Is Agility out there? Agile Practices in Game Development

Fabio Petrillo
Institute of Informatics - Federal University of Rio Grande do Sul (UFRGS)
9500 Bento Gonçalves Avenue
Porto Alegre, Brazil
fabio@petrillo.com

Marcelo Pimenta
Institute of Informatics - Federal University of Rio Grande do Sul (UFRGS)
9500 Bento Gonçalves Avenue
Porto Alegre, Brazil
mpimenta@inf.ufrgs.br

ABSTRACT

Game development is a very complex and multidisciplinary activity and surely the success of games as one of most profitable areas in entertainment domain could not be incidentally. The goal of this paper is to investigate if (and how) principles and practices from Agile Methods have been adopted in game development, mainly gathering evidences through Postmortem Analysis (PMA).

Then we describe how we have conducted PMA in order to identify the good practices adopted in several game development projects. The results are discussed, comparing similarities and differences on how these practices are taken in account in (traditional) software development and game development.

1. INTRODUCTION

The creation of electronic games is nowadays an incredibly complex task [12], much harder than someone might initially imagine [6].

The increased complexity, combined with the multidisciplinary nature of the process of game development (art, sound, gameplay, control systems, artificial intelligence, human factors, among many others) interacting with the traditional software development, creates a scenario which also increases this complexity. Some authors (for example [7]) recommend a methodology for taking in account software engineering expertise in the field of digital games.

However, despite these difficulties, the gaming industry is currently one of the most powerful in the entertainment industry, with billions of dollars in profit and creating trillions of hours of fun. Major game projects have cross-functional teams formed by highly skilled individuals, including software developers, designers, musicians, script writers and many others. Thus, the game developer career is currently one of the most dynamic, creative, challenging and potentially profitable that someone can choose [12]. This scenario suggests that the strength and profitability do not happen by chance. From the analysis of real projects' reports, it seems that to achieve these results, a common set of good practices were adopted in these projects.

The games industry can benefit tremendously by acquiring knowledge of software engineering, allowing developers to use good and proven practices. In fact, according to [11], a clear understanding of the tools available and how to apply them can enhance the results in game development.

In the traditional software industry, many papers and books have been published about good practices in software engineering [23, 30, 32]. However, are these practices also found in game development? Which practices are most prominent? How often these practices are found in game projects? What practices are found in both industries? Are there good practices found only in game development? Our intention in this paper is to discuss these issues.

The game development community has a vast literature, especially when it comes to technology issues. However, few works of software engineering dedicated to the electronic game industry, standing out above all the works [4] and [11]. It is interesting to note that these two are eminently philosophical view of the propagators of the waterfall development process (more details in section 2). In particular, the work of [4], one of the most quoted by the gaming community, demonstrating a rooted culture processes prescriptive and non-iterative. In this paper, [4] explicitly advocates the adoption of a "comfortable" subset of the Unified Process (UP - Unified Software Development Process) as a process of development of electronic games, for the simple fact of being a standard in he software industry.

The aim of this paper is to investigate whether (and how) some agile principles and practices [15, 18] have been applied in game development. This research can help demystify the impression that the adoption of agile practices by game developers is a difficult process. Indeed, if many of the agile practices are already being (even partially) taken, we believe that many developers may try to better understand the fundamentals of agile software development and more easily find ways to put it into action in their daily work.

The paper is structured as follows: after this introduction, section 2 presents a summary of the games development process. Section 3 discusses the best practices of software engi-

SIGDOC 2010, September 27–29, 2010, S.Carlos, SP, Brazil.

neering found in process of game development, from analysis of the literature on game development and mainly through the analysis of postmortems. Section 4 discusses the results of these analysis. Finally, the conclusions are presented in section 5.

2. THE TRADITIONAL GAME DEVELOPMENT

The game development community has a vast specialized literature, especially about technical issues. The industry also has a series of titles that describe how to teach and work in a games company, such as Gershenfeld et al. [12].

