*, *Working on Multiple Projects*, and *Absent or Inadequate Tools*. The next section describes these anti-pattern candidates. Due to the lack of space, the candidates that could be mapped into the software-engineering literature are available on-line[4].

[4]https://zenodo.org/record/5828315

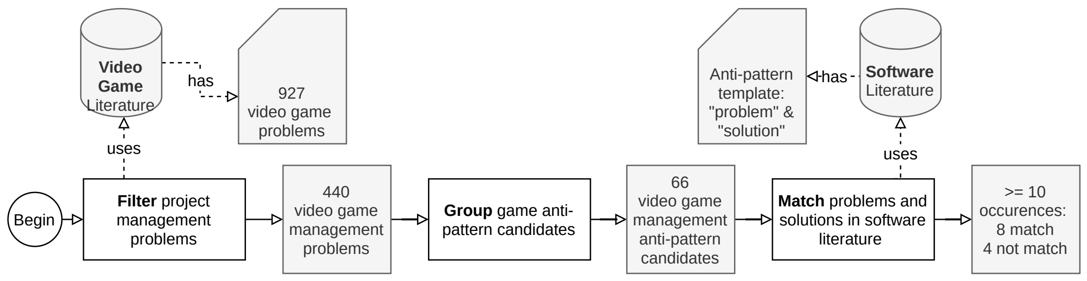
Video Game Literature
has
927 video game problems
uses
Begin
Filter project management problems
440 video game management problems
Group game anti-pattern candidates
66 video game management anti-pattern candidates
Match problems and solutions in software literature
>= 10 occurences: 8 match 4 not match
Anti-pattern template: "problem" & "solution"
has
Software Literature
uses

Figure 1: From the video game problems dataset [14], we filtered out those not related to management. From these, we grouped anti-pattern candidates. After that, we mapped the candidates to the software literature [3, 9].

Table 1: The most common anti-patterns in video game literature (postmortems) and its mapped software anti-patterns.

Video game AP candidate	Occ.	Software AP	AP Category	Source
Underestimated Tasks	37	Project Mismanagement	Management	Brown et al. [3]
Feature Creep	33	-	-	-
Death March	30	Death March	Management	Yourdon [15]
Feature Cuts	27	-	-	-
Understaffed Team	26	Project Mismanagement	Management	Brown et al. [3]
Underbudgeted Project	23	Shoeless Children	Environmental	Laplante and Neill [9]
Neglected Game Design	13	Cover your Assets	Architecture	Brown et al. [3]
Working on Multiple Projects	13	-	-	-
Absent or Inadequate Tools	12	-	-	-
Testing/QA Issues	11	Project Mismanagement	Management	Brown et al. [3]
False Surrogate Endpoint	10	False Surrogate Endpoint	Management	Cunningham [5]
Lack of Technical Knowledge/Planning	10	Project Mismanagement	Management	Brown et al. [3]

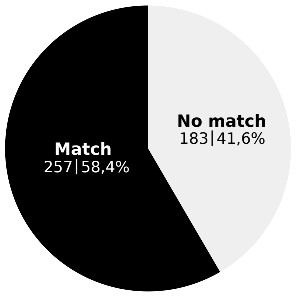

Figure 2: Number of postmortem problems that could be mapped to software literature anti-patterns

Feature Creep (33 cases)

Name: "Feature Creep".

Anti-pattern problem: Features keep being added to the project, expanding its initial scope excessively. While the new features might be worthy additions to the game, they might also overwhelm the game developers and distract from core features.

Refactored solution: Aiming for clear game design requirements and definitions from the beginning. Even though the development is iterative and features may change, keep them in check to avoid losing control.

Case example: Baldur's Gate 2, 2001

Problem: "While we shipped with nearly all the features we originally planned, we did start cutting quests and characters well before the final testing phase. We still ended up with over 200 hundred hours of gameplay."

Solution: "In retrospect we should have started this process many months earlier. One of the dangers of development is that game developers have a tendency to always add content if they are given time. (...) Its wise to use that prioritized feature list to hone the work (of course ours was informal, which made it a little difficult)"

Full postmortem: https://www.gamedeveloper.com/design/-i-baldur-s-gate-ii-i-the-anatomy-of-a-sequel

Feature Cuts (27 cases)

Name: "Death by a Thousand Cuts", "Ripple Effect", "Domino Effect"

Anti-pattern problem: A video game project enters production with a well-defined feature set, but without proper resource allocation and priority definitions. Features may be partially developed, but when more in-depth planning is made, the management decides to cut some of them to save time and–or money.

