User stories as actives for game development

Victor Schetinger Cesar Souza Lisandra Manzoni Fontoura Cesar Tadeu Pozzer

Laboratório de Computação Aplicada
Universidade Federal de Santa Maria

Abstract

Despite the lack of research in project management and software engineering for game development, recent studies have already proven that agile methodologies are very adequate for this purpose. The focus on individuals, communication and reactive iterations such as described by SCRUM and Extreme Programming are very fitting to game development scenarios, where multi-disciplinary teams are frequent and requirements can be very subjective. To further support these methodologies, this work proposes a new purpose to user stories, an already common practice in agile development. By using user stories in addition to minimum set of key information dubbed the "three Rs": *related*, *responsible* and *result*; this paper describes a flexible form of documentation, that can aggregate value to the project and provide an easy channel of communication between involved parties.

Keywords:: Project management, agile methodologies, game development, user stories

Author's Contact:

{victor,cesar,lisandra,pozzer}@inf.ufsm.br

1 Introduction

Project management and software engineering have been active areas of research in the past few decades, mainly due to the growth of the software industry and related branches. As a relatively new field with its own particularities, deviations from traditional management philosophies were needed in order for the projects to achieve success.

From heavily generic and documented approaches as the PMBOK to applied methodologies, such as the agile SCRUM [Rising and Janoff 2000] and XP [Beck and Andres 2004], there has been an effort to adapt and optimize project management to particular software development scenarios. Generally the need for improvement and research comes from economic interests, as can be seen in web development in recent years.

The game industry, despite its growth in the later years still lacks proper project management. Its proximity to the entertainment industry and multidisciplinarity creates a particular scenario unfitting for most current approaches.

Petrillo [Petrillo et al. 2009] analyses current problems in game development, exposing flaws in different projects and showing that there is a great deal of topics for research. The main problems obtained from his results point to the subjective aspect of scope, result evaluation and the great dynamism. Other issues common to any project such as delays and technological problems still appear, but in a different fashion, adapted to the industry context.

Some works show that agile methodologies are intrinsic to game development, and as such its problems should be solved with agile thinking. This work approaches game project management issues with a communication-focused solution based on user stories, a common resource in extreme programming. A different usage is proposed, however, making user stories a high-level language to organize different aspects of game development and create a simple channel of communication between involved entities.

The remainder of the paper is organized as it follows: the next section discusses related works regarding both general game project management and user stories. Section 3 gives a brief description of agile game development and scenarios where our solution should approach. Section 4 further details user stories and introduces our propositions. An example of implementation is illustrated in section 5, while section 6 presents final thoughts, considerations and future works.

2 Related Work

As previously stated in the introduction, the research in project management for game development is very limited. Most of the existent works have a narrow scope, focusing in particular practices [Machado et al. 2010] in game development or educational applications [Diefenbach 2011], for instance.

The study of agile methodologies in game development has been covered in works such as Game-Scrum [Godoy and Barbosa 2010], and the book "Agile Game Development with Scrum" [Keith 2010], both which contributed to the motivations of this work.

Regarding user stories there are mainly two works related to this one. The first one, "A Feasible User Story Tool for Agile Software Development?" [Rees 2002] presents a general-use tool to manage user stories, focused on implementation issues. The second, "Supporting Program Comprehension in Agile with Links to User Stories" [Ratanotayanon et al. 2009] proposes the indexing of source code related to user stories.

3 Agile game development

Game projects are hard to be generalized, for different factors such as scope, business model and platform make very particular scenarios. If taken for example three different popular game projects: World of Goo [2D Boy 2008], Assassins Creed [Ubisoft 2007] and World of Warcraft [Blizzard Enterteinment 2004] lots of particularities can be easily outlined. The first is a short puzzle game focused on an interesting game mechanic and innovative art style, developed by only two people and costing less than U$ 10,000. The two latter are big titles with millionaire budgets and teams of professionals ranging from art directors to sound engineers. Even so, Assassins Creed has a limited, single-player storyline whereas World of Warcraft is a massively multi-player game, where aspects such as server scalability and new content must be continuously worked into the project.