However, not many software engineering works are dedicated to the gaming industry: two remarkable exceptions are Bethke [4] and Schofield [11]. It is interesting to emphasize that these two works are related to a *waterfall* philosophical viewpoint, in particular the work of Bethke [4], one of the most cited by the games community, demonstrating the existing culture of prescriptive and non-iterative procedures. Bethke [4] explicitly advocates the adoption of a subset of the *Unified Software Development Process* as the development process for electronic games, for the simple fact that this is a "standard" in the software industry.

According to [10], it is possible to identify a common cycle of development in different teams: basically, the waterfall model [23] adapted to produce games. In particular, two stages of this model may be highlighted: a) defining the game rules and b) the production of **Document Set** or **Project Document**.

The rules of a game determine the behavior and interaction among the characters and their environment, and can be viewed as requirements. In fact, game designing is nothing more than creating a set of rules [9], usually defined in an informal process, involving all members of the development team [26].

The Project Document (*(*Design Document)) is the main, often the only, documentation of a game. Your goal is to describe and detail the mechanics of the game, in other words, what the player is able to do within the environment of the game, as he is able to do it and how it can lead to a satisfactory experience. It also usually includes the main components of the story and describes the scenarios in which the game is set, supporting the description of the player's actions. Many developers refer to it as a functional specification, using it as a basis for the architectural design [24].

The creation of a solid project document is considered, traditionally, as the most important step in the game development. The difficulties in creating this document are caused, mainly, by the nature of the task and the tools used in their conception, it is not possible, for example, document the "fun" [19]. The development team could use his experience and intuition in defining the mechanics of the game, but the quality of entertainment provided by the game in general can only be assessed in stages of testing.

The game modeling is not changed in recent years and continues based on narrative techniques, like scripts and storyboards, borrowed from other entertainment media, such as cinema [17]. The conception of a game can also count on a higher abstraction level, organized in the Concept or Proposal Document. This document can analyze aspects such as market, budget, schedule, technology, art style, profile of the development group and some high level description of the gameplay. However, the preparation of the Concept Document is not common to all projects [24].

The development process used in the production of most games is based on the waterfall model [25]. This process consists of phases that are executed sequentially, in which each one generates a product and is independent from the others. The features inherent in producing games require some adjustments to the classic process, which can be showed in figure 1.

Some authors argue that the waterfall model, while serving as a common denominator between the cycles exist, **not to be fully implemented**. Typical features of a game, as dynamics of design or difficulties in planning the gameplay, make it difficult - or impossible - to specify completely a game without writing any code, that is, without a version of the system in which the project can be tested. This results in development cycles that involve links between the stages of specification of the game and test procedures. [25] quotes a cycle of incremental development, the Staged Delivery Model, as an alternative to the waterfall model.

3. PMA TO IDENTIFY GOOD PRACTICES

In this section, we describe how we have gathered evidences of adoption of good practices in game industry. The Postmortems Analysis (PMA) is a technique borrowed from Empirical Software Engineering. Empirical Software Engineering aims to investigate and collect relevant data to generalize the results, learning from mistakes and successes, providing a future reuse of this knowledge.

Controlled experiments, case studies and surveys approaches are most commonly used and better known than the PMA [29, 33, 28]. The main reason for the PMA not be adopted in software projects in different fields of application is not their difficulty of operationalizing, neither the resources involved or much less cost, but rather the absence of postmortems.

The term *postmortem* designates a document which summarizes the project development experiences, with a strong emphasis on positive and negative aspects of the development cycle [14]. It is commonly done right after the project finishes, by managers or senior project participants [7]. In a software engineering viewpoint, postmortems are important tools for knowledge management [5], from which the group can learn from its own experiences and plan future projects. In fact, the postmortem analysis can be so revealing that some authors [5] argue that any project should be finished without it's own postmortem.

Traditionally, IT teams have no habit of creating postmortems, although there are obviously exceptions (see for example [31]) and strong recommendations to do it [5]. Fortunately, a rare exception is the domain of games.

