Revista de Sistemas de Informação da FSMA
n. 9 (2012) pp. 16-23

SISTEMAS DE INFORMAÇÃO
http://www.fsma.edu.br/si/sistemas.html

TRILHA PRINCIPAL

Implantação e Adaptação do Scrum em um Laboratório de Pesquisa e Desenvolvimento de Projetos de Software

Igor Ribeiro Lima, Tiago de Castro Freire, Heitor Augustus Xavier Costa

***Abstract* — Agile software development has gained importance in the industry because of its approach on the issues of human agility and return on investment. This paper shows how Scrum agile software project management methodology has been deployed and adapted to the model of software project management of a research and development laboratory. As a result of this deployment, experiences and lessons learned in seven real projects developed by the authors are reported.**

***Resumo* — As formas ágeis de desenvolvimento de software têm ganhado destaque no mercado por causa da forma como abordam as questões humanas e a agilidade no retorno sobre o investimento. Este artigo apresenta como a metodologia ágil Scrum foi implantada e adaptada ao modelo de gerenciamento de projetos de software de um Laboratório de Pesquisa e Desenvolvimento. Como resultado dessa implantação, são relatadas experiências e lições aprendidas em sete projetos reais desenvolvidos.**

***Palavras-Chave* — Scrum, Gerência de Projetos**

I. INTRODUCÃO

O interesse das organizações desenvolvedoras de software tem despertado cada vez mais para a utilização de metodologias ágeis, cujo foco está na colaboração do cliente, no valor dos indivíduos e na adaptação às mudanças. Este interesse é fomentado pelo fato destas metodologias terem mostrado ganhos de produtividade nos mais diversos tipos de projetos de desenvolvimento de software. A escolha da metodologia mais adequada para o desenvolvimento de software não é uma tarefa trivial nem garante o sucesso de um projeto. Por outro lado, as metodologias ágeis têm despertado o interesse do mercado, apresentando evidências de melhoria na produtividade [4].

O movimento original de melhoria no setor foi o que introduziu a noção de metodologia, ou seja, uma abordagem disciplinada para o desenvolvimento de software com o objetivo de tornar o processo mais previsível e eficiente [3]. A definição de metodologia ágil foi elaborada em fevereiro de 2001 em uma reunião de metodologistas de processo de software e resultou no Manifesto para Desenvolvimento Ágil de Software (*Agile Software Development*) [8].

Esse manifesto é uma declaração simples e concisa que busca mudar a visão tradicional de desenvolvimento de software. Além disso, ele acentua o valor [9]: i) dos indivíduos e interações serem mais importantes que processos e ferramentas; ii) do software funcionando ser mais importante que documentação detalhada; iii) da colaboração dos clientes ser mais importante que negociação de contratos; e iv) da adaptação às mudanças ser mais importante que seguir um plano. O Manifesto Ágil é baseado em 12 princípios [8]:

1. Priorizar a satisfação do cliente através de entregas contínuas e frequentes;
2. Receber bem as mudanças de requisitos, mesmo em uma fase avançada do projeto;
3. Realizar entregas com frequência, sempre na menor escala de tempo;
4. Ter sinergia entre a equipe de negócio e a equipe de desenvolvimento de modo a trabalharem juntas diariamente;
5. Manter uma equipe motivada fornecendo ambiente, apoio e confiança necessários;
6. Circular a informação de maneira eficiente por meio de uma conversa face a face;
7. Ter o sistema funcionando é a melhor medida de progresso;
8. Promover desenvolvimento sustentável por meio de processos ágeis;

Igor Ribeiro Lima é aluno de mestrado no curso de Engenharia de Sistemas (PPGESIS) no Departamento de Engenharia (DEG) da Universidade Federal de Lavras (UFLA) e Gerente de Projetos no Laboratório de Pesquisa e Desenvolvimento onde foi realizado o estudo. E-mail: igorlima@comp.ufla.br.