Even considering a wide range of different game projects, there are still some ever-present constraints that define the generalized context of the industry. The main goal of every game is to deliver fun, meaning its features should be accounted as a whole. In this way, linear development methodologies such as waterfall are inadequate for game projects, since they take too long to integrate features into a prototype capable of play testing to evaluate the fun factor. A more adequate solution is the utilization of iterative and cycle-based approaches, creating a good scenario to apply agile methodologies.

The "agile manifesto" [Highsmith and Fowler 2001] is based upon four main postulates, which change the usual paradigm for project management and software development. These postulates have a strong connection with the context of game development:

- Individuals and interactions over processes and tools, for the teams are mostly interdisciplinary and creativity plays a great role in game development;
- Working software over comprehensive documentation, for the game must be played to be evaluated, and documentation has little significance to the end-user;

- Customer collaboration over contract negotiation, for the subjective character of games makes it hard to define the final outcome. The constant satisfaction of the stakeholders guides the project;
- Responding to change over following a plan, for it is difficult to predict how certain features will interact in the final result.

It is also important to note that we are not talking only about software features and code issues. The art of the game as well as its story and mechanics are also part of the project, sometimes being hard to dissociate all of them.

This close relation between game development and agile thinking has already been used to adapt methodologies as discussed in section 2, each focusing on different aspects and scenarios. Instead of aiming at any particular agile methodology, this paper proposes a practice to supplement any of them using a flexible documentation form, based on user stories, to index useful information.

The usual approach to documentation in project management is avoided in agile methodologies due to the excessive time and work it can consume, sometimes without compensating for it. As previously stated, this is specially true in game development where usually the documentation has little utility to the final user. We suggest, however, that a minimal level of record keeping can bring benefits to the game project management, acting in an analogous way as traditional project actives.

Considering the interpersonal dynamics in game development, firstly there needs to be a clear channel of communication despite the multidisciplinary nature of teams for any kind of documentation to work. For this end we suggest the use of user stories, which will be discussed in further detail in the next section, as a simple language between involved parties to communicate features, activities and general aspects of the project in high-level. Furthermore, each user story should have at least three kinds of information: *result*, *responsible* and *related*, dubbed the "Three R's".

4 User stories

An user story is a brief description of a need, a feature or a desire from the point of view of a specific user role in the software project. It may also contain an explanation of its importance, its benefits to the project and criteria for completion. The basic structure of an user story can be as it follows:

"As a (role) I want (something) so that (benefit)."

In agile methodologies user stories are used mainly to estimate time and organize sets of tasks. Its simplicity and high-level language ensures that any person involved in the project, be it a client, manager or developer can be able to understand them and even create new ones.

4.1 User stories in game development

Keith [Keith 2010] recommends the use of user stories in game development, not only for its application in agile methodologies, but for they acquire a special meaning in the context of games. Using the definition of user story described in the previous section, the following example can be suggested: "As a player, I would like to play using a game pad". The player is the final user of the game and in this user story the support for a particular input device is required. The effort to solve this issue could require the work of several kinds of specialists, such as programmers to implement changes in the game, game designers to come up with good ways to control all aspects of the game and maybe even an engineer to project a new device. Due to the nature of game projects, however, user stories will need additional expression power.

As a general rule, a game tries to simulate a particular scenario to immerse players, regardless of genre. These scenarios may vary from a simple table of chess to complex environments like cities with inhabitants and traffic, as can be seen in GTA [Rockstar Games 1997]. Creating user stories for these scenarios differ from the common usage in software projects where a straight-forward use case is designed based on the necessities of the systems users, as the example in the previous paragraph. Since our focus is not on the utility of the different scenario uses, whilst in its aggregated value in the game as a whole, sometimes is needed to model solutions based on the point of view of entities within the game. Consider the following examples, all expressing plausible issues in a game project:

- "As a player, I would like to change my class at any time during an online game";
- "As a medic, I would like to be able to see the health of my allies, because it would make it easier to help injured players";
- "As an enemy unit, I would like to be able to steal items from players";
- "As an animator, I would like to easily import md2 models to the game engine";
- "As the art director, I would like the concept art to be based on cyberpunk fiction, because dystopian futurism is a rising trend on the market".