Postmortems are much used in the game industry. Many game websites devote entire sections to present these doc-

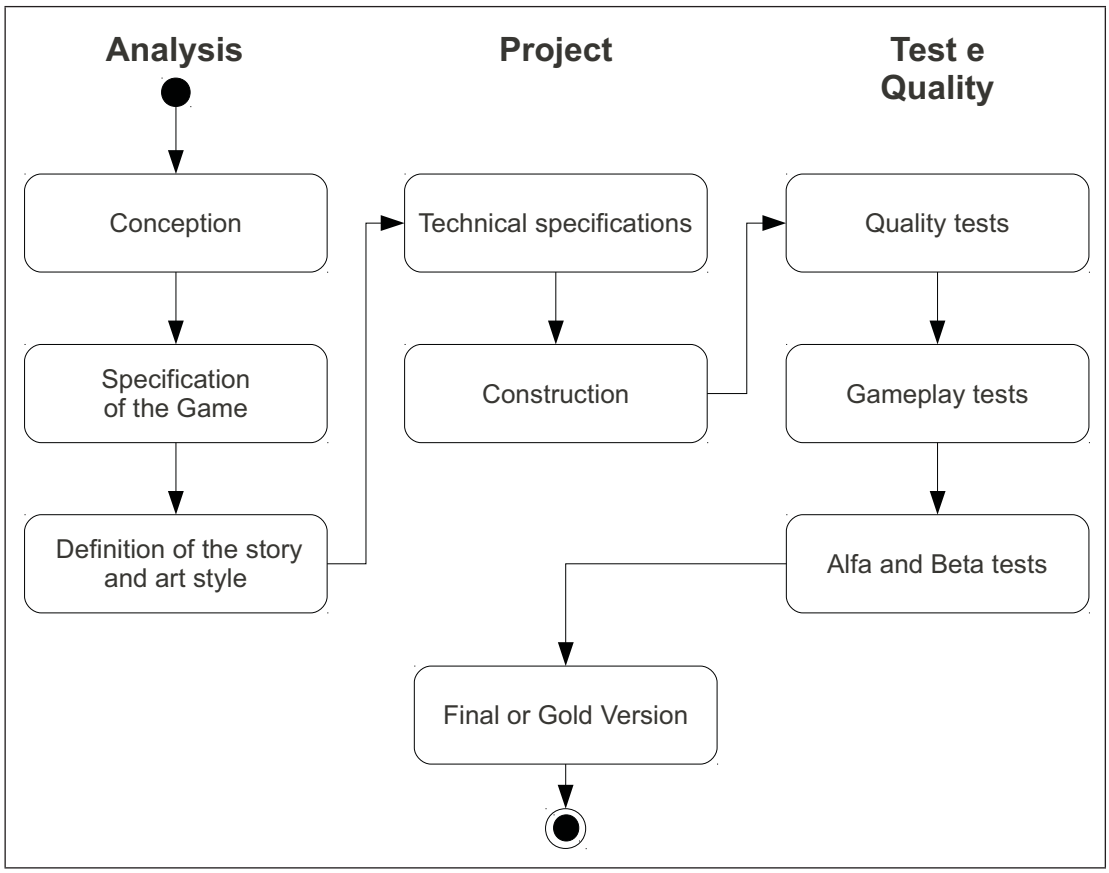

Figure 1: Waterfall process applied to game development

uments, such as *Gamasutra* (`http://www.gamasutra.com`) and *Gamedev* (`http://www.gamedev.net`). It is also very interesting to note the variety of development teams profiles and projects behind these documents, varying from few developers in small projects to dozens of developers in five-year-long projects.