Tiago de Castro Freire é Gerente de Projetos no Laboratório de Pesquisa e Desenvolvimento onde foi realizado o estudo. E-mail: kuruma@bsi.ufla.br.

Heitor Augustus Xavier Costa é Professor no Departamento de Ciência da Computação (DCC) da Universidade Federal de Lavras (UFLA). E-mail: heitor@dcc.ufla.br.

9. Ter atenção contínua a excelência técnica e a um bom projeto aumentam a agilidade;
10. Ter simplicidade;
11. Auto-organizar equipes por meio da utilização de melhores arquiteturas, requisitos e projetos;
12. Refletir, em intervalos regulares, sobre como se tornar mais efetivo, assim, ajustam-se e otimizam seu comportamento.

Na literatura, podem ser encontradas diversas iniciativas para o desenvolvimento de software que primam e incorporam esses princípios ágeis, tais como, *Extreme Programming* (XP) [19, 20, 22, 23, 24, 25], Scrum [1, 20, 25, 26, 27, 28], Crystal [1, 13, 14], *Feature Driven Development* (FDD) [1, 20, 21, 22], *Dynamic Systems Development Method* (DSDM) [15, 17, 19] e *Adaptative Software Development* (ASD) [15, 16, 17, 18].

Neste artigo, o objetivo é apresentar a experiência e a forma de pensar do time de desenvolvimento de um Laboratório de Pesquisa e Desenvolvimento com o uso da metodologia ágil Scrum para a gerência de projetos de desenvolvimento de software com constante alterações/intervenções por parte dos clientes envolvidos nesses projetos. Por exemplo, na experiência a ser relatada neste artigo, propostas de arquitetura de software e formas/maneiras de documentação requisitos surgem, pois os membros do time podem ser designados para outras tarefas (dinamicidade) o que proporciona uma visão ampla do software. Além disso, reuniões diárias permitem aos membros do time encontrarem soluções diretas, pois há envolvimento de todos, o que os tornam mais efetivos na realização de seu trabalho.

A metodologia ágil Scrum foi escolhida para implantação no Laboratório em detrimento às demais mencionadas anteriormente por ser a mais comumente utilizada no desenvolvimento de software pelas empresas que adotam os princípios ágeis [10, 11, 12]. Essa escolha foi por causa da adaptabilidade oferecida pela metodologia e por responder rapidamente às constantes mudanças nos projetos de software.

Este trabalho está organizado da seguinte forma. Conceitos e fundamentos do Scrum são abordados sucintamente na Seção 2. A consolidação da cultura ágil no Laboratório de Pesquisa e Desenvolvimento e a busca de um processo único, adaptado e adequado são apresentados na Seção 3. Lições aprendidas com a utilização da metodologia ágil Scrum são descritas na Seção 4. Considerações finais são discutidas na Seção 5.

I. SCRUM

O Scrum foi desenvolvido por Jeff Sutherland em 1993 [28] e seu objetivo é ser uma metodologia de desenvolvimento e de gerenciamento que segue os princípios da metodologia ágil. A equipe do Scrum é composta por [5]:

- Time. Consiste na equipe de desenvolvimento do projeto constituída por até dez pessoas em que cada membro tem uma habilidade em determinada área, mas os membros não são impedidos de executar tarefas em outra área. Assim, além de estarem integrados, os membros do time conhecem o software, o que diminui o impacto da saída de algum membro;
- *Product owner*. Responsável por especificar a funcionalidade do software e tirar as dúvidas que surgirem durante o desenvolvimento. Ele é o representante do cliente para acompanhar o andamento do projeto de perto e auxiliar na construção de um software que atenda melhor as necessidades do cliente;
- *Scrum master*. Responsável por conduzir o time e eliminar impedimentos que surgem no decorrer do processo. Impedimento é algo que pode atrapalhar um membro do time na realização de seu trabalho. Por exemplo, solicitações referentes a outras atividades que não digam respeito ao projeto, problemas no servidor de teste, dificuldades com a tecnologia e requisitos não planejados que podem atrapalhar a execução da *sprint.*

No contexto do laboratório, o time é constituído de 4 a 7 pessoas, esta quantidade tende a ser pequena essencialmente para manter uma melhor comunicação. O *product owner* (PO) é um membro do Laboratório que fica constantemente no cliente. Em decorrência da dificuldade de ter o cliente no laboratório, houve a necessidade de ter um membro do Laboratório (PO) no cliente. O *Scrum Master* de cada projeto é eleito pelo coordenador de Tecnologia da Informação do laboratório.

