Game-Scrum: An Approach to Agile Game Development

André Godoy
Ellen F. Barbosa
Instituto de Ciências Matemáticas e de Computação (ICMC-USP)
PO. Box 668, Zip Code 13560-970 – São Carlos (SP), Brazil

Abstract

The use of agile methodologies for developing games has become very common. However, such methodologies must be adapted to the team reality and to the particularities of the game development. Since there are few agile methodologies that specifically address the problems found in game development, this paper aims at investigating some alternatives and suggests an approach to guide people who are starting in this area. The proposed methodology, named Game-Scrum, has been applied in the development of a game for teaching software engineering. The results from its application are also discussed in the paper.

Keywords:: Game Development, Agile Methodologies, Scrum, XP, Educational Game

Author's Contact:

andregodoy@grad.icmc.usp.br
francine@icmc.usp.br

1 Introduction

The recent advances in technology enabled the development of increasingly more complex and realistic games, but the cost of production of a top-level game has reached millions [Wikia 2009], leaving publishers more and more risk averse. This creates a demand in the use of techniques and methods to ensure that a game can be developed with as minimal as possible delays and mistakes. Such techniques and methods should also facilitate the process of discovering "the fun" – a important factor for the success of a game.

Although some people may already have knowledge in software development, there are specific features of game development that can prevent the success of great games. This results in major problems in project management, rising the already high costs and causing delays. The use of a methodology focused on game development may help to avoid those problems. In this way, the use of some iterative method is increasing in this industry [Gregory 2008], and agile methodologies have become popular.

Unfortunately, there are few works in the current literature on agile methodologies oriented to game development, and the existing ones have not been effectively used in practice. The objective of this paper is to present an approach of an agile methodology based on Scrum and eXtreme Programming for game development, and review two existing agile methodologies with this same orientation.

This paper is organized as follows. Section 2 will see some of the of agile methodologies most known, and section 3 will review some proposals specific for games available today. Section 4 will deal with the most common problems in game development, while an approach to solve some of these problems will be threated in section 5. Section 6 will describe and analise the application of the suggested approach while developing an educational game.

2 Agile Methodologies

As the methodologies discussed in the sections below are based primarily on Scrum and XP, this section will deal on these methodologies in order to distinct their practices from the approach suggested section 5. According to Abrahamsson [Abrahamsson et al. 2003], the main characteristics of agile methodologies are: cooperation, simplicity, adaptiveness and being incremental.

Within these characteristics, the Agile Manifesto [Beck et al. 2001] brings together the main values of various agile methodologies, which are important to understand an approach to game development based on an agile methodology. Through these values we can notice that the focus of agile methods is on the product, rather than the process used. Thus, we avoid unnecessary documentation, there is flexibility to accept and react to changes, and also the customer collaboration in monitoring the development.

One of the best known agile software development is eXtreme Programming (or XP), which is proposed by Beck [Beck 1999] and whose fundamental beliefs are based on five principles or values [Teles 2006]: communication, courage, feedback, respect and simplicity. Its practices were designed aiming to satisfy these values, with complete freedom to use them or not. Who defines what and how practices will be applied in the project is the coach, that is also the project leader and responsible for helping to maintain the dynamics and facilitate the process for the team, providing resources and coordinating activities, thus ensuring that his team correctly applies the practices defined.

Scrum [Schwaber and Beedle 2001] is another agile methodology also based on short iterations called sprints. In the sprints, the backlogs must be delivered, which are a set of atomic requirements designed to be done at each iteration. The backlogs should be prioritized according to customer value, production cost, risk and knowledge required to produce it or other parameters. The Scrum Master act as a servant leader to the Team, which is responsible for the development. There is alsso the Product Owner, who gives the feedback of the project. The iterations are accompanied by four kinds of meetings: one for planning, one for accompaniment, one for review what was done and one for inspect the process, people and tools.

Discover soon what in the game will provide the "fun factor" enables the team to focus on developing and improving it over a much larger part of the project - greatly increasing the likelihood of success of the game. A simple solution to this problem is adopt iterative methodologies, in which the agile ones are based. These methodologies allow to receive an early feedback of the features implemented, thus improving them if necessary is easier. Another advantage is to facilitate communication and cooperation among all those involved in the game creation.