The postmortems published by *Gamasutra* mainly follow the structure proposed by the *Open Letter Template* [20], which is composed by three sections. The first section summarizes the project and presents some important aspects of the development. The next two sections, however, discuss the most interesting aspects to the game developers:

- **What went right:** it discusses the *best practices* adopted by developers, solutions, improvements, and project management decisions that have improved the efficiency of the team. All these aspects are critical elements to be used in planning future projects.
- **What went wrong:** it discusses difficulties, pitfalls, and mistakes experienced by the development team in the project, in both technical and management aspects.

The postmortem is closed with a final message from the author, commonly followed by a project technical brief, which includes the number of full- and partial-time developers, length of development, release date, target platforms, and the hardware and software used in the development.

The information contained in the postmortems constitute knowledge base that can be reused by any development team, which includes examples and real life development experiences. They can help in knowledge sharing, and can be very useful for planning future projects.

Thus, the 20 *postmortems* were examined to find good practices. The process of recognizing the good practices took place in 3 phases. At first, each *postmortem* has been read and the quotes that were deemed relevant were highlighted. In the second stage, based on traditional and agile software engineering knowledge [1, 23, 3], the practices to be tabulated were selected. In the third stage, each report was read again, with special attention to highlighted passages, and each citation was tabulated according to the classification previously made.

To perform the postmortems analysis and the organization of good practices, we need to select some postmortems , analyze them and compile the results, forming a set of practices that are approved or disapproved by these developers. We performed the same procedure (analysis of literature and analysis focused on the postmortems) to carry out a diagnosis of the major problems and difficulties encountered in the development of games (see [21]).

Among the various existing postmortems, we selected 20 postmortems posted on Gamasutra, listed in tables 1 and 3. The 20 postmortems analyzed were selected randomly, having one important criterion for the selection: we selected postmortems of projects that resulted in a complete game and delivered to market, not been analyzed reports of projects failed, canceled or terminated without a product.

In preparation stage, we studied the main agile practices [8, 1, 3, 22] and good practices in the gaming industry [27, 13]. In this study, 12 good practices were listed for analysis.

The 20 postmortems were read and sentences quoting practices were highlighted. During this stage, another practice was found ("Belief in the success of the project"), which was added to the analyzed set, totaling 13 practices, which are listed in Tables 1 and 2 and also in Figure 2. Finally, a table for data collection was organized (table 1), with projects arranged in rows and practices in columns.

4. DISCUSSION

In order to somehow quantify the practices found in the postmortems, the number of "Yes" occurrences was recorded both in terms of lines as of columns. This way, the quantity of "Yes" found in the lines represents how many different types of practices were presented in a certain game. As for the number recorded in columns, it represent the number of occurrences of this practice in a set of analyzed games. The count of columns, which can be seen in the penultimate line, made possible to organize them in order of occurrence. The last line contains the percentage of occurrences in relation to the number of projects studied.

When we look more carefully to these results, we can see that the most common practices were a **qualified, motivated or cohesive team** or the **belief in the project's success** with 90% (18 out of 20) of the projects reporting these 2 practices. Then, the **stimulus to creativity** and **focus on the product** were highlighted, with 80% of the project citing them (16 out of 20). Even in that context, the next most common 2 practices were **source version control**, with 65% (13 out of 20) and the **utilization of simple or productive tools**, with 60%. Figure 2 shows the good practices occurrence histogram in descending order, where a graphic comparison of these results can be made.

Despite these results being not be surprising, they show the importance of the team for the project success, confirming the statements made by Bach [2] that processes are useful, but not the central elements for the success of software projects. The central point is the man - the hero who solves ambiguous problems, which distinguishes between an expressed need and the one which will actually make the client fully satisfied [2]. Moreover, it is clear the persistence and belief in the success of teams who intensely focused on the software product.

Another interesting point is that management practices are a clear problem in the games industry. It does not reach 50% the number of projects in which a **defined process** is adopted and only 25% (5 of 20) of the projects adopted **good management practices**. Moreover, only 40% of the projects used practices of quality control and was 10% (2 to 20) the occurrence of **continuous integration** found the analyzed projects.