O Scrum baseia-se em práticas representadas por (i) reuniões diárias, (ii) reunião de planejamento da *sprint*, (iii) reunião de revisão da *sprint*, (iv) ordenação do *backlog* e (v) apresentação de *release* [2]. As reuniões diárias são realizadas com os membros do time em pé de frente ao *kanban*. O *kanban* é a utilização de cartões (*post-it*) para indicar o andamento de uma determinada tarefa, por exemplo, “para executar” (*To Do*), “em andamento” (*Doing*) ou “finalizado” (*Done*) (Figura 1).

Figura 1 - *Kanban* [6]

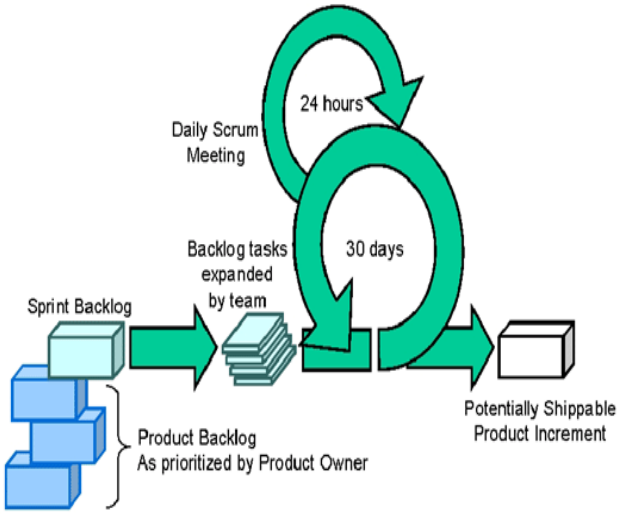

Figura 2 - *Sprint* [7]

As reuniões têm duração de aproximadamente 15 minutos, quando são discutidas questões como o que cada membro do time fez, o que cada um pretende fazer e quais foram os impedimentos encontrados no desenvolvimento daquele dia para que o *Scrum Master* fique ciente destes impedimentos e possa eliminá-los [2].

Uma *sprint* corresponde a um ciclo de desenvolvimento do projeto de software e deve ter duração de duas semanas a um mês (Figura 2). A finalidade da reunião de planejamento da *sprint* é apresentar os itens do *backlog* e estimar as tarefas pelos membros do time. Um *backlog* é a lista dos requisitos do software, ordenado de acordo com a prioridade de cada um, sendo que os itens com prioridade mais alta são listados primeiro e os de prioridade mais baixa ficam por último. Assim, os requisitos são desenvolvidos de acordo com a sua prioridade.

Um dos métodos utilizados para estimar as tarefas é o *planning poker* (Figura 3), cujo objetivo é cada membro do time escolher uma carta com a estimativa da tarefa. Os membros que escolheram a menor e maior pontuação discutem os motivos que os levaram a escolher aquela estimativa. Em seguida, há novas rodadas até que os membros do time entrem em consenso [2].

Figura 3 - *Planning Poker* [29]

A revisão da *sprint* é realizada em uma reunião na qual uma retrospectiva da *sprint* é feita e os pontos positivos e os pontos negativos identificados no decorrer do desenvolvimento são discutidos. Assim, torna-se possível manter os pontos fortes e tenta-se traçar estratégias para melhorar os pontos fracos. Esta é uma maneira para ter *feedback* do desenvolvimento do projeto e aperfeiçoar e evoluir os membros do time [2].