3 Problems in Game Development

The problems found in software development in general are well known, but very little is known about the real issues affecting the gaming industry. According to Petrillo [Petrillo et al. 2008] in the literature, the main problems reported are related to scope, planning and crunch time (a term used for periods of extreme work overload, occurring usually closer to deadlines). After a survey of games postmortems with teams and projects of varying sizes, it was discovered that the vast majority of problems encountered was related to management problems.

Part of these problems are often associated with the multidisciplinary team. Although multidisciplinarity creates an environment that encourages creativity it may end up creating a division between "the artists" and "the developers", resulting in difficulty to get an effective communication among all team [Petrillo et al. 2008]. Another point is the difficulty in determining certain elements like the fun of the game, which has no efficient technique that describes when this goal is achieved [Petrillo et al. 2008].

Some techniques for solving these problems are suggested by Kanode and Haddad [Kanode and Haddad 2009], such as the use of iterative methods, building prototypes and creating a pipeline to the

artistic content of the game. For a detailed info about these suggestions, please refer to Kanode and Haddad's work [Kanode and Haddad 2009]. Part of the problems commonly encountered can be solved by applying the methodology proposed in the next section.

4 Related Work

Very little is said about agile methodologies specifically focused on game development. Most of the existing initiatives use some derivation of the waterfall method [Flood 2003] or, more recently, some iterative method [Gregory 2008]. Some proposals involving existing agile methodologies include eXtreme Game Development [Demachy 2003] and Game Unified Process [Flood 2003].

4.1 eXtreme Game Development

eXtreme Game Development [Demachy 2003], or XGD, is an adaptation of XP for game development. According to Demachy, its focus is on how to adapt XP to game design and creation of multimedia content and how to automatically test specific elements of the game, like the "fun factor". Some adaptations from XP include to add multimedia content in continuous integration; make tests for multimedia content; consider the team as a whole and use UML to describe the elements of game design and interaction and communication between game designers, programmers and even artists;

The methodology clarifies some adaptations from XP to XGD, but does not address, however, how to work with multidisciplinary teams. It only recommends the vision of the team as a whole, and suggests paired work for artists such as pair programming is used for developers in order to accomplish this. Also, it does not discuss the results of applying the methodology in the projects mentioned.

4.2 Game Unified Process

Game Unified Process [Flood 2003] is a methodology that combines the RUP (Rational Unified Process) model and techniques of XP and was developed primarily to address issues of problems in game development using the waterfall model. Although considered to be a heavy process, RUP adopts a methodology that enables the occurrence of fewer changes in the software, while the XP provides practical ways to avoid over-documentation.

Of the preliminary conclusions that Flood obtained from the application of his methodology, we emphasize two: the focus on a rapid cycle of XP with a focus on long cycles of RUP that has brought benefits to the development team, despite the difficulties of the team in order to adopt the style of development of an agile methodology; and the recognition that the creation of content in game development is iterative, leading Flood to suggest this methodology also for artists and designers. Despite the positive feedback, the team result was compared with the waterfall method, which is not recommended as a method for game development, as Flood comments in his proposal. Also, how to handle with artists in iterations is not discussed.

5 Game-Scrum

According to Schwaber [Schwaber 2009], Scrum is a framework focused on project management - how to divide and coordinate the tasks so that everything can be done without impediments, under which you can use any other agile practices. In this view, XP would be more focused on the engineering of the project - which techniques are best to complete tasks efficiently. Game-Scrum uses these two methodologies as basis, adapting them with the experience of professionals and focusing in people with little or no experience in game development. As project management is a major falling point in game development, is emphasized that the team understand the iterations and meetings of Scrum, since these details will not be covered in this part, but remain important to the success of the project.

During the stages of development of an agile methodology, all the iterations have approximately the same distribution of the basic steps for developing a software. But according to Keith [Keith 2010], the distribution of work in game development is not equally distributed during its iterations. To support this reality, Game-Scrum is divided into three phases, discussed below.

5.1 Pre-production

In this phase the game must be "discovered", that is, what really are the game objectives and what would be the "fun factor" in it. Here the game concept will be improved, the programming language, platform and others definitions. As Kanode and Haddad state [Kanode and Haddad 2009], "great pre-production reduces the need to find that elusive element of 'fun' during the production stage, and allows the team to focus on implementing the game, rather than experimenting with it".

This phase defines the direction that the Production phase will take, without stifling the creativity that might emerge there. Here the work will be to find the ideal game concept and design, often through trial and error. For this to be accomplished, brainstorming is recommended in order to develop ideas and aggregate new ones. Developing a prototype also will help to preview the "fun factor" that the game or a portion of it can offer. The prototype must provide a basic and simplified navigation for the user and the features required to test. Usually, the code produced is not used again, and the prototype is discarded.

