Adoção de Metodologias Ágeis para Produção de Jogos Sociais com Times Distribuídos

Jamilson Batista Antunes, João Emanoel Ambrósio Gomes, Lenin Ernesto Abadié Otero, Vinicius Cardoso Garcia, Silvio Romero de Lemos Meira

Centro de Informática
Universidade Federal de Pernambuco (UFPE) - Recife - PE - Brasil

{jba, jeag, leao, vcg, srlm}@cin.ufpe.br

Abstract. *The use of agile methodologies for software development has become a demand in geographically distributed teams. This paper aims to report the experiences obtained in a factory open-source software, created by post-graduate students of Federal University of Pernambuco, in developing social games multiplatafoma. Describing the process of adapting the Scrum methodology in development of social games with distributed teams.*

Resumo. *A utilização de metodologias ágeis para desenvolvimento de software tem se tornado uma demanda em times geograficamente distribuídos. Este trabalho tem por objetivo relatar as experiências obtidas em uma fábrica de software open-source, criada por alunos de pos-graduação da universidade federal de Pernambuco, no desenvolvimento de jogos sociais multiplatafoma. Descrevendo o processo de adaptação da metodologia Scrum no desenvolvimento de jogos sociais com equipes distribuídas.*

1. Introdução

Desde a popularização das redes sociais, novas alternativas são buscadas na produção de jogos para criar "meta-redes", possibilitando a implantação de sistemas de comercialização e divulgação de marcas a partir das aplicações.

Com o crescente mercado de jogos sociais e as constantes exigências de mais recursos para conseguir atrair um maior número de usuários, empresas necessitam expandir a capacidade de produção e descentralização no desenvolvimento dos seus projetos, uma vez que o desenvolvimento centralizado de software tem se tornado cada vez mais oneroso e menos competitivo.

Como consequências desse novo modelo de desenvolvimento de jogos, surgem benefícios como a redução de custos e a facilidade no momento de encontrar mão de obra capacitada. Contudo, como grande parte dos desafios na Engenharia de Software não é limitada apenas a aspectos técnicos [Kontio 2004], o desenvolvimento distribuído de software ainda deixa muitas dúvidas quanto a sua real eficácia, como, por exemplo: times distribuídos têm a mesma eficiência que times centralizados? A comunicação distribuída, e muitas vezes assíncrona, é tão eficiente quanto à comunicação síncrona? Os processos de software atuais são capazes de lidar com as características do desenvolvimento distribuído e ao mesmo tempo garantir a qualidade do produto?

Neste contexto, é possível visualizar os métodos ágeis como solução para auxiliar no desenvolvimento distribuído de jogos, acelerando o rítmo no

desenvolvimento e organização dos times, e consequentemente ganhando produtividade. Como exemplo, tem-se um *framework* baseado em princípios ágeis, denominado *Scrum*, no qual foi projetado para acrescentar foco, comunicação, clareza, e transparência para desenvolvimento de software.

Baseando-se nisso, este trabalho relata as experiências adquiridas no desenvolvimento de um jogo social multiplaforma, adaptando-se a metodologia *Scrum* para o processo de gerenciamento de times distribuídos. Adicionalmente, utilizando ferramentas e técnicas apropriadas para divisão e organização dos times.

Este trabalho está organizado conforme se segue: a seção 2 apresenta uma visão geral de desenvolvimento de jogos sociais *open source*; a seção 3 apresenta o relato de experiência: modelo ágil com times distribuídos para desenvolvimento de jogos sociais; na Seção 4 são descritos os problemas enfrentados e as lições aprendidas; e finalmente, na seção 5 temos as conclusões sobre o trabalho.

2. Visão Geral de Desenvolvimento de Jogos Sociais Open Source

Um crescente movimento em torno do desenvolvimento de software livre é observado ao longo dos anos, proporcionando alguns benefícios, como: realização de *releases* (liberação de uma nova versão do produto) mais frequentes e projetos com menores custos. Paralelo a isto, o desenvolvimento distribuído vem ganhando força no cenário de desenvolvimento *software*, impulsionado pela crescente demanda de mercado, requisitando que sejam satisfeitas restrições de custos, prazos e qualidade.