The first two examples represent a need from the player, the final user, but while one explicitly declares it, the other uses in-game context to represent a particular feature, better describing the desired scenario the game is trying to achieve. These user stories would likely appear commonin modern FPS games where players can choose different roles in a team, such as artillery, recon, engineers and medics. Examples include Battlefield, Team Fortress and Call of Duty.

In the third story, a non-playing entity wants the possibility for it to steal items from players. How is it possible that this enemy, not an user of the game in any sense can make requests? As stated before, in the context of games user stories extrapolate their usual expressiveness to create means to describe a feature. The author of this story could be an AI-designer, trying to create new behaviors for enemies, while needing the help of someone to implement these changes into the project. If this was the case, then this user story would be firstly communicating a request between different members of the team, probably multi-disciplinary, secondly describing a new aspect of the game, which is the possibility of have your items stolen, and depending on the context of the development team it could become a task on the schedule. In fact, lots of implications may be derived from this simple user story, for it defines a relation between two general entities. Similar user stories in this layer of abstraction can be easily used to brainstorm the game design process.

The two latter examples can clearly be characterized as team communication just as the one in the previous paragraph, yet many differences can be noticed between them. Animators and art directors are actual people in a game development team instead of entities from which point of view a feature is described. But what does this tell us from the last two user stories? In their proper contexts and with different precision both express complete ideas, easily understood by anyone even outside the scope of the team.

Rather than suggest or define particular forms to apply user stories, the previous examples merely illustrate possible applications in game development and how they could provide benefits. It is expected that this section introduced a broader range of usages for user stories, for the notion of generalizing different aspects of game project management through them is crucial to this paper.

4.2 The three Rs

Having established that user stories are powerful tools in game development, in this section we shall propose a minimum set of information that should be linked with them in order to achieve our objectives. Good practices and organization of user stories are intentionally not contemplated by the scope of this work, for there is already literature in the topic and our solution focus on a different aspect, allowing flexibility for project management to keep existing policies or implement regarding such issues.

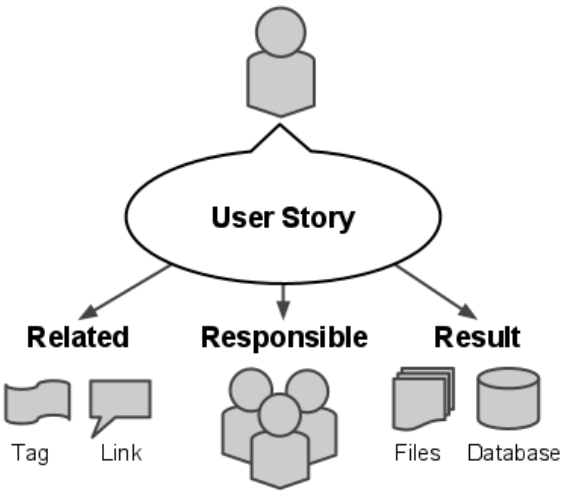

Figure 1: *Abstract framework representing the "Three R's" and their connection to an user story.*

Considering the context of game development discussed until this point, we propose that in an information level, any user story should easily be able to provide who was involved in it, what was made of it and its context. In a simpler form: *responsible*, *result* and *related*, or the "three R's" (Figure 1). Note that these elements are not contained within the user story itself, for this would be a deviation of its purpose. The user story should be seen as a unit of information, a brief description, a topic or an individual in an ontological sense, encapsulating the three R's in a different layer. There are several ways to implement this concept. An intuitive approach could be a relational database, for instance. Details on a suggested implementation will be discussed in section 5, while this section should focus on justifying the purpose.

In a game project scenario where user stories are already being used and registered similarly to what was described in the previous section, what uses could involved parties have for such information? Some common situations are suggested in order to illustrate this point:

- A programmer needs to implement support for a new 3D model format;
- A game designer needs to balance classes for online play;
- The project leader needs to negotiate time to implement a set of features with a stakeholder;
- A new project is being initiated and its concept must be outlined.

All these situations were specifically chosen to relate to user stories from the last section. Suppose they were all real cases from a company that uses user stories, keeps track of them between different projects and development team members have open access to them.