5.1.1 Game Design Document

The creation of the game design document is an important step to be accomplished in the pre-production phase of the game, being responsible for guiding the project's scope (and thus the entire development and testing of the game). A poor game design document may result on feature creep, and as a consequence, delays and missed milestones may also occur.

Although there is no standard way to build a game design document, Schuytema [Schuytema 2008] states that this document must have a comprehensive description of the game in all its aspects, so that the development team can build it. While describing the items, objects and characters in the game, it should be documented not only what they do, but also what they affect, how they interact and their role and behavior in the game. Despite the efforts to make the document fairly complete, the document still may change. However, one should evaluate the risks of changes and if the deadlines can still be met.

As this document will be later translated to a Product Backlog in the production phase, for small games it may be optional, translating the requirements directly as a Product Backlog. This may save time from the team as they go faster to the production, but may also increase risks of feature creep or not a very entertaining game.

5.2 Production

At this stage, one should already have a well defined project scope, and thus a good idea of what the game is really about and what actually should be done. Here the game design document should be translated in a product backlog. And at each iteration the most important backlogs remaining should be divided in smaller pieces and have its tasks defined, if not already.

For teams composed with many artists, it is needed to think in a better approach. Very often an artistic task is usually performed by several specialists working in some kind of production line. For instance, the modeler delivers his work to the animator, who passes it to the sound designer and so on. Keith [Keith 2010] suggests the use of Kanban for those responsible for creating artistic game content. This allows that at the end of a sprint there is still work that is under development, unlike Scrum, which requires that all work be done at the end of a iteration. The goal is to achieve a continuous flow in content creation.

Another practice suggested by Keith is time-boxing, limiting the time available to complete a particular task, taking into account the importance of the artifact being produced and its usefulness in the game. For teams composed for only developers or with few artists,

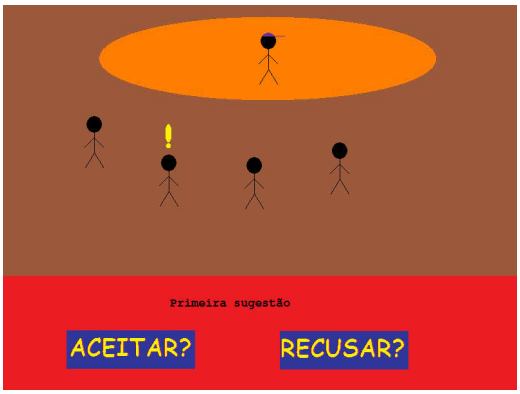

Figure 1: *The development of EngReq's prototype*

Figure 2: *The appearance of the final game*

once it is known what to build in the game, it is recommended to apply the Scrum management and XP techniques that best fit in the team reality.

5.3 Post-Production

After the game be completed, playtesting helps to ensure the quality and the fun of the game. But here we will focus on the feedback provided by the whole process, creating a postmortem. The postmortems are important because they allow to know the strengths and weaknesses of the development process used, problems that occurred and suggestions for improvements. Through this feedback one can get a better estimate for future projects and could make the necessary adjustments to the process with greater assertiveness.

Myllyaho [Myllyaho et al. 2004] describe the benefits and advantages of postmortems analysis, and also talks about the postmortems available in Gamasutra's website (`http://www.gamasutra.com/`). As described in their work, usually postmortems provides a summary of the project description, identifying the lessons learned in the three aspects: 'the good', containing a list of good solutions, improvements, suitable tools, etc., including areas of project management, communication, marketing and engineering practices; 'the bad', with a list of key issues and problems and the main cause of them; and 'the ugly', for a list of critical issues that absolutely must be corrected.

Also, the postmortems typically include the following game project metrics in addition to the experiences: publisher and developer; number of full-time developers, part-time developers, and contractors; length of the development, and release date; game platforms, and development hardware and software used.

Although the approach proposed by Game-Scrum do not bring new elements to the game development, its innovation is to join together several suggestions with the same goal. This allows to create a systematic method to develop a game, having not only practical references from those working in the area, but also the support of researchers confronting experiences with the knowledge available in literature.

6 Example of Application

For the sake of illustration, in this section Game-Scrum is applied to the development of EngGame, an educational game composed of several minigames in the domain of software engineering. The first of these minigames, called EngReq, aims to assist in learning about the requirements document for students of software engineering. Although not quite worked, the idea of EngReq already existed, but was necessary to develop the game from that idea.