Refactored solution: Prototyping and playtesting can help designers and developers understand what features are valuable to players and thus prioritise them. When cuts are unavoidable, discussing and documenting them with the participation of all developers is important to avoid excessive re-work.

Table 2: Example of the mapping process for problems, anti-pattern candidates and the literature

	Postmortem description	Anti-pattern candidate	Relates to	Literature description
Problem	· "Golden Abyss had 34 next-gen quality levels, each of which had five times the content of any PSP game we had ever made" · "From January to December of 2011, the team worked pretty much nonstop (...)"	Death March	Death March (Yourdon, 2003)	· The schedule has been compressed to less than half the amount estimated by a rational estimating process · The functionality or other technical aspects of the project are twice what they would be under normal circumstances
Solution	· "Negotiating as reasonable a schedule as possible (...)" · "Holding weekly meetings with every individual on the team, where issues (...) can be raised directly." · "Vetting the scope of our next project up front (...)"			· Agreement on interfaces · Peer reviews · Metric-based scheduling and management · Formal risk management
Problem	· "Because we didnt́ have a firm design, it was impossible to do accurate time estimates" · "The only way we could get the game done on time was to cut deeply into our testing schedule"	Underestimated Tasks	Project Mismanagement (Brown et al, 1998)	"Often, key activities are overlooked or minimized. These include technical planning (architecture) and quality–control activities (inspection and test)"
Solution	"Our current focus is on getting solid designs done up front and solid testing done on the back end"			· Prototyping · Create and document a risk management analysis · Define architectural dependencies
Problem	"(...) the original game lacked a truly focused design. We knew what the fundamentals of the game would be, but we did not have the specifics that we needed to create a solid, cohesive product."	Neglected Game Design	Cover your Assets (Brown et al, 1998)	"Document–driven software processes often employ authors who list alternatives instead of making decisions"
Solution	· "We created total level walkthroughs, with each room and encounter written out" · " Flowcharts were used to draw the preliminary levels, and concept art was used for key location elements"			Creating architectural blueprints, which are "a small set of diagrams and tables that communicate the operational, technical, and systems architecture of current and future information systems."

Case example: The Political Machine, 2004
Problem: "We had several other features in mind that we had to cut because of lack of time, time we would have had if we hadn't done multiplayer. (...) When gamers demand multiplayer, what they don't realize is that something has to be sacrificed for it. You have a finite budget and a finite time to use it (...)"
Solution: "I think the inclusion of different maps to play on would have increased the replayability of the game a great deal. (...). But the time in development and testing on multiplayer eliminated those kinds of things. (...) With that in mind, I wish we hadn't done multiplayer in The Political Machine."
Full postmortem: https://www.gamedeveloper.com/disciplines/postmortem-stardock-s-i-the-political-machine-i-

Working on Multiple Projects (13 cases)
Name: "Divide and Not Conquer", "False Parallelism"
Anti-pattern problem: Game developers must work multiple jobs, often for financial reasons, or manage several activities inside their company, taking a workload much higher than normal. They become overwhelmed and misaligned with project goals over time.
Refactored solution: Analysing the time availability of each game developer on the team and carefully managing resource allocation may be a way to mitigate this issue. Staffing up may also be an option when budget and structure allow.

Case example: Rollers of the Realm, 2014
Problem: "Having the production timeline stretch out meant Rollers rubbed against the work-for-hire projects that we lined up because we need to keep the studio lights on. (...)"
Solution: "We needed to split our resources into two project teams. Phantom Compass often works on multiple projects at once, and in many ways this is a strength, but in this case it caused more strain on finishing up Rollers than we would have liked (...)."

Full postmortem: https://gamedeveloper.com/design/postmortem-pinball-rpg-hybrid-i-rollers-of-the-realm-i-

Absent or Inadequate Tools (12 cases)

Name: "Bringing a Knife to a Gun Fight", "Wrong Tool for the Job"

Anti-pattern problem: Developers and–or artists begin production without having a standardized set of tools for their activities or using outdated/improvised tools. They do so due to management inexperience, lack of time, budget, or qualified staff.