If we calculate the average number of good practices adopted in the analyzed projects, we will see that it is 7.2 practices, with standard deviation of 2.6. This means that most projects employed between 5 and 11 practices, and that the average is about **half** of the examined practices (7 to 13). This result shows that the game industry adopts a considerable number of good practices in their projects, undoing a little pessimistic view of some authors, as [4] and Flood [10].

If we analyze these results, we can see that in general, **good practices adopted in the traditional software industry are also found in the games industry**. In both studies, the quality of the team was dominant for the success of the project, as well as the concern with the adoption of good practices for programming.

The game industry, informally, has employed best practices in software development, as presented in section 3. We can combine the agile practices analyzed in accordance with the agile practices of Scrum, XP and Agile Modeling (AM), forming the Table 2.

Good practice already adopted	It adheres to the method...
Qualified team	AM, Scrum, XP
Belief in the success of the project	Scrum, XP
Creativity stimulus	Scrum, AM
Focus on the product	XP, Scrum
Version control	XP
Using simple tools	AM, XP
Programming good practices	XP
Agile modeling	XP, AM
Defined process	Scrum, XP
Quality control	XP
Feedback quickly	XP, Scrum
Good practices of management	XP, Scrum
Continuous integration	XP

Table 2: Adherence of best practices already adopted in games to agile methods

Game development teams are adopting agile practices instinctively. This scenario - the deployment of agile methods like Scrum and XP - can occur naturally, since the teams already use several principles of agility in their routines. Thus, the table 2 can point to the adoption of agile methods related to good practices analyzed in this work.

A new analysis can be offered to collate the results presented by [21], which outlines the problems encountered with the analysis results of the section 3, which contains good practice raised in postmortems. In evaluating, for example, the project *Cel Damage*, who took **85%** of good practice, we observed a lower incidence of problems, with only **20%** of reported problems. Are there a linear correlation between the number of problems encountered and good practices adopted in game projects?

For this analysis, was prepared to table 3, which is made up of the game projects analyzed, the percentage of problems found and the percentage of good practices adopted.

The management practices are a clear deficiency in the game industry. Not reach 50% the number of projects in which we identified the adoption of a **defined process of work**

Table 1: Occurrence of good practices in projects

Game	Qualified team	Belief in the success of the project	Creativity stimulus	Focus on the product	Version control	Using Yesple tools	Programming good practices	Agile modeling	Defined process	Quality control	Feedback quickly	Good practices of management	Continuous integration	Total	% good practices found
Beam Runner Hyper Cross	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	9	69%
Gabriel Knights	Yes	No	No	No	Yes	No	No	No	No	Yes	No	No	No	3	23%
Black & White	Yes	Yes	Yes	Yes	No	No	No	Yes	No	Yes	No	No	No	6	46%
Rangers Lead the Way	No	No	No	No	No	Yes	Yes	No	No	No	No	No	No	2	15%
Wild 9	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	No	No	No	No	7	54%
Trade Empires	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	12	92%
Rainbow Six	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	No	No	No	5	38%
The X-Files	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	No	No	No	5	38%
Draconus	Yes	Yes	Yes	No	No	No	Yes	No	No	No	No	No	No	4	31%
Cel Damage	Yes	Yes	No	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	Yes	11	85%
Command and Conquer: Tiberian Sun	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	No	No	No	No	8	62%
Ashteron's Call	No	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	Yes	No	No	Yes	9	69%
Age of Empires II: The Age of Kings	Yes	Yes	Yes	No	Yes	Yes	No	No	No	Yes	No	Yes	No	7	54%
Diablo II	Yes	Yes	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Yes	No	No	9	69%
Operation Flashpoint	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No	Yes	No	No	10	77%
Hidden Evil	Yes	Yes	Yes	Yes	Yes	No	No	No	No	Yes	No	No	No	6	46%
Resident Evil 2	Yes	Yes	No	Yes	No	Yes	No	Yes	Yes	No	Yes	Yes	No	8	62%
Vampire: The Masquerade	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes	Yes	No	No	No	No	8	62%
Unreal Tournament	Yes	Yes	Yes	Yes	No	No	Yes	Yes	No	No	Yes	Yes	No	8	62%
Tropico	Yes	Yes	Yes	Yes	Yes	Yes	No	No	No	No	No	No	No	6	46%
Occurrences	18	18	16	16	13	12	11	9	9	8	6	5	2	**143**	**7,2**
%	90%	90%	80%	80%	65%	60%	55%	45%	45%	40%	30%	25%	10%	**0,55**	**55%**