6.1 The Pre-Production

So, in the first brainstorming it was discussed the idea of presenting several minigames themed in different ages for the player, each addressing a specific part or concept of software engineering. EngReq would occur in the stone age, and the player's objective would be of helping the people of his community to accept the requirements needed for a particular task that the community wishes to conclude, and rejecting them if they are not a valid requirement.

After, a prototype was made as a way of testing the ideas discussed during the brainstorming, and if was possible to create a game about software engineering that does not happens inside an office. In the prototype (Figure 1), it was made a drawing to represent the idea of the scenario of the game, and some screen transitions and the presentation of requirements for the player. It was discovered that although it seemed to be a good idea at first, the elaboration of tasks with similar requirements of a software system in the stone age was very difficult, so it was decided in another brainstorming that the game would occur in middle age.

As it was a simple minigame that would to be developed over a relatively short period, it was not written a game design document. As though the game is composed of minigames that although having a certain chronological sequence in sense of ages, they would not be connected in their stories. So, as the minigame idea was also very clear to the team, the requirements of this minigame were described directly as a product backlog.

6.2 The Production

In the end of pre-production, EngReq was a minigame very similar to a quiz, where the inhabitants of a village in the middle ages presents, in they opinion, what a determined system would require. The role of the player is, according the requirements shown, rank them between functional or non-functional requirement, or reject them if is not a requirement at all. A hotel system was adapted to the minigame, and there was the possibility of adding new systems through adding new text files in a specific sub folder of the game. EngReq was developed in C++ with SDL to display multimedia content on screen. To accompany the development of this project used the site urlwww.xp-dev.com, a website with free repository with tools focused on agile development.

The game's development was through a small team without artistic skills. For this reason, the design of scenery and characters has been outsourced, but even so it was applied the model of iterations, where the designer met regularly with the developers and both parties tried to be as clear as possible in presenting their ideas. The end result of EngReq can be seen in Figure 2.

6.3 The Post-Production

With EngReq ready for beta testing, a questionnaire was designed and applied to undergraduate and graduate students, as well as to software engineering professors to evaluate the game. Thus, through their answers we could analyze the effectiveness of the minigame. The questionnaire used a Likert scale with five levels where the answer 'A' corresponds to 'totally agree' and 'E' corresponds to 'totally disagree'. There where 11 closed questions and has also 4 open questions, where they can give their opinion about the minigame, its strengths and weaknesses, and suggestions of improvements. The questionnaire was applied to five undergraduate

students, six graduate and two teachers.

From the answers, it was clear that the audience believed that the game was tailored to its proposal, but the gameplay was a bit tiring. It was suggested to increase the number of requirements available and show a limited amount of them randomly. The answers also showed that the minigame has the correct concepts about software engineering, even not appearing to be about this subject. One point to improve was the feedback given to the player, where the questionnaire participants suggested decrease the response time of feedback, showing it to the player right after his choice instead of a screen at the end of the game.

A postmortem of EngReq was produced with the feedback of the project. As the first application of this methodology, there were several issues that could be solved with more practice. But some of them included the different schedules of classes between the team, what delayed the project and could not be solved easily. Even so, the project could be considered as successful.

7 Conclusion and future work

The environment of game development with increasing costs and high mutability of the requirements, makes essential to know as sooner as possible whether the investment will return. Knowing what brings "fun" to the game and work it well is a pre-requisite to the it's success. But in traditional development, most of this discovery occurs only in late development, when much time and money already has been invested and there are great risks in making changes. An iterative methodology allows to have features ready soon and thus discover and work the "fun" of the game is easier, while an agile process lets you focus on what really matters.

There are few alternatives for agile methodologies focused on the needs of an environment of game development, where the multidisciplinary teams are common and there is a need to find the "fun factor", that is not easily quantified. The Game-Scrum tries to solve specific problems of game development by assimilating suggestions from professionals of the industry and confronting it with researchers of the area. Some of the approaches made by Game-Scrum may be summarized as follows:

- *Artistic content*: allowing to have work undone at the end of an iteration, and assembling a production chain were the first asset must be done in this iteration, and the next ones over the following iterations.
- *Project scope*: making use of techniques like brainstorming, prototyping and the game design document as part of preproduction facilitates the discovery of the "fun" of the game, and iterations enable to evolve this factor. Also, these techniques may reduce the amount of feature creep in the game.
- *Project management*: using the framework of Scrum to develop games makes a strong process as base of Game-Scrum. As in Scrum itself, the constant evaluation of the feedback of the project helps to improve it each time it is applied.
- *Team organization*: suggesting some alternatives to improve organization, under which the team may choose based on its strengths and weakness those who best suit to the team's culture and preferences.