Refactored solution: Looking for third-party tools that fit the project's needs or developing custom in-house tools are activities that must be carefully considered early in the project.

Case example: Swords & Soldiers, 2010

Problem: "The levels took a lot of time to build with the limited tools we had. With only one programmer, you cant ask for a good level design tool For design, then, we had to use the stock Windows text editor, Notepad."

Solution: "Our AI routines were built in a simple logic tree editor, which was built by a programming intern. And though the editor was a bit buggy, it was still a major improvement over editing those XML files in Notepad. (...) In hindsight, we probably should have put some time in coding creating a visual level editor."

Full postmortem: https://www.gamedeveloper.com/audio/postmortem-ronimo-games-i-swords-soldiers-i-

4 DISCUSSION

4.1 Mapped Anti-pattern candidates

Project Mismanagement, Underestimated Tasks and Death March are the top-3 most frequent. Most of the anti-pattern candidates were mapped to the Project Mismanagement anti-pattern, which describes a broad range of situations.

Knowing the prevalence of these anti-patterns can help project managers and product owners proactively monitor their development teams and correct their actions accordingly.

The mapping also shows the difficulty faced by game developers in creating a well-organized and stable work environment. Such issue has been frequently addressed in the literature [4, 7, 10]. Our results reinforce this issue.

4.2 Unmapped Anti-patterns candidates

The most important contribution of this work resides in the anti-pattern candidates not mapped into the software-engineering literature. Researchers should conduct further studies to better understand in which scenarios they arise, as well as solutions to solve or mitigate them. Researchers could also rely on this data to build a video game-oriented anti-pattern catalog.

We now discuss the circumstances in which these anti-patterns happen, contrasting the experiences of game developers and traditional software developers.

4.2.1 Feature Creep. In the context of traditional software, Elliot [8] argues that Feature Creep "is a natural result of building something new and exploring new territory". We argue this is also true for video game development, especially when looking for the fun factor, as described by the creators of *Matt Hazard: Blood Bath and Beyond*:"it took longer to feel out the fun factor of our game than we had anticipated, we crept over our allotted budget on both time and money". *Command and Conquer* creators also stated that "a team has to be able to incorporate new ideas during development" but must also "be able to cut features diplomatically when it is in the best interest of the project".

Additionally, Feature Creep and Feature Cuts are reported by Petrillo et al. [10] to be the problems most commonly described in postmortems. Politowski et al. [14] report that "feature creep, and, to a less degree, delays, are recurring problems during game development". Therefore, even though this anti-pattern is not referenced by Brada and Picha [2] in their catalog, we argue it is not a game-specific issue, since it is widely known both in video games and traditional software industries.

4.2.2 Feature Cuts. Similarly, the "Feature Cut" anti-pattern candidate is well-known. It distinguishes from "Feature Creep" because it describes the removal of features that were included in the initial scope, and not included afterward. When "Feature Creep" occurs in a video game project, the team may choose not to cut down the newly-added features and just move on with the extra workload. *Fallout Tactics* creators describe that "rather than reduce the size of the maps or lose maps altogether, we decided to press on and fill them with interesting objectives". However, the choice of adding/removing features is a common challenge in any project so we also argue it is not a game-specific problem.

4.2.3 Absent or Inadequate Tools. The recurrence of the "Absent or Inadequate Tools" anti-pattern candidate reflects the impact of tools in video game production. For example, *Brutal Legend* manager Caroline Esmurdoc states that "good tools needed to be a priority to ensure faster iteration times for our team". *Swords & Soldiers* creators say that developing a better AI scripting tool "would have saved the rest of the team a lot of time".

Different from traditional software, video game production encompasses several art processes: animation, graphics, and sound edition. In the postmortems, these tools are pointed out as the most frequent source of problems when they are not properly chosen and standardized.

Tool maturity also has an impact on project stability. Game engines, for instance, are continuously changing their internal APIs and systems to adapt to new GPU features, gaming hardware, and operating systems. Such upgrades may cause bugs or require developers to refactor code written for previous versions of the tool.