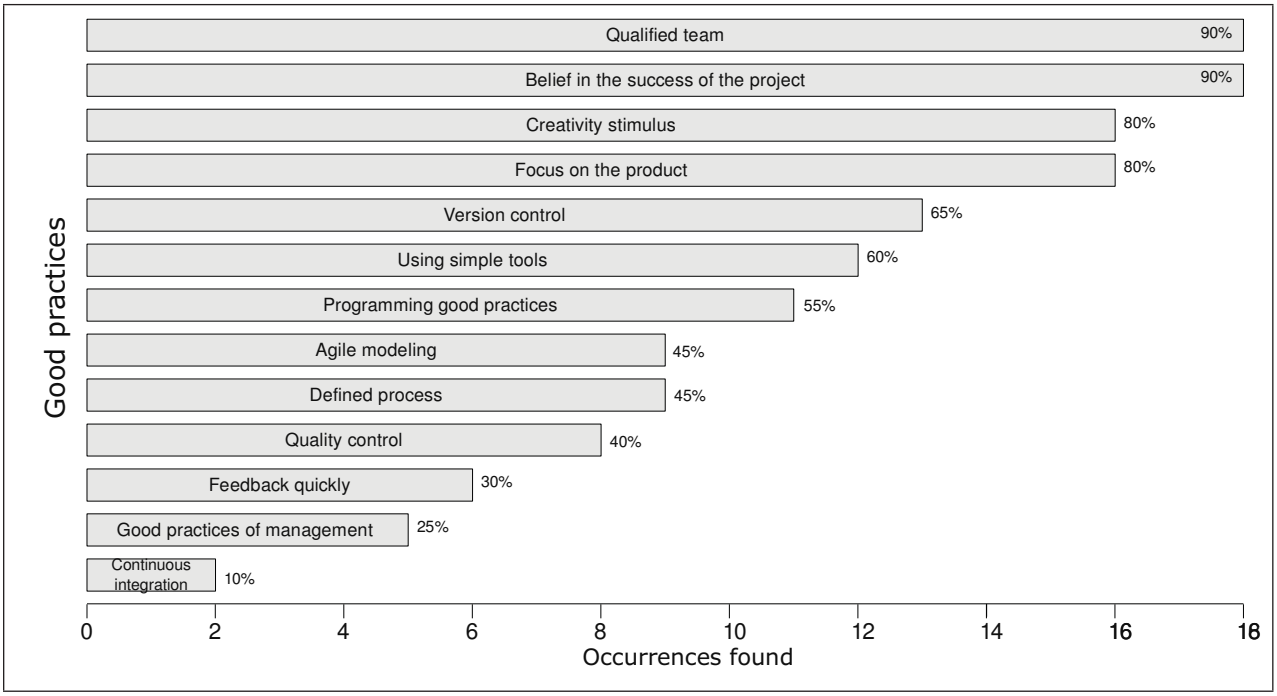

Figure 2: Occurrence of good practices

Project	% of problems found in project	% of good practices found in project
Cel Damage	20.0%	84.6%
Trade Empires	26.7%	92.3%
Diablo II	33.3%	69.2%
Command and Conquer	33.3%	61.5%
Tropico	26.7%	46.2%
Age of Empires II	33.3%	53.8%
Resident Evil 2	40.0%	61.5%
Beam Runner	46.7%	69.2%
Asheron's Call	46.7%	69.2%
Hidden Evil	33.3%	46.2%
Vampire	46.7%	61.5%
Unreal	46.7%	61.5%
Wild 9	53.3%	53.8%
The X-Files	40.0%	38.5%
Operation Flashpoint	80.0%	76.9%
Black & White	60.0%	46.2%
Rainbow Six	73.3%	38.5%
Draconus	80.0%	30.8%
Rangers Lead the Way	46.7%	15.4%
Gabriel Knights	86.7%	23.1%