The first situation could be a complicated task, and for a programmer without the proper experience in the technologies used in the project it would be hard even to know where to start. As we assumed this programmer has access to the records of the user stories from section [user stories in game], he could find the particular one where an animator asks for support to the md2 format. How easy it would be for this user story to be found would depend on how the information is organized on this particular database, for we have not provided means to facilitate this search yet. Since md2 is a famous 3d model format and was an actual term in the user story, lets say our programmer was lucky and found it. Now, what possible benefit could this provide to help him in his task? The high level of dynamism, reaction, teamwork and flexibility demonstrated so far to be present in the context of game development suggest this programmer would either want to know how was this user story solved to adapt his own solution or ask for suggestions from the involved parties. In other words, what he wants from this user story are the

Figure 2: *Screen from our Redmine implementation.*

results its responsible.

Once again we are not enforcing a particular way to apply this model yet, keeping an open abstraction from which to relate to a variety of agile methodologies and game development practices. For the sake of understanding, however, a few examples can be provided to better illustrate our purpose. The result linked to a particular user story should exist, but could be a wide set of things, depending on its nature. Screenshots, version builds, code, tasks in a management system, videos or even descriptions can be considered valid results with different levels of detail and satisfaction, but nevertheless useful for this intended purpose.

The interpersonal dynamics of agile methodologies have already proven to be efficient, eliminating hierarchic communication, reducing bureaucracy and focusing in direct conversation. To properly benefit from this aspect, being able to easily find individuals who might be helpful or necessary in each particular situation is crucial. For this purpose, knowing who was *responsible* for each user story is sufficient to establish a channel of communication, even if it is a simple email, phone call or quick conversation. The second R is based on the premise that at least one *responsible* must be identifiable for each user story, but linking to additional individuals might prove useful.

As previously stated, the solution suggested in this work should be used to aggregate value to game development projects and teams, similarly to actual actives, focused on the communication level. This means the *responsible* is the most important element of any user story in our approach, for accessing an individual ultimately results in creating communication and accessing information.

The third R, *related* addresses the issue previously mentioned of how relevant stories are to be found. A simple way to see it is as meta-information, such as tags, aggregated to the user story. In this way, it is even possible to enforce certain practices in tagging such as key genres and naming conventions, improving the organization of this classification. Furthermore, links can be established between different user stories and projects that are considered to be related in a relevant way, broadening the universe of possible solutions.

While the first two examples from this section can be considered similar, where the user stories are being used as source of information and inspiration for development and troubleshooting, the last ones illustrate another interesting usage. Instead of solving open issues and existing tasks, the processes of planning, conceptualizing and negotiating from the forementioned examples can be greatly improved by having information organized relevantly. For instance, project leaders may use our approach to user stories as a portfolio, being able to quickly demonstrate the capabilities of a team and estimate the amount of work for a certain task.

5 Implementation

Although many projects have their user stories written on paper, our approach suggests that they should be digitally registered by using specialized software, making it easier to search and gather all the necessary related information. There are many free or open-source solutions for project management, such as Redmine and

Trac, which can be easily adapted to implement our approach to user stories.

Redmine is a popular open source tool that has many interesting native features for project and task management. For its flexibility and ease of use and configuration, it was chosen to illustrate the propositions of this work in this session.

Redmine allows the registry of projects and issues, organizing them in different types, such as bugs and features. Our implementation suggests a new type of issue called User Story. In this way we create a new classification for the different kinds of issues while still allowing the registration of bugs and tasks from user stories. This also means it is possible to integrate this functionality with an existent usage of Redmine, ensuring compatibility.

Our User Story issue format uses the title field to describe the story itself, complying with our fore-mentioned idea of condensing information in the story text and keeping the description field available to additional details. To improve the searches we create a new field called Tags, so we can add other keywords not present in the actual story, representing the *related* element.

There are many approaches for creating tasks based on user stories, generally guided by practices of a particular methodology. Usually it is recommended to break large user stories, also called "epics", in smaller ones, easier to estimate and to fit in a development iteration. In Redmine we were able to register the original story (epic) and link substories or tasks to it, keeping a full activity log and a *related* relation.

Using a version control system, such as subversion, is heavily encouraged, for it simplifies the storage of source code, images, animations, builds and documents created in the project. Each user story will *result* in the creation and modification of many files, which will be part of the version control reviews. Redmine allows the association of issues with those reviews through comments on the commits.