Additionally, game engines are not as varied as traditional software IDEs. Unreal and Unity are used by the wide majority of commercial projects [12] and this has an influence on professionals' learning choices. Since education institutions focus on courses for these popular tools, the developer's knowledge may not be broad enough, which makes it harder for them to choose or develop their own tools. Therefore, we argue this candidate is tightly related to video games.

4.2.4 Working on Multiple Projects. Especially in the indie scene, we can observe that the "Working on Multiple Projects" anti-pattern candidate is very recurrent since the informality and time flexibility of the production process allows it. For example, while one of the Mini Metro programmers worked full-time as a banking software developer, "Sportsfriends" developer Douglas Wilson says that he was "finishing up his PhD" concurrently with the development of the game. However, both video game studios and software companies may work on multiple projects simultaneously, and thus we cannot argue this is a game-specific anti-pattern candidate.

5 THREATS TO VALIDITY

Limited coverage of problem databases: The database of postmortems on which our research was based does not represent the whole industry or all issues that may arise in the management of a video game development team. Additionally, the anti-pattern catalog may also not represent the set of all documented issues of this kind, and in some cases, it does not provide accurate sources to deepen research.

Not all pairs of problems/solutions are documented: Since postmortems are informal memoirs of a video game's development, they do not always provide all details on how a problem has manifested and how it has been mitigated or solved. Consequentially, we cannot state which problems represent a fully-fledged anti-pattern. Similarly, there is no documented problem/solution description for all software anti-patterns. Especially on those documented by Cunningham [5], there is often only a problem description and little to no discussion regarding solutions.

Problems and anti-patterns were manually mapped: We relied on our interpretation of the problem quotes to group them into anti-pattern candidates, and also to relate these candidates to software anti-patterns. Aiming to reduce bias, we read the postmortems in full when these quotes did not provide enough context on both problem and solution. We then made several iterations of discussion, updating our candidate anti-pattern categorization when necessary.

6 RELATED WORKS

Brown et al. [3], Laplante and Neill [9] were our main reference about software anti-patterns. In their books, they illustrate the overall concepts and provide practical examples. The issue is that some of the descriptions are either too broad, short, or informal to convey a proper definition.

Brada and Picha [2] shows a more complete and up-to-date review of the software project management anti-patterns, which they refer to as "process anti-patterns". They created a open catalog of anti-patterns using various sources[5]. However, after further investigation, the list is not complete to this day.

To the best of our knowledge, video game project management anti-patterns are not present in the literature. At least not in the same format of traditional software anti-pattern.

As for development anti-patterns (related to code), Borrelli et al. [1] performed a similar work. The authors identified, from the video game literature (textbooks, interviews, gray literature), six types of bad smells related to performance, maintainability, and incorrect behavior problems. After that, they used a static analysis tool on 100 open source video games made with the Unity engine. They found the following bad smells:

- Allocating and destroying GameObjects in updates;
- Getting a GameObject reference finding it by name;
- Heavyweight Update methods;
- Lack of separation of concerns;
- Coupling objects through the IDE Inspector;
- Animation speed depends on the frame rate.

[5]https://github.com/ReliSA/Software-process-antipatterns-catalogue

7 CONCLUSION

In this work, we focused on understanding the relation between anti-patterns that occur in video games and the traditional software industry. Taking postmortem problems as a starting point, we identified anti-pattern candidates, which could be mapped to documented software anti-patterns in the majority of cases.

Regarding the four anti-pattern candidates that could not be mapped, we concluded that only "Absent or Inadequate Tools" closely relate to the game development industry, while the others may happen on any software project. Therefore, our findings show that management problems in video game projects are not exclusive since they also happen in the traditional software industry.

In future works, we intend to validate the candidate anti-pattern mini-template definitions via survey with video game developers and managers. We will also expand the analysis to include software development anti-patterns (code smells).

ACKNOWLEDGMENTS

The authors were partly supported by the NSERC Discovery Grant and Canada Research Chairs programs.