Segundo [Raymond 1998], *software open source* é desenvolvido por times auto-organizados e distribuídos, que raramente se reúnem presencialmente, coordenando suas atividades através de ferramentas específicas. O mesmo autor, ainda descreve um trabalho relevante sobre o processo de *software open source* quando associa o desenvolvimento nas comunidades de *software* livre, desprovido de qualquer processo formalizado, a um "Bazar" no qual as contribuições ocorrem de forma *ad hoc*. Enquanto que o modelo de desenvolvimento tradicional de software está associado a uma "Catedral" e possui um processo formal bem definido.

Os jogos eletrônicos, por sua vez, estão cada vez mais sofisticados do ponto de vista de definição de imagem, jogabilidade, níveis diferenciados de interatividade, dentre outras características. Os jogos digitais estão presentes em diversas mídias e formatos, desde computadores pessoais a diversos tipos de dispositivos portáteis. Neste contexto, jogos sociais veem surgindo com intuito de proporcionar uma maior interatividade entre os usuários e gerar um elevado faturamento para a indústria de games.

De acordo com [Lovell 2011], jogos sociais são aplicativos de entretenimento que utilizam as plataformas da *web social*, com o objetivo de se propagarem. Estes apresentam como principais características a formação de "meta-redes" (redes de jogadores dentro de uma rede social), compartilhamento de pontuações, geração de desafios e competições entre os "adversários virtuais".

Como trabalhos relacionados podemos citar [Soltis 2008], neste trabalho o autor relata a importância de desenvolver métodos para compreender e projetar jogos digitais, analisando, dessa forma, as interações sociais geradas pelos mesmos. Já em [Kirman 2010], o autor argumenta que a experimentação e diversão são pontos chaves para o

desenvolvimento de jogos sociais, apresentando ainda, um estudo de caso detalhado do jogo social *paidic*. A seguir veremos um pouco sobre o *framework Srum*, a metodologia ágil adaptada e utilizada durante todo o projeto.

Na próxima seção, será apresentada com detalhe a forma de planejamento do *Scrum* durante o desenvolvimento do jogo *Catch the Pigeon*, assim como a adaptação do mesmo em times distribuídos.

3. Relato de Experiência: Modelo Ágil com Times Distribuídos para Desenvolvimento de Jogos Sociais

O estudo foi realizado na fábrica de *software* MOSAIC (*Mobile Social Applications in the Cloud*). A fábrica é formada por 23 (vinte e três) alunos do curso de Pós-Graduação do Centro de Informática da Universidade Federal de Pernambuco, dentre eles mestrandos e doutorandos que fazem parte da estrutura organizacional da fábrica. Essa estrutura é composta por um Comitê Gestor, responsável pelas principais decisões estratégicas da Fábrica, times de produção liderados pelo *Scrum Master*, o *Product Owner* e os *Stakeholders*.

O projeto consiste no desenvolvimento de um jogo social multiplataforma e *open source* denominado *Catch the Pigeon* [Mosaic 2011]. Utiliza serviços da rede social *facebook* e foi desenvolvido para mais de uma plataforma, o que permite a sua execução em ambientes *web* ou móveis que utilizam o Android como sistema operacional. A história do jogo gira em torno de um pombo correio, e o objetivo dos jogadores é proteger o pombo enquanto leva mensagens aos seus amigos da rede social, ou seja, um meio de proporcionar entretenimento na interação dos usuários [Mosaic 2011].

O presente estudo é apresentado seguindo os tópicos: metodologia *Scrum*, práticas da metodologia, organização e gerenciamento das atividades na fábrica, divisão dos times, artefatos gerados e por fim ferramentas utilizadas.