Table 3: Comparison between the good practices and problems found by projects

and only 25% (5 of 20) of projects adopted **good management practices**. In addition, only 40% of projects used quality control practices and was 10% (2 of 20) recorded the occurrence of the **continuous integration**.

5. CONCLUSION

The major contribution of this work was helping to organize the universe of agile practices for the domain of games, from the analysis of problems and good practices found, being a point of departure for the use of agile methods in the process development of electronic games.

If we think that all major problems of the traditional software industry are also found in the games industry (see [21], we can note that they are correlated and the solutions adopted in the traditional software industry also should be investigated for the gaming industry.

Even informally, the game development teams are adopting a set of agile practices . In this scenario, the deployment of agile methods like Scrum and XP, can occur naturally, since the teams already adopt in their activities several principles of agility.

Our discussion of the results described in this article lead us to believe that game developers often **not** adopted **deliberately** these good practices because are **Agile Practices**. Perhaps, like many software developers, they think - because they are informal - do not correspond to the notions of rigor and systematization usually associated with software engineering.

However, these practices are studied and applied increasingly by the community of software engineering, although not always recognized and disclosed as disciplined ways of solving the chronic problems of development. As we have seen, many of the agile practices are already being adopted (even partially), and we believe that developers - if they are willing to better understand the fundamentals of agile software development - can easily incorporate them into their activities.

Since there are few academic studies on the use of agile methods in game development, this work opens perspectives for further research. We believe for example that modeling (preferably agile) of complex elements such as fun gameplay and fun deserves more investigation.

Based on studies done by [34], [16] and [13] we believe that the adoption of agile practices in development of games can achieve promising results. The agile practices seem to include the best features of the game industry, as multidisciplinarity and the difficulties in modeling aspects like user experience, the fun and enjoyment, whereas empirical methods are more adaptive and react better to changes during project implementation. We hope that this paper is a contribution to increasing adoption of agile practices in the process of game development.

6. REFERENCES

[1] S. W. Ambler. *Modelagem Ágil*. Bookman, São Paulo, 2004.

[2] J. Bach. Enough about process: what we need are heroes. *IEEE Software*, 12(2):96–98, março 1995.

[3] K. Beck. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional, Reading, MA, 1st edition, 1999.

[4] E. Bethke. *Game Development and Production*. Wordware Publishing, Plano, 2003.

[5] A. Birk, T. Dingsoyr, and T. Stalhane. Postmortem: never leave a project without it. *IEEE Software*, 19, maio/junho 2002.

[6] J. Blow. Game development: Harder than you think. *ACM Press Queue*, 1(10):28–37, February 2004.

[7] D. Callele, E. Neufeld, and K. Schneider. Requirements engineering and the creative process in the video game industry. In *13th IEEE International Conference on Requirements Engineering*, August 2005.

[8] A. Cockburn. *Agile Software Development*. Addison Wesley Longman, Reading, MA, 1st edition, 2000.

[9] D. Cook. Evolutionary design - a practical process for creating great game designs. *GameDev.net*, janeiro 2001.

[10] K. Flood. Game unified process. *GameDev.net*, Maio 2003.

[11] J. P. Flynt and O. Salem. *Software Engineering for Game Developers*. Software Engineering Series. Course Technology PTR, 1st edition, November 2004.