A ordenação do *backlog* é feita de acordo com a prioridade de cada item, sendo calculada a partir da importância de determinada funcionalidade para o cliente. Dessa forma, os itens com maior prioridade são implementados antes dos que têm prioridade menor, o que pode ocasionar maior satisfação ao cliente [2]. Um release é uma versão funcional do software que pode ser entregue ao cliente para homologação. Para cada *release*, uma apresentação da parte funcional é feita ao cliente. Dessa forma, o cliente pode acompanhar o andamento do projeto e homologar o sistema por partes.

II. ADAPTAÇÃO DO SCRUM

A metodologia ágil Scrum não é um processo ou uma técnica para o desenvolvimento de produtos, mas um *framework* iterativo e incremental [28]. Esse *framework* pode ser empregado com diversos processos e técnicas, funcionando bem em um ambiente de constante mudança [5].

O Scrum possibilita revelar o que se pode corrigir com o time e sua essência está fortemente ligada à personalidade dos indivíduos. Dessa forma, é preciso validar constantemente as decisões, as práticas e os processos, segundo os princípios e os valores nos quais o time acredita [4]. O Scrum foi adotado após alguns membros do Laboratório vivenciar relatos de experiências em eventos de caráter científicos que abordaram o tema desenvolvimento e gerência ágeis de projetos de software.

Com isso, sua escolha foi debatida de maneira informal dentro do Laboratório e concluiu-se que a metodologia ágil Scrum poderia ser adaptada e atenderia melhor as constantes alterações provenientes da parte do cliente (*Princípio Ágil 1*). Com isso, haveria entregas frequente com mais valor para o cliente e focaria em maximizar o retorno do seu investimento (*Princípio Ágil 3*). Além disso, evitaria o desperdício e priorizaria a comunicação e a visibilidade do andamento dos projetos, de forma que os membros do time saberiam o que deveria ser feito e o que estaria sendo feito em cada momento (*Princípio Ágil 4*). Contudo, houve a necessidade de adaptá-lo para o cenário em que foi adotado (Laboratório de Pesquisa e Desenvolvimento), objetivando adequação e melhor retorno aos clientes.

Antes da efetiva implantação da metodologia ágil Scrum, não existiam processos de gerência e de desenvolvimento de software bem definidos/estabelecidos no Laboratório. O acompanhamento dos projetos não era feito diariamente, apenas existiam prazos de entrega entre os times e, quando este prazo estava para ser expirado, o responsável vinha à tona e cobrava os resultados. Caso houvesse a possibilidade de

atraso das atividades, os membros do time tinham que cumprir horas extras para cobrir o prazo de entrega. Em poucas palavras, o Laboratório não tinha uma experiência sólida em processo de desenvolvimento de software. Talvez, uma das razões fosse o fato de ser um recente Laboratório no desenvolvimento de software.

Ao longo da adequação da metodologia ágil Scrum, houve atenção para identificar quais mudanças na cultura organizacional eram necessárias para adotar princípios e conceitos de agilidade. Para isso, foram considerados os quatro valores salientados no manifesto ágil mencionados anteriormente. Para que essa cultura ágil fosse consolidada, foi necessária a busca de um processo único, adaptado e adequado à realidade do time.

A. Múltiplos Projetos

Por questão de sigilo, o local do Laboratório de Pesquisa e Desenvolvimento, o nome dos projetos gerenciados e desenvolvidos e os clientes desses projetos são omitidos. Os projetos foram desenvolvidos em Laboratório de Pesquisa e Desenvolvimento de projetos de software localizado dentro de uma Instituição de Ensino Superior do estado de Minas Gerais. Nesse Laboratório, os times estão organizados de acordo com as capacidades técnicas e a disponibilidade de seus membros.

Esses projetos são sistemas de software para auxiliar a gestão ambiental de uma organização governamental do estado de Minas Gerais.