3.1. Metodologia Scrum

Segundo [Keith 2010] em cada *sprint* a metodologia utilizada no desenvolvimento de um jogo recebe contribuições advindas, quais sejam: *Concept*, *Design*, *Coding*, *Asset Creation*, *Debugging*, *Optimizing*, *Tuning and Polishing*.

As atividades em uma *sprint* são organizadas dentro do ciclo de vida em cascata. Esta foi uma alternativa adotada pela fábrica Mosaic com base nos projeto de jogos desenvolvidos pelo C.E.S.A.R (Centro de Estudos e Sistemas Avançados do Recife).

O processo adotado segue as atividades (levantamento de requisitos, análise e projeto, desenvolvimento, testes e operação), bem como a aplicação de práticas de testes de software, como, por exemplo, TDD (*Test Driven Development*). A Fig. 1 apresenta uma visão geral do ciclo de trabalho.

Nos próximos tópicos abordam-se as práticas adotadas do *Scrum* e organização dos times, bem como a adaptação das ferramentas de gerenciamento e configuração para essa metodologia.

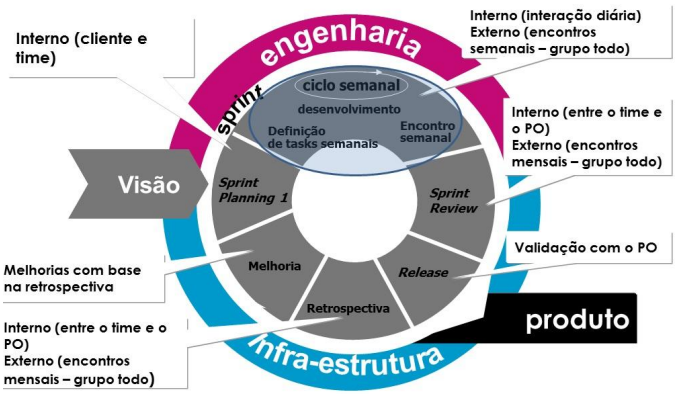

Fig. 1 - Visão geral do ciclo de trabalho da Fábrica MOSAIC

3.2. Práticas

Seguindo a organização proposta pelo *Scrum* para o gerenciamento, o processo da Fábrica MOSAIC é composto das seguintes práticas: *Sprint Planing Meeting*, *Sprint*, *Daily Scrum*, *Sprint Review Meeting*, *Release e Sprint Retrospective*. Em [Mosaic 2011]. são apresentas as discursões e descrições de todas estas práticas.

3.3. Atividades

As atividades presentes nas *Sprints* apresentam características específicas da aplicação, as mesmas estão documentadas e delineadas em [Mosaic 2011]. Cada *sprints* é composta pelas atividades listadas a seguir: *Concept, Design*, *Coding*, *Asset creation*, *Debugging*, *Optimizing e Tuning and polishing*.

Na Fig. 2, são ilustradas as fases durante o ciclo de desenvolvimento de jogos adotado pela fábrica Mosaic. Este modelo apresentado foi adaptado a partir do modelo em cascata (modelo de desenvolvimento tradicional de software). No entanto, a abordagem é distinta quando aplicada a metodologia *Scrum* para desenvolvimento de jogos, pois esse ciclo se repete a cada *Sprint*, já no modelo tradicional executamos uma única fase por vez.

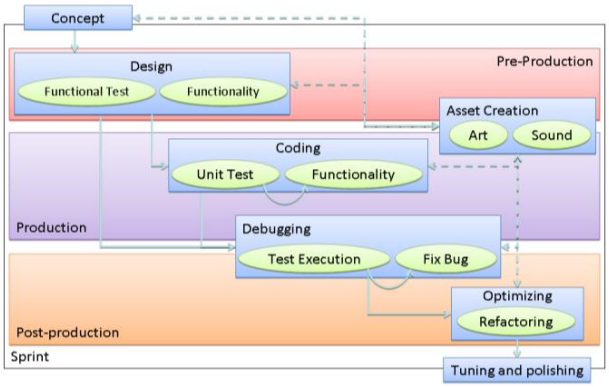

Fig. 2 - Atividades que compõem cada *sprint*.