[12] A. Gershenfeld, M. Loparco, and C. Barajas. *Game Plan: the insider's guide to breaking in and succeeding in the computer and vieo game business*. St. Martin' s Griffin Press, New York, 2003.

[13] A. Gibson. Agile game development and fun. Technical report, University of Colorado Department of Computer Science, 2007.

[14] W. Hamann. Goodbye postmortems, hello critical stage analysis. *Gamasutra - The Art & Business of Making Games*, julho 2003.

[15] J. Highsmith and A. Cockburn. Agile software development: The business of innovation. *IEEE Computer*, 34:120–122, 2001.

[16] L. C. C. Kasperavicius, L. N. M. Bezerra, L. Silva, and I. F. Silveira. Ensino de desenvolvimento de jogos digitais baseado em metodologias Ágeis: o projeto primeira habilitação. In *Anais do XXVIII Congresso da SBC - Workshop sobre Educação em Computação*, pages 89 – 98, Belém do Pará, Julho 2008.

[17] B. Kreimeier. The case for game design patterns. Gamasutra - The Art & Business of Making Games, `http://www.gamasutra.com/features/20020313/kreimeier\_01.htm`, March 2002.

[18] M. Marchesi, G. Succi, D. Wells, and L. Williams. *Extreme Programming Perspectives*. Addison Wesley, 2002.

[19] M. McShaffry. *Game Coding Complete*. Paraglyph Press, Scottsdale, 2003.

[20] M. Myllyaho, O. Salo, J. Kääriäinen, J. Hyysalo, and J. Koskela. A review of small and large post-mortem analysis methods. In *IEEE France*, Paris, November 2004. 17th International Conference Software & Systems Engineering and their Applications.

[21] F. Petrillo, M. Pimenta, F. Trindade, and C. Dietrich. What went wrong? a survey of problems in game development. *ACM Computer in Entertainment*, CIE: 7(1), 2009.

[22] M. Poppendieck and T. Poppendieck. Principles of lean thinking. 2003.

[23] R. S. Pressman. *Engenharia de Software*. McGraw-Hill, São Paulo, 6th edition, 2006.

[24] R. Rouse. *Game Design: theory & practice*. Wordware Publishing, Inc, 2001.

[25] R. Rucker. *Software Engineering and Computer Games*. Addison Wesley, December 2002.

[26] E. Schaefer. Postmortem: Diablo ii. *Gamasutra - The Art & Business of Making Games*, outubro 2000.

[27] B. Schofield. Embracing fun: Why extreme programming is great for game development. Gamasutra: The Art & Business of Making Games, March 2007.

[28] F. Shull, M. Mendonça, V. Basili, J. Carver, J. C. Maldonado, S. Fabbri, G. H. Travassos, and M. C. Ferreira. Knowledge-sharing issues in experimental software engineering. *Empirical Software Engineering*, 9(1-2):111–137, 2004.

[29] F. Shull, J. Singer, and D. I. Sjøberg. *Guide to Advanced Empirical Software Engineering*. Springer-Verlag, London, 2008.

[30] I. Sommerville. *Software Engineering*. International computer science series. Addison-Wesley, London, 6th edition, 2001.

[31] T. Stalhane, T. Dingsoyr, G. K. Hanssen, and N. B. Moe. *Post Mortem? An Assessment of Two Approaches*, chapter Empirical Methods and Studies in Software Engineering, pages 129–141. Springer Berlin / Heidelberg, 2003.

[32] F. Tsui and O. Karam. *Essentials of software engineering*. Jones and Barlett Publishers, São Paulo, 6th ed edition, 2007.

[33] C. Wohlin, M. Höst, and K. Henningsson. *Empirical Methods and Studies in Software Engineering*, chapter Empirical Research Methods in Software Engineering, pages 7–23. Springer Berlin / Heidelberg, 2003.

[34] S. Xu and V. Rajlich. Empirical validation of test-driven pair programming in game development. *Computer and Information Science, 5th IEEE/ACIS International Conference on*, 0:500–505, 2006.