Even proving good practices during the game's development, Game-Scrum is not able solve all of its problems. There may still be some feature creep, as some ideas only arise in later development. Also, in spite of brainstorming be used to aggregate ideas in order to avoid feature creep, it may lead to the opposite problem of cutting features and enlarge scope if not well done.

Although Game-Scrum have been tested with the development of EngReq, it is needed to mature the suggested methodology. The postmortem of the minigame suggested that the approach have potential, and the experience learned may improve the results of the next minigame. Future work on this subject include apply Game-Scrum in projects of the Fellowship of the Game (`http://fog.icmc.usp.br/`), a group of undergrad students at University of São Paulo focused in developing games. With several ongoing projects, the first step will be to coach the team, applying Game-Scrum to its projects and analyze the postmortems. Collecting this feedback will help to refine and improve Game-Scrum, until it becomes able to solve most problems in game development with unexperienced or experienced teams.

Acknowledgements

The authors would like to thank the Brazilian funding agencies (FAPESP, CAPES and CNPq) and the University of São Paulo (Programa Aprender com Cultura e Extensão) for their financial support. Also, to God, for making all things possible.

References

ABRAHAMSSON, P., WARSTA, J., SIPONEN, M. T., AND RONKAINEN, J. 2003. New directions on agile methods: a comparative analysis. In *ICSE '03: Proceedings of the 25th International Conference on Software Engineering*, IEEE Computer Society, Washington, DC, USA, 244–254.

BECK, K., BEEDLE, M., VAN BENNEKUM, A., COCKBURN, A., CUNNINGHAM, W., FOWLER, M., GRENNING, J., HIGHSMITH, J., HUNT, A., JEFFRIES, R., KERN, J., MARICK, B., MARTIN, R. C., MELLOR, S., SCHWABER, K., SUTHERLAND, J., AND THOMAS, D., 2001. Agile manifesto. `http://www.agilemanifesto.org`. Retrieved in October 2009.

BECK, K. 1999. *Extreme Programming Explained: Embrace Change*, first ed. Addison-Wesley Professional.

DEMACHY, T., 2003. Extreme game development: Right on time, every time. `http://www.gamasutra.com/resource_guide/20030714/demachy_pfv.htm`. Retrieved in July 2010.

FLOOD, K., 2003. Game unified process. `http://www.gamedev.net/REFERENCE/ARTICLES/ARTICLE1940.ASP`. Retrieved in July 2010.

GREGORY, D., 2008. Building a mindset for rapid iteratioon part 1: The problem. `http://www.gamasutra.com/view/feature/3645/`. Retrieved in July 2010.

KANODE, C. M., AND HADDAD, H. M. 2009. Software engineering challenges in game development. In *ITNG*, IEEE Computer Society, S. Latifi, Ed., 260–265.

KEITH, C., 2010. Kanban for video game development. `http://www.infoq.com/presentations/kanban-video-game-dev`. Retrieved in July 2010.

MYLLYAHO, M., SALO, O., KÄÄRIÄINEN, J., HYYSALO, J., AND KOSKELA, J., 2004. A review of small and large postmortem analysis methods.

PETRILLO, F., PIMENTA, M., TRINDADE, F., AND DIETRICH, C. 2008. Houston, we have a problem...: a survey of actual problems in computer games development. In *SAC '08: Proceedings of the 2008 ACM symposium on Applied computing*, ACM, 707–711.

SCHUYTEMA, P. 2008. *Design de games: uma abordagem prática*. Cengage Learning.

SCHWABER, K., AND BEEDLE, M. 2001. *Agile Software Development with Scrum*, first ed. Alan R. Apt.

SCHWABER, K., 2009. Scrum guide. `http://www.scrumalliance.org/resources/598`. Retrieved in July 2010.

TELES, V. M., 2006. Extreme programming. `http://improveit.com.br/xp`. Retrieved in October 2010.

WIKIA, 2009. Video game industry. `http://vgsales.wikia.com/wiki/Video_game_industry`. Retrieved in July 2010.