3.4. Times

Além dos papéis inerentes ao processo *Scrum* (*Product Owner, Scrum Master,* entre outros), serão incorporados papéis específicos ao processo de desenvolvimento de jogos. Estes papéis compõem a equipe de desenvolvimento do grupo de game design, Web Móvel, podendo haver casos de intersecção:

- *Arte Designer:* responsável por desenhar personagens, cenários, instrumentos do jogo, efeitos resultantes das ações do jogador, ícones e menus do jogo;
- *Áudio Designer*: será responsável por toda sonorização do jogo, desde sons emitidos pelos personagens, aos sons das armas e demais efeitos pertencentes ao jogo.
- *Animador:* responsável por criar e manter as animações dos elementos existentes no jogo.

A equipe da fábrica encontra-se organizada em subgrupos com times constituídos de 2 a 8 membros. Os times selecionam o objetivo da *sprint* e especificam os produtos de trabalho em conjunto. Cada time tem direito de fazer o que estiver previsto entre as atividades do projeto para alcançar o objetivo da *sprint.* Ao final, o que foi produzido é apresentado ao *Product Owner* e *stakeholders*.

A Fig. 3 apresenta uma visão geral da organização da fábrica, seus times e papéis.

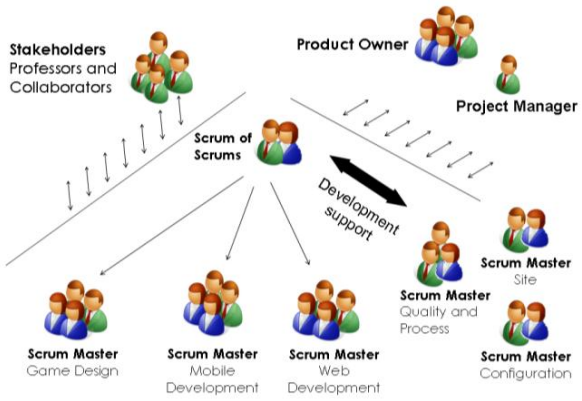

Fig. 3 - Organização da Fábrica MOSAIC.

Ressaltando-se, ainda, que durante o desenvolvimento do projeto, pode-se contar com a participação de colaboradores externos. O *Scrum Master* (líder) do time que o colaborador demostrou o interesse em participar, é responsável por coordena-lo nas atividades.

3.5. Ferramentas

Foram utilizadas diversas ferramentas para auxiliar no gerenciamento do projeto e dos times, desde o controle de versão do código ate ferramentas para dar uma maior visibilidade no andamento do projeto pelos S*takeholders*. Porém algumas destas foram adaptadas para o cenário de desenvolvimento distribuído de jogos utilizando a metodologia *Scrum*, durante a implementação do jogo *Catch The Pigeon*.

Dentre as ferramentas, podem ser citadas como cruciais no processo de adaptação da metodologia para o DDS: *FireScrum*[1]*, Mantis*[2]*, GitHub*[3]*,* TestLink[4]. Estas ferramentas auxiliaram a implantação do *Scrum* no ambiente de desenvolvimento de jogos sociais com times distribuídos, e através destas os problemas de comunicação entre os times e de visibilidade pelos *stakeholders* foram solucionados.