By default, Redmine has the limitation of linking only the author of an issue to it, which would be the most intuitive approach to implement the *responsible* element. A simple way to get around this limitation is registering additional users that would qualify as *responsible* as watchers in the issue. In this way, they will be able to add information to the user story and will be listed in the issue page.

An example user story page in our Redmine implementation can be seen in figure 2. Notice that all the "three Rs" are present and can be seen in the same page, tough they are not properly organized in an intuitive layout. Even so, since all information is contained in the same page, a simple web search engine, preferably configured to search through all projects registered in Redmine, can be used to easily find and access desired user stories.

In our implementation, if someone wanted to find medieval-themed images that the user "designer senior" participated in the creation, maybe to invite him for a future project, for example, searching for the words "designer senior", "medieval" and ".jpg" would be sufficient. The user story from figure 2 would be one of the top results, since it contains all the search terms.

The implementation presented in this session through the use of Redmine is not a complete solution, and its actual practical use may be subject for research. Its main objective was to present a concrete application supporting the usage of user stories as proposed in this work. By using a well-know software as Redmine and describing simple steps to implement the proposed features, it is expected that the scenarios presented thorough the article can be better visualized in practical way.

6 Conclusion

In this paper it was proposed a new form of documenting based on user stories to support agile methodologies for game development. Through the conceptual framework of the "three Rs" it is provided a minimal structure to encapsulate commonly relevant information in game project management without enforcing compulsory practices, which is specially relevant in the context of agile methodologies.

Even though the actual benefits of our proposals are still to be tested in practice, the described scenarios used in the argumentation were adapted from the literature and are very plausible in the context of game development. Furthermore, the feasibility of implementing the basic structure of user stories and the "three Rs" was demonstrated through the software Redmine.

As currently described in this paper, this work still has need for improvement and further research. Firstly, it would be ideal to test its usefulness in real case scenarios, for instance in game development teams and companies. Based on the obtained results, then, it would be possible to focus on specific aspects such as implementation, synergy with particular methodologies and even corrections.

7 Acknowledgments

This work was supported by CAPES-Brazil through the Project RH-TVD # 133/2008.

References

2D BOY, 2008. World of goo.

BECK, K., AND ANDRES, C. 2004. *Extreme Programming Explained: Embrace Change (2nd Edition)*. Addison-Wesley Professional.

BLIZZARD ENTERTEINMENT, 2004. World of warcraft.

DIEFENBACH, P. J. 2011. Practical game design and development pedagogy. *IEEE Comput. Graph. Appl. 31* (May), 84–88.

GODOY, A., AND BARBOSA, E. F. 2010. Game-scrum: An approach to agile game development. *IX SBGames* (November).

HIGHSMITH, J., AND FOWLER, M. 2001. The agile manifesto.

KEITH, C. 2010. *Agile Game Development with Scrum*, 1st ed. Addison-Wesley Professional.

MACHADO, T. L. D. A., RAMALHO, G. L., ALVES, C. F., GARCIA, V. C., ARAUJO, L. F., VAMBERTO, L., GOMES, V. C. F., SILVA, A. P., AND BARROS, G. X. D. S. 2010. Game development guidelines: Practices to avoid conflicts between software and design. *IX SBGames* (November).

PETRILLO, F., PIMENTA, M., TRINDADE, F., AND DIETRICH, C. 2009. What went wrong? a survey of problems in game development. *Comput. Entertain. 7* (February), 13:1–13:22.

RATANOTAYANON, S., SIM, S. E., AND GALLARDO-VALENCIA, R. 2009. Supporting program comprehension in agile with links to user stories. In *Proceedings of the 2009 Agile Conference*, IEEE Computer Society, Washington, DC, USA, AGILE '09, 26–32.

REES, M. J. 2002. A feasible user story tool for agile software development? In *Proceedings of the Ninth Asia-Pacific Software Engineering Conference*, IEEE Computer Society, Washington, DC, USA, APSEC '02, 22–.

RISING, L., AND JANOFF, N. S. 2000. The scrum software development process for small teams. *IEEE Softw. 17* (July), 26–32.

ROCKSTAR GAMES, 1997. Grand theft auto.

UBISOFT, 2007. Assassin's creed.