REFERENCES

[1] Antonio Borrelli, Vittoria Nardone, Giuseppe A. Di Lucca, Gerardo Canfora, and Massimiliano Di Penta. 2020. Detecting Video Game-Specific Bad Smells in Unity Projects. In *Proceedings of the 17th International Conference on Mining Software Repositories*. ACM, Seoul Republic of Korea, 198–208. https://doi.org/10.1145/3379597.3387454

[2] Premek Brada and Petr Picha. 2019. Software process anti-patterns catalogue. In *Proceedings of the 24th European Conference on Pattern Languages of Programs - EuroPLop '19*. ACM Press, Irsee, Germany, 1–10. https://doi.org/10.1145/3361149.3361178

[3] William J Brown, Raphael C Malveau, Hays W McCormick Iii, Thomas J Mowbray, John Wiley, Robert Ipsen, and Theresa Hudson. 1998. *Refactoring Software, Architectures, and Projects in Crisis.* Robert Ipsen, New York. 157 pages.

[4] Amanda C. Cote and Brandon C. Harris. 2021. The cruel optimism of "good crunch": How game industry discourses perpetuate unsustainable labor practices. *New Media & Society* 0, 0 (2021), 14614448211014213. https://doi.org/10.1177/14614448211014213 arXiv:https://doi.org/10.1177/14614448211014213

[5] W. Cunningham. 2010. Management Anti Pattern Road Map. http://wiki.c2.com/?ManagementAntiPatternRoadMap. Accessed: 2021-12-01.

[6] J.P. Doran and M. Casanova. 2017. *Game Development Patterns and Best Practices.* Packt Publishing, none.

[7] Henrik Edholm, Mikaela Lidström, Jan-Philipp Steghöfer, and Håkan Burden. 2017. Crunch Time: The Reasons and Effects of Unpaid Overtime in the Games Industry. In *2017 IEEE/ACM 39th International Conference on Software Engineering: Software Engineering in Practice Track (ICSE-SEIP).* IEEE, Buenos Aires, Argentina, 43–52. https://doi.org/10.1109/ICSE-SEIP.2017.18

[8] Bill Elliott. 2007. Anything is possible: Managing feature creep in an innovation rich environment. In *2007 IEEE International Engineering Management Conference.* IEEE, none, 304–307. https://doi.org/10.1109/IEMC.2007.5235049

[9] Phillip A. Laplante and Colin J. Neill. 2006. *AntiPatterns: identification, refactoring, and management.* Auerbach Publications, Boca Raton. OCLC: ocm61240797.

[10] Fábio Petrillo, Marcelo Pimenta, Francisco Trindade, and Carlos Dietrich. 2009. What Went Wrong? A Survey of Problems in Game Development. *Comput. Entertain.* 7, 1, Article 13 (feb 2009), 22 pages. https://doi.org/10.1145/1486508.1486521

[11] Fábio Petrillo and Marcelo Soares Pimenta. 2010. Is agility out there?: agile practices in game development. In *Proceedings of the 28th Annual International Conference on Design of Communication, SIGDOC 2010, São Carlos, São Paulo state, Brazil, September 26-29, 2010*, Júnia Coutinho Anacleto, Renata Pontin de Mattos Fortes, and Carlos J. Costa (Eds.). ACM, none, 9–15. https://doi.org/10.1145/1878450.1878453

[12] Cristiano Politowski, Fábio Petrillo, João Eduardo Montandon, Marco Tulio Valente, and Yann-Gaël Guéhéneuc. 2021. Are game engines software frameworks? A three-perspective study. *J. Syst. Softw.* 171 (2021), 110846. https://doi.org/10.1016/j.jss.2020.110846

[13] Cristiano Politowski, Fábio Petrillo, Gabriel Cavalheiro Ullmann, Josias de Andrade Werly, and Yann-Gaël Guéhéneuc. 2020. Dataset of Video Game Development Problems. In *MSR '20: 17th International Conference on Mining Software Repositories*, Sunghun Kim, Georgios Gousios, Sarah Nadi, and Joseph Hejderup (Eds.). ACM, Seoul, Republic of Korea, 553–557. https://doi.org/10.1145/3379597.3387486

[14] Cristiano Politowski, Fábio Petrillo, Gabriel Cavalheiro Ullmann, and Yann-Gaël Guéhéneuc. 2021. Game industry problems: An extensive analysis of the gray literature. *Inf. Softw. Technol.* 134 (2021), 106538. https://doi.org/10.1016/j.infsof.2021.106538

[15] E. Yourdon. 2003. *Game Development Patterns and Best Practices.* Financial Times Prentice Hall, none.