3.6. Artefatos

Além dos artefatos provenientes do *Scrum*, serão incluídos outros que são relacionados à natureza da aplicação que estará sendo desenvolvida. Para que o processo possa dar suporte à internacionalização, todas as atividades realizadas no desenvolvimento do jogo são documentadas em inglês. A seguir são listados os principais artefatos que compõem o processo:

- *Game Design Specification*: *template* para auxiliar na especificação do *game design* da aplicação, com explicações sobre cada funcionalidade do jogo;
- *Product backlog*: artefato típico do *Scrum* indica os requisitos, a prioridade dos requisitos e os grupos de requisitos priorizados em cada *sprint* e release.
- *Sprint backlog* - Artefato típico de *Scrum* estabelece as tarefas que a equipe define para transformar o *Product backlog* selecionado para aquela *sprint* em um incremento de funcionalidade do produto potencialmente disponível (*release*), ou seja, uma lista de tarefas que define o trabalho da equipe para um *Sprint*;
- *Burndown chart*: mostra o trabalho estimado ao longo do projeto em Scrum e a quantidade de trabalho restante ao longo do tempo;
- *Impediment list*: uma lista de qualquer item em torno de um projeto em Scrum que impede sua produtividade e qualidade. Deve ser tratado pelo Scrum Master a fim de garantir que a equipe de desenvolvedores do projeto siga com sua implementação;
- *Project increment*: ao final de cada *Sprint* em Scrum a equipe deve entregar um incremento de qualidade da produção do que deve ser entregue ou do sistema, demonstrado para o *Product Owner*, e a cada revisão do projeto, para revisores externos. É um artefato opcional, em formato de textual;
- *Documento de Arquitetura*: provê uma descrição sobre a estrutura da aplicação, capaz de demonstrar quais soluções foram empregadas no projeto do software;
- *Casos de teste*: inclui as estratégias definidas para teste da aplicação, envolvendo os testes de unidade, testes de aceitação e os resultados de testes.

4. Problemas Enfrentados e Lições Aprendidas

Ao longo do desenvolvimento do projeto foi percebido que nem todas as práticas do *Scrum* eram diretamente aplicadas ao contexto de desenvolvimento distribuído de software. A seguir apresentam-se os maiores desafios para a utilização de *Scrum* no

[1] http://sourceforge.net/projects/firescrum/

[2] http://www.mantisbt.org/docs/master1.2.x/en/administration_guide.pdf

[3] https://github.com/

[4] http://testlink.sourceforge.net/docs/testLink.php

desenvolvimento de jogos sociais *open source* com times distribuídos e como foram solucionados:

- O *Scrum* defende a unidade da equipe de desenvolvimento. Isso está fortemente relacionado com a presença física da equipe e com iterações diárias. Na experiência aqui relatada, a equipe geograficamente distribuída conseguiu superar este desafio com o uso sistemático de fóruns, listas de discussões e ferramentas de comunicação, como *chats*, que permitiram uma boa iteração entre as equipes;
- As reuniões diárias de quinze minutos previstas no *Scrum* para responder às três perguntas básicas foram substituídas por 2 (duas) reuniões semanais de cada time, conforme o processo definido para a fábrica de software;
- O *Scrum* é focado em equipes auto-organizadas e auto-gerenciáveis, além de prever a questão motivacional como principal aspecto de sucesso do projeto. Esses mesmos aspectos puderam ser percebidos no desenvolvimento distribuído, onde a equipe poderia ser formada por qualquer membro da comunidade *open source*, ou seja, colaboradores externos para os projetos, coordenados pelo líder do time.

A preocupação com os requisitos não funcionais (qualidade, usabilidade, jogabilidade, escalabilidade e segurança) do game deverá ser constante por parte de todos os times envolvidos no desenvolvimento. Principalmente, pelo time de processo e game design (time responsável por elaboração dos requisitos não funcionais). Outro ponto de adaptação foi em relação às responsabilidades do *Scrum Master* que, neste contexto, além de ter parte de seu tempo dedicado ao gerenciamento, também fazia parte da equipe de desenvolvedores.

A partir da experiência obtida nesse trabalho, teve-se como contribuições à engenharia de software, as ferramentas e métodos adaptados, experimentados e testados pela fábrica de software Mosaic. Dentre eles a forma como gerenciar equipes distribuídas em nível de conhecimento e geograficamente, durante o processo de desenvolvimento de um jogo social *open source*.