B. Caracterização dos Times e Tecnologias Utilizadas

O time de projetos, em geral, é composto por três a cinco programadores, um a dois administradores de banco de dados, um analista de negócio, um a dois profissionais responsáveis pelo teste e um a três responsáveis em documentar o sistema. A equipe de programadores de cada projeto é composta por desenvolvedores experientes e estagiários, sendo que em cada um destes projetos possui pelo menos dois profissionais com três anos de experiências. A equipe de banco de dados e de analista de negócio possui experiência de, pelo menos, dois anos. A equipe de teste e a equipe de documentação são compostas pela combinação de profissionais experientes e estagiários.

A composição de cada time depende da complexidade do projeto. Um dos projetos com complexidade alta consistiu em realizar o geoprocessamento de uma área específica do estado de Minas Gerais. Nesse projeto, o time foi formado por cinco programadores, dois administradores de banco de dados e cinco pessoas da qualidade de software.

Outro exemplo de projeto, não tão complexo quanto o primeiro, consistiu basicamente no preenchimento de cadastros simples para emissão de formulários. Nesse projeto, foram utilizados três programadores, um administrador de banco de dados e duas pessoas da qualidade de software.

Um terceiro projeto de complexidade média consistiu em um ambiente virtual para apoiar o processo de ensino e aprendizagem na capacitação a distância no sistema de zoneamento econômico e ecológico do estado de Minas Gerais. Nesse projeto, foram utilizados dois programadores, um administrador de banco de dados e uma pessoa da qualidade de software.

Os projetos foram desenvolvidos na linguagem de programação Java (J2EE - *Java 2 Enterprise Edition*), utilizando os *frameworks* MVC (*Model - View - Controller*), Spring[1] e VRaptor[2] e, na camada de *front-end*, foram utilizados Flex, JavaScript e JSP (*Java Server Page*). Oracle[3] e Postgres[4] foram utilizados como sistemas de gerenciamento de banco de dados. A comunicação entre os membros do time é constante e iterativa de acordo com o previsto na metodologia ágil Scrum. Isto quer dizer que as reuniões são realizadas diariamente com os membros do time em horários específicos e combinados entre eles em uma sala (peculiaridade, os membros ficam de pé). A comunicação é feita preferencialmente face a face ao invés da utilização de documentos escritos e o time de um determinado projeto é agrupada em uma sala a fim de aumentar a interação entre os seus membros (*Princípio Ágil 6*).

Com a integração entre a equipe de desenvolvimento e a equipe de teste, não é necessário aguardar a liberação das funções da *sprint* para que a equipe de testes inicie suas atividades. Em apenas um projeto, a equipe de teste utilizou a técnica TDD (*Test Driven Development*). Esse projeto foi o de menor complexidade, pois houve a necessidade de aprendizagem por parte da equipe de teste. Nos demais projetos, foram utilizados testes unitários e testes de comportamento. Assim que uma função do sistema estivesse concluída, o teste iniciava-se e, caso algum problema fosse encontrado, a solicitação da correção era imediata.

C. Aplicação da Adaptação do Scrum

O escopo do projeto é revisado a cada planejamento de *sprint* para garantir que a equipe dedique seus esforços no que é mais prioritário (*Princípio Ágil 7*). Em cada revisão, o cliente tem a liberdade de fazer ajustes e rever as prioridades das funções (*Princípio Ágil 2*). O planejamento das atividades a serem executadas é feito em uma sala de reunião. Geralmente, o planejamento das atividades inclui os membros da equipe e tem duração de cerca de oito horas, sendo dividida em três partes (*Princípio Ágil 10*). A primeira parte é o momento em que os membros da equipe definem o que será feito. A segunda parte do planejamento é para debater como as atividades serão desenvolvidas e para a equipe de desenvolvimento listar as tarefas necessárias para implementar as atividades planejadas. A terceira parte do planejamento é para estimar as atividades, baseando-se no consenso dos membros da equipe e valores entre 1 a 24 horas são utilizados para estimar essas atividades.

[1] http://www.springsource.org/
[2] http://vraptor.caelum.com.br/
[3] http://www.oracle.com/index.html
[4] http://www.postgresql.org/

A estimativa é feita pelos membros da equipe, utilizando cartas com sequência de Fibonacci (*planning poker*). Cada membro seleciona uma carta que acha correspondente a tarefa e, depois das cartas serem escolhidas, elas são expostas à mesa. Os membros que escolheram a menor e maior pontuação discutem os motivos que os levaram a escolha da estimativa. Em seguida, as cartas são jogadas novamente até que a equipe entre em consenso. Há situações em que valores diferentes da sequência de Fibonacci são escolhidos. A escolha destes valores diferentes ocorre quando a estimativa de uma tarefa ultrapassa três rodadas de cartas. Assim, os membros entram em um consenso utilizando valores intermediários.

No decorrer do desenvolvimento, a equipe realiza reuniões diárias e cada desenvolvedor relata o que foi feito e o que pretende fazer como próxima tarefa. Caso algum desenvolvedor relate algum impedimento, assuntos técnicos são discutidos de forma bem sucinta logo após a reunião diária. A ideia é nortear o desenvolvedor que esteja com algum impedimento para uma possível solução. O local da reunião diária é no próprio ambiente de desenvolvimento, onde informações que mostram o andamento do projeto são fixados nas paredes, tais como, *kanban*, *burndown*, *product backlog*, *sprint backlog* e relatório de erros (*Princípio Ágil 12*).

Essa gestão à vista tem como objetivo disponibilizar informações necessárias de uma forma simples e de fácil assimilação. Dessa forma, o trabalho torna-se menos árduo e a qualidade na produção do software melhora (*Princípio Ágil 9*). O horário das reuniões diárias não é fixo, alterna entre o início do turno da manhã e o início do turno da tarde. O horário é acordado entre os próprios desenvolvedores que possuem horários flexíveis de trabalho e, portanto, é possível ter a presença dos membros da equipe nas reuniões diárias.

Apesar do ambiente e da equipe serem autogerenciáveis (ou auto-organizáveis), existem pequenas atribuições e delegação de tarefas (*Princípio Ágil 11*). A função de controle pertence aos próprios membros da equipe como um todo, que escolhe a melhor maneira de trabalhar para cumprir os objetivos do projeto. Caso algum integrante encontre dificuldade em uma tarefa ou tenha algum impedimento, este recorre à equipe, que tentará ajudá-lo assim que tiver disponibilidade. O membro do grupo que tem conhecimento sobre o domínio em questão pode auxiliá-lo na dificuldade técnica.

Uma situação exemplo vivenciada entre os membros de um time com relação a esse auxílio foi a colaboração com um programador com pouco/nenhuma experiência em utilizar a biblioteca de processamento geográfico (*Princípio Ágil 5*). Um membro mais experiente na utilização dessa biblioteca percebeu que mais membros da equipe não tinham destreza na utilização dessa biblioteca e dedicou tempo para auxiliá-los na utilização da funcionalidade provida por essa biblioteca.

Em suma, as adaptações feitas foram basicamente a flexibilidade do horário das reuniões diárias e a presença integral de um membro do time (*Product Owner*) no cliente, em consequência da dificuldade em ter um representante do cliente envolvido no projeto (*Princípio Ágil 8*).

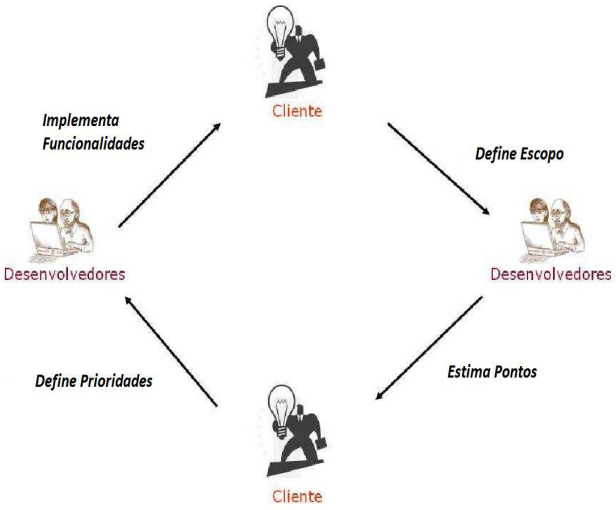

Figura 4 - Ciclo de Vida das Atividades dos Projetos (adaptado de [2])

III. LIÇÕES APRENDIDAS

Como citado anteriormente, as práticas ágeis podem evidenciar problemas no decorrer do processo de implantação da metodologia ágil. Conforme é descrito pela metodologia ágil, a

Figura 4 ilustra o ciclo de vida de um projeto e a iteração entre cliente e desenvolvedores: i) o cliente define o escopo; ii) os desenvolvedores estimam prazos; iii) o cliente define as prioridades; iv) os desenvolvedores implementam as funções; e iv) retorna ao início do ciclo onde o cliente redefine o escopo até que a funcionalidade seja entregue.

Na experiência vivida, foi evidenciado, no que se refere à análise de negócio, que a falta de experiência com as práticas ágeis causou problemas na interpretação das necessidades dos clientes. Isso refletiu no desenvolvimento do software, pois os membros do time de desenvolvimento não tinham acesso direto ao cliente. Uma solução foi a criação de mais artefatos, como o *workflow* da regra de negócio, com a presença do cliente, *Product Owner* e dos membros do time de desenvolvimento.

Outro fato evidenciado foi a necessidade da colaboração e do interesse dos membros do time. A eficiência deles esteve diretamente relacionada ao comprometimento de cada colaborador e, caso houvesse algum indivíduo que não compartilhasse da mesma ideia e filosofia, havia um razoável desgaste no restante do time, deixando um membro do time sobrecarregado em relação aos demais.

Eventualmente, alguns indivíduos com dificuldade de trabalhar em equipe e sem característica de pró-atividade se recusam a auxiliar outros integrantes da equipe com o argumento de que não é tarefa deles. Há ainda, falta de comprometimento por parte de alguns em resolver as tarefas que eles "pegaram para fazer" as quais tornam-se impedimento, considerando a dependência entre as tarefas,

para outros membros da equipe.

Ressalta-se que os membros do time são compostos por profissionais e estagiários, sendo que os estagiários são estudantes da Instituição de Ensino Superior em que o Laboratório de Pesquisa e Desenvolvimento está localizado. Apesar das dificuldades encontradas pelos estudantes, pôde-se perceber que houve crescente curva de aprendizagem, pois eles aprenderam a lidar com as práticas que envolve um processo de desenvolvimento de software, por exemplo, documentação, teste, implementação e reuniões com o cliente.

Dos sete projetos mencionados nesta experiência de implantação de uma adaptação da metodologia ágil Scrum, apenas dois (dos primeiros) tiveram prazos estourados, os quais foram renegociados com o cliente. Acredita-se que esse atraso foi em decorrência da inexperiência dos membros do time em utilizar a metodologia ágil Scrum. Assim os principais motivos do atraso desses dois projetos foram (i) falta de envolvimento de parte da equipe, (ii) integrantes que se comprometiam mais do que podiam e (iii) pouca transparência e comunicação dentro da equipe.

Uma forma que evitou atrasos foi melhorar o critério de seleção dos membros do tipo, o que possibilitou sucesso maior.

Como lições aprendidas, foi possível notar que para a implantação da metodologia ágil Scrum em uma organização é recomendado (i) fazer uma apresentação sobre o *framework* para as pessoas que vão trabalhar diretamente no projeto (time), para que os envolvidos saibam como funciona e o que eles terão de tarefas agregadas no dia a dia, (ii) aconselhar a implantação do *framework* em um projeto piloto com o objetivo de selecionar as pessoas, (iii) observar os possíveis impactos, (