Adoção de SCRUM em uma Fábrica de Desenvolvimento Distribuído de Software

Felipe S. Furtado Soares, Leila M. Rodrigues de Sousa Mariz, Yguaratã C. Cavalcanti, Joseane P. Rodrigues, Mário G. Neto, Petrus R. Bastos, Ana Carina M. Almeida, Daniel Thiago V. Pereira, Thierry da Silva Araújo, Rafael S. M. Correia, Jones Albuquerque

Centro de Informática – Universidade Federal de Pernambuco (UFPE)
Caixa Postal 7851, Cidade Universitária – 50.732-970 – Recife – PE – Brasil

{fsfs, lmrsm, ycc, jpr, mgn, prb, acma2, dtvp, tsa, rsmc joa}@cin.ufpe.br

Abstract*. The use of agile software development methodologies has become a demand in distributed software teams. This work has the objective to report the experiences acquired on the adaptation of a distributed development process based on SCRUM executed by an open source software factory.*

Resumo. *O uso de metodologias ágeis de desenvolvimento de software tem se tornado uma demanda em equipes distribuídas de software. Este trabalho tem por objetivo relatar as experiências obtidas na adaptação de um processo de desenvolvimento distribuído com base no SCRUM realizado por uma fábrica de software open source.*

1. Introdução

Ao longo dos últimos anos, organizações estão cada vez mais motivadas a aderir ao modo de desenvolver software de forma distribuída. Boa parte dessa motivação vem do sucesso adquirido por grandes projetos de software open source, como Linux e Open Office, os quais têm como uma das principais características a distribuição geográfica de seus membros.

Outro fator com influência relevante no desenvolvimento distribuído é o avanço das comunicações através da Internet e o crescente surgimento de ferramentas voltadas para Engenharia de Software, as quais visam administrar de forma mais organizada e eficiente o desenvolvimento distribuído de software. Em [Martin e Hoffman 2007] é feito um levantamento de como tais ferramentas (como controladores de versões, ferramentas de triagem de bugs, fóruns de discussão, listas de e-mail etc) têm sido utilizadas por times distribuídos na empresa Kitware para desenvolver software. Alguns sites também oferecem todas essas ferramentas gratuitamente, como é o caso do SourceForge.net [SourceForge 2004].

Como conseqüência direta desse novo modo de desenvolvimento de software, temos a diminuição de custos e uma maior agilidade e praticidade na hora de encontrar mão-de-obra. 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: 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?

Em paralelo a essa discussão estamos vivendo a partir do ano 2000 uma tendência para o desenvolvimento ágil de aplicações devido a um ritmo acelerado de mudanças e inovações na tecnologia da informação, em organizações e no ambiente de negócios [Boehm 2006].

Boehm cita, ainda, que no final dos anos 90 acompanhamos o surgimento de vários métodos ágeis, entre eles: Adaptive Software Development, Crystal, Dynamic Systems Development, eXtreme Programming (XP), Feature Driven Development e Scrum. Todos esses métodos empregam princípios ágeis, tais como ciclos iterativos, entrega rápida de software funcionando e simplicidade, como definido no Manifesto para Desenvolvimento Ágil [Beck et al. 2001] publicado em 2001. A essência desse movimento é a definição de novo enfoque de desenvolvimento de software, calcado na agilidade, na flexibilidade, nas habilidades de comunicação e na capacidade de oferecer novos produtos e serviços de valor ao mercado, em curtos períodos de tempo [HighSmith 2004].

Inserido neste contexto de desenvolvimento distribuído de software, este trabalho apresenta a aplicação do Scrum em um processo de desenvolvimento de uma fábrica de software open source a qual possui fortes características de desenvolvimento distribuído.

Este trabalho está organizado da seguinte maneira: a Seção 2 apresenta uma visão geral de desenvolvimento open source; na Seção 3, é descrita uma visão geral do Scrum; a Seção 4 compreende o estudo de caso com as principais adaptações do Scrum no processo distribuído; na Seção 5 são apresentadas as lições aprendidas e conclusões finais.

2. Visão Geral de Desenvolvimento de Software Open Source

A cada dia é observado um crescente movimento em torno do desenvolvimento de software livre, caracterizando, dessa forma, o desenvolvimento de projetos por equipes geograficamente distribuídas. Além disso, a crescente demanda de mercado por software livre a ser comercializado necessita que sejam satisfeitas restrições de custo, prazo e qualidade [Hecker 2000]. Dessa forma, o processo deve ser definido com a intenção de ser o mais leve possível, mantendo, entretanto, a formalidade necessária para o desenvolvimento distribuído.

Raymond [Raymond 1998] relata que software open source é desenvolvido por times auto-organizados e distribuídos, que raramente se reúnem presencialmente e coordenam suas atividades através de comunicações baseadas em computadores. Ele 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 *ad hoc*. Enquanto que o modelo de desenvolvimento de *software* tradicional está associado a uma "*Catedral*" e possui um processo formal bem definido.

Conforme descrito por González e Robles em [González et al 2003], muitos são os benefícios do modelo de desenvolvimento open source, como a realização dos *releases* mais freqüentes; o baixo custo dos projetos, visto que desenvolvedores e testadores trabalham de forma voluntária; alta qualidade e confiabilidade, visto que são muitas pessoas revisando e testando o mesmo código, em arquiteturas e ambientes distintos, aliado ao fato de que os usuários são tratados como co-desenvolvedores e, em muitos casos, ao deparar-se com o erro, já identifica a solução e envia o pacote com o problema solucionado.

3. Visão Geral do SCRUM

O *Scrum* foi criado em 1996 por Ken Schwaber e Jeff Sutherland e destaca-se dos demais métodos ágeis pela maior ênfase dada ao gerenciamento do projeto. Reúne atividades de monitoramento e *feedback*, em geral, reuniões rápidas e diárias com toda a equipe, visando à identificação e correção de quaisquer deficiências e/ou impedimentos no processo de desenvolvimento [Schwaber 2004].

O método baseia-se ainda em princípios como: equipes pequenas de, no máximo, sete pessoas; requisitos que são pouco estáveis ou desconhecidos; e iterações curtas. Divide o desenvolvimento em intervalos de tempos de no máximo, trinta dias, também chamados de *Sprints*.

O *Scrum* implementa um esqueleto iterativo e incremental através de três papéis principais [Schwaber 2004]: *Product Owner*: representa os interesses de todos no projeto; Time: desenvolve as funcionalidades do produto; *ScrumMaster*: garante que todos sigam as regras e práticas do *Scrum*, além de ser o responsável por remover os impedimentos do projeto.

Um projeto no *Scrum* se inicia com uma visão do produto que será desenvolvido [Schwaber 2004]. A visão contém a lista das características do produto estabelecidas pelo cliente, além de algumas premissas e restrições. Em seguida, o *Product Backlog* é criado contendo a lista de todos os requisitos conhecidos. O *Product Backlog* é então priorizado e dividido em *releases*. O fluxo de desenvolvimento detalhado do *Scrum* é mostrado na Figura 1.

Todo o trabalho no *Scrum* é realizado em iterações chamadas de *Sprints*. Schwaber [Schwaber 2004] explica que cada *Sprint* inicia-se com uma reunião de planejamento (*Sprint Planning Meeting*), na qual o *Product Owner* e o Time decidem em conjunto o que deverá ser implementado (*Selected Product Backlog*). A reunião é dividida em duas partes. Na primeira parte (*Sprint Planning* 1) o *Product Owner* apresenta os requisitos de maior valor e prioriza aqueles que devem ser implementados. O Time então define, colaborativamente, o que poderá entrar no desenvolvimento da próxima *Sprint*, considerando sua capacidade de produção. Na segunda parte (*Sprint Planning* 2), o time planeja seu trabalho, definindo o *Sprint Backlog*, que são as tarefas necessárias para implementar as funcionalidades selecionadas no *Product Backlog*. Nas primeiras *Sprints*, é realizada a maioria dos trabalhos de arquitetura e de infra-estrutura. A lista de tarefas pode ser modificada pelo Time ao longo da *Sprint,* e as tarefas podem variar entre quatro e dezesseis horas para a sua conclusão.

Durante a execução das *Sprints*, diariamente o time faz uma reunião de quinze minutos para acompanhar o progresso do trabalho e agendar outras reuniões necessárias.

Na reunião diária (*Daily Scrum Meeting*), cada membro do time responde a três perguntas básicas: O que eu fiz no projeto desde a última reunião? O que irei fazer até a próxima reunião? Quais são os impedimentos?

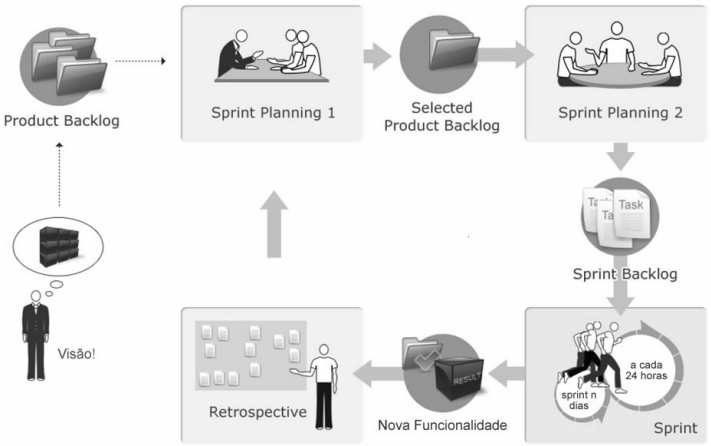

Figura 1. Visão geral do processo do Scrum (adaptada de [Gloger 2007])

No final da *Sprint* é realizada a reunião de revisão (*Sprint Review Meeting*) para que o Time apresente o resultado alcançado na iteração ao *Product Owner*. Neste momento, as funcionalidades são inspecionadas e adaptações do projeto podem ser realizadas. Em seguida, o *ScrumMaster* conduz a reunião de retrospectiva (*Sprint Retrospective Meeting*), com o objetivo de melhorar o processo/time e/ou produto para a próxima *Sprint*.

4. Estudo de Caso - Uso do Scrum no Processo Distribuído

O estudo de caso aqui apresentado foi realizado por uma fábrica de desenvolvimento de software open source [O3S 2007]. Esta fábrica é formada por dez alunos do curso de Pós-Graduação do Centro de Informática da Universidade Federal de Pernambuco. Todos os integrantes são do curso de mestrado e fazem parte da estrutura organizacional da fábrica, conforme ilustra a Figura 2. Essa estrutura é composta por um Comitê Gestor, responsável pelas principais decisões estratégicas da Fábrica e cinco Unidades de Produção, cada uma com atribuições bem definidas e complementares, trabalhando juntas com os clientes e a comunidade externa.

O projeto executado está inserido na área de saúde coletiva/pública, denominado ANKOS (A New Kind Of Simulator) [ANKOS 2007], que surge como um sistema capaz de organizar as informações coletadas por pesquisadores em áreas de estudo da esquistossomose, bem como as imagens de satélite da região, em um banco de dados gerenciável e com possibilidade de consultas, recuperação e indexação da informação, formando uma base sólida para a aplicação de autômatos celulares que possibilitem a geração de cenários capazes de levar a tomada de ações estratégicas de combate e prevenção da doença.

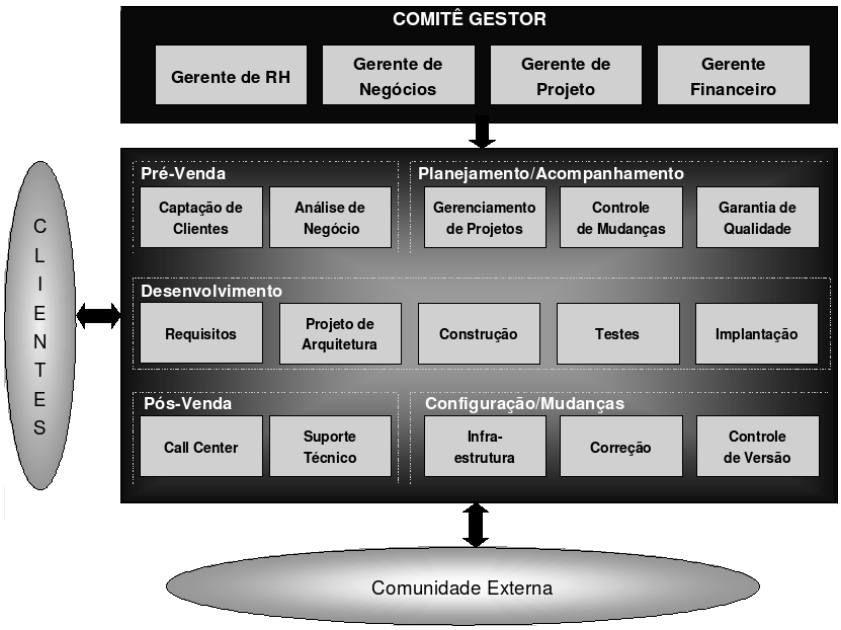

Figura 2 – Estrutura Organizacional da Fábrica O3S

O processo da fábrica O3S é um *tailor* do Hukarz Process [Moraes 2007], focado nas melhores práticas de engenharia de software, no RUP [RUP 2003] e no Manifesto Ágil [Beck, K. et al. (2001)]. É um processo evolucionário orientado a manutenção, baseado em esforço colaborativo e em gerência comunitária. Além de ser executado de forma distribuída, assíncrona e descontínua, caracterizando, assim um processo social, diferentemente dos processos tradicionais.

O processo foi definido baseado em alguns princípios fundamentais: ciclo de vida iterativo e incremental; planejamento nas fases iniciais do projeto; menor quantidade de documentação; certo caos, porém controlável durante o desenvolvimento; adaptação à comunicação remota; desenvolvimento centrado na arquitetura e adaptável de acordo com a necessidade do projeto.

O processo foi descrito baseado em políticas [Johnson K 2001] que definem as diretrizes básicas a serem adotados por todos os projetos da fábrica de software. Elas estão voltadas para o desenvolvimento de software com características open source, ou seja, código compartilhado, equipe distribuída, desenvolvimento colaborativo e descentralizado:

- O projeto de software deve ser modular para facilitar o desenvolvimento concorrente;
- A prototipação deve ser fechada: um pequeno grupo desenvolve a versão inicial do produto antes de liberar para a comunidade externa;
- A melhoria do produto é iterativa e incremental;
- O desenvolvimento é concorrente em vários níveis. Várias atividades de fases diferentes podem ser realizadas em paralelo;
- A revisão de código deve ser realizada em larga escala. Vários usuários revisam e inspecionam o código de outros usuários;
- Os requisitos também podem ser oriundos da comunidade. Os usuários e desenvolvedores definem, coletivamente, as funcionalidades do software;

- Ferramentas de controle de versão e controle de mudanças são necessárias para a boa comunicação entre a equipe. Todo projeto deve ter um repositório, sob gerência de configuração, para armazenar a documentação e o código fonte;
- A forma de comunicação principal é assíncrona;
- A informação deve ser compartilhada: código fonte, listas de discussões, reportagem de erros, solicitação de novos requisitos etc;
- A colaboração é descentralizada;
- A liderança deve ser compartilhada;
- A motivação para contribuição deve ser incentivada pelo desejo de aprendizado, pela criação de uma comunidade, pela disseminação de tecnologia e inovação.

Diversos aspectos do Scrum puderam ser utilizados para o desenvolvimento distribuído. Primeiramente, o comitê da fábrica de software realizou o estudo de viabilidade baseada na visão apresentada pelo cliente (*product owner*). Em seguida, uma proposta comercial foi descrita com todo o *product backlog* inicialmente negociado com a lista dos requisitos da aplicação. Este *product backlog* foi priorizado e dividido nas *sprints* do projeto. Cada *sprint* foi dividida em quinze dias, onde, ao final de cada uma, um conjunto de novo produto era disponibilizado.

Dentro de cada *sprint*, eram realizadas reuniões assíncronas com o *product owner* para que este indicasse os itens do backlog com maior valor de negócio. Em seguida, a equipe analisava colaborativamente através de fóruns e lista de discussões, qual a complexidade desses itens e o que caberia dentro da *sprint* em função de sua capacidade de produção. A técnica de estimativa de Pontos de Casos de Uso foi utilizada sempre considerando uma produtividade média das últimas *sprints* realizadas.

Em seguida, os requisitos eram detalhados e, então, a partir daí o time selecionava os itens de backlog priorizados, dividindo-os em tarefas de até 1 semana por participante. Essas tarefas eram cadastradas numa ferramenta de planejamento e acompanhamento de projetos para projetos que utilizam métodos ágeis [XPlanner 2002].

Durante os quinze dias da *sprint*, a equipe fazia uso do fórum e lista de discussões para simular o *Daily Scrum Meeting*. De tal forma que, diariamente, todos do time sabiam o andamento das atividades e os impedimentos encontrados até o momento. Era responsabilidade de um integrante do comitê da fábrica exercer o papel do *Scrum Master* no sentido de resolver todos os impedimentos reportados por qualquer membro do time.

Assim como a comunicação da equipe, durante a *sprint* a comunicação entre a equipe e o *product owner* era realizada através do fórum, e-mail ou ferramentas de comunicação assíncrona, seja para aprovação de documentos, seja para o esclarecimento de dúvidas ou alinhamento do andamento da *sprint*.

No final da *sprint*, uma reunião síncrona, através de Skype ou MSN, era realizada com o objetivo de apresentar ao *product owner* os resultados obtidos até o presente momento. Em seguida, todos os integrantes do time eram convidados a participar da retrospectiva da *sprint*, que na maioria das vezes era uma reunião realizada de forma presencial. Uma reunião de lições aprendidas era conduzida por um integrante

do time que ficava responsável por coletar os pontos fortes, fracos e sugestões de melhoria de cada participante.

Algumas métricas foram coletadas e analisadas ao longo de cada *sprint* com o objetivo de otimizar o uso do processo adaptado. Duas delas referem-se a variação de esforço de desenvolvimento e a produtividade da equipe. Ambas têm o objetivo de verificar a eficácia do processo de estimativas de esforço adotado. A variação média do esforço realizado pela equipe por *sprint* tem sido em torno de 15% do valor estimado. Enquanto que a produtividade teve uma melhora significativa: iniciando o projeto com uma estimativa de 18 homem-hora para cada ponto de caso de uso e estabilizando em 10 homem-hora para cada ponto de caso de uso produzido.

5. Conclusões e Lições Aprendidas

Este trabalho apresentou um estudo de caso sobre a aplicação de métodos ágeis, particularmente baseado na abordagem proposta pelo Scrum, em um processo para desenvolvimento de software open source com fortes características de desenvolvimento distribuído.

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. Os seguintes aspectos apresentaram os maiores desafios para o uso das práticas ágeis:

- O Scrum defende a unidade da equipe de desenvolvimento. Isso está fortemente relacionado com a presença física do time e com as iterações diárias. Na experiência aqui relatada, o time geograficamente distribuído conseguiu superar este desafio com o uso sistemático do fórum, listas de discussões e ferramentas de chat que permitiam uma boa iteração da equipe;
- As reuniões diárias de quinze minutos previstas no Scrum para que todos respondam às três perguntas básicas foram substituídas pela comunicação assíncrona semelhante ao item anterior;
- 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 que por vontade própria quisesse fazer parte do projeto. Cada integrante é quem escolhia qual atividade do backlog iria desenvolver;
- No Scrum, o *product owner* deve participar ativamente de vários pontos do processo: visão, *sprint planning*, *release* etc. No desenvolvimento distribuído, nem sempre é possível ter essa participação sistemática devido ao alto índice de comunicação assíncrona. Em alguns casos, foi necessário que o time tivesse uma postura pró-ativa para tomar algumas decisões sem envolver o *product owner*;
- 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 dos impedimentos reportados pelo time, também fazia parte do time de desenvolvedores.

Apesar do Scrum não cobrir todas as características específicas para equipes geograficamente distribuídas, foi possível fazermos uso de diversos aspectos de desenvolvimento ágil sem, no entanto, comprometer as particularidades exigidas por esses tipos de projetos.

Referências

ANKOS (2007) "A New Kind Of Simulator", http://sourceforge.net/projects/ankos/, 2007.

Boehm, B. (2006), "A View of 20th and 21st Century Software Engineering", ICSE 2006.

Beck, K. et al. (2001), Manifesto for Agile Software Development, http://www.agilemanifesto.org/, Dezembro 2006.

Gloger, B. (2007), "The Zen of Scrum", http://www.glogerconsulting.de.

Gonzáles, J. e Robles, G.(2003) "Free Software Engineering : A Field to Explore", Upgrade - Software Engineering State of Art, Novática, volume IV, N. 4.

Hecker, F. (2000) "Setting Up Shop: The Business of Open-Source Software", IEEE Software.

Highsmith, J. (2004) "Agile Project Management – Creating Innovative Products", AddisonWesley.

Johnson, K. A. (2001) "Descriptive Process Model for Open-Source Software Development", Master Thesis, Univ. Calgary, Alberta.

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

Martin, K. and Hoffman, B. (2007). "An open source approach to developing software in a small organization". IEEE Software, 24(1):46–53.

Moraes, A. (2007). "Open Source Development Process-like in the Enterprise", Dissertação de Mestrado, Universidade Federal de Pernambuco.

O3S (2007) "O3S Open Source Software Solutions", http://www.yguarata.org/o3s/, 2007.

Raymond, E. S. (1998) "The Cathedral and the Bazaar", Disponível em: http://www.firstmonday.org/issues/issue3_3/raymond/, Acessado em Maio/2007.

RUP (2003) "Rational Unified Process," R. S. Corporation, Ed., 2003.06.01 ed, 2003.

Scacchi, W. (2001) "Understanding the Requirements for Developing Open Source Software Systems". In IEE Proceedings Software, volume 148, number 1, pp. 24-39.

Schwaber, K. (2004), Agile Project Management With Scrum, Microsoft.

SourceForge (2004) "SourceForge.net", https://sourceforge.net/, Acessado em Julho/2007.

XPlanner (2002) "Project planning and tracking tool for agile development teams", http://www.xplanner.org/, Acessado em Julho/2007.