5. Conclusão

Este trabalho apresentou um relato de experiência sobre a adaptação de métodos ágeis, baseado na abordagem proposta pelo *Scrum*, em um processo para desenvolvimento de jogos socais *open source* com times distribuídos.

O projeto teve duração de 5 (cinco) meses (entre fevereiro e julho de 2011), durante este período foram criados 36 (trinta e seis) itens de *backlog* e realizadas 303 (trezentos e três) tarefas, divididas entre os times da fábrica Mosaic.

Apesar do *Scrum* não cobrir todas as características especificadas para equipes distribuídas, foi possível fazer uso de diversos aspectos de desenvolvimento ágil sem comprometer os requisitos exigidos no projeto.

Como trabalhos futuros, temos: analisar o desempenho dessa experiência utilizando outras metodologias de desenvolvimento de software, integrar o jogo a rede social *twitter*, efetuar um estudo sobre os impactos que são gerados entre as equipes distribuídas em caso de falha de algum projeto.

Referências

Audy, J. and Prikladnicki, R. (2007). "Desenvolvimento Distribuído de Software: Desenvolvimento de Software com Equipes Distribuídas". Rio de Janeiro: Elsevier.

Carmel, E. (1999) "Global Software Teams – Collaborating Across Borders and Time Zones", Prentice Hall, EUA.

Cavalcanti, E. O. (2009) "FIRESCRUM: Ferramenta de Apoio à Gestão Ágil de Projetos Utilizando Scrum", Dissertação de Mestrado – C.E.S.A.R – Recife – PE.

Freitas, A. V. (2005) "APSEE Global: a Model of Processes Manegement of Distributed Software Processes", Faculdade de Informática – UFRS – RS – Brazil.

Keith, C. (2010) "Agile Game Development with Scrum", Addison-Wesley Professional; 1º ed, 384 pg.

Kirman, B. (2010) "Emergence and Playfulness in Social Games". 4th International Academic MindTrek Conference: Envisioning Future Media Environments. ACM 978-1-4503-0011-7/10/10.

Komi-Sirvo, S and Tihinen M. (2005). "Lessons Learned by Participants of Distributed Software Development", Journal Knowledge and Process Management, vol. 12 n 2 p. 108 – 122.

Kontio, J., Höglund, M., Rydén, J. and Abrahamsson, P. (2004) "Managing Commitments and Risks: Challenges in Distributed Agile Development", 26th International Conference on Software Engineering, pp. 732/733.

Lovell, N. (2011) "What is a Social Game?", http://www.gamesbrief.com/2011/01/what-is-a-social-game/. 18 jun 2011.

MOSAIC (2011) "Mobile Social Applications in the Cloud", http://www.mosaic.eng.br. 15 jun 2011.

Posea, V., Balint, M., Dimitriu A., and Iosup, (2010) "A. An Analysis of the BBO Fans Online Social Gaming Community".

Prikladnicki, R. (2003) "MuNDDoS Um Modelo de Referência para Desenvolvimento Distribuído de Software". Dissertação de Mestrado, PUC/RS, Porto Alegre, RS, Brasil.

Raymond, E. S. (1998) "The Cathedral and the Bazaar", http://www.firstmonday.org/issues/issue3_3/raymond/. 15 dec. 2011.

Schwaber, K. and Sutherland, J. (2010) "Scrum", http://www.scrum.org/storage/scrumguides/Scrum%20Guide%20-%20PTBR.pdf. 17 mai. 2011.

Soltis, D. (2008) "Moving Parts: The Interdependence of Game Play and Social Dynamics in Digital Games". Sandbox Symposium 2008, Los Angeles, California, ACM 978-1-60558-173-6/08/0008.

Zanoni, R. (2002) "Modelo de Gerência de Projeto Baseado no PMI para ambientes de Desenvolvimento de Software Distribuído". Dissertação de Mestrado, PUC/RS, Porto Alegre, RS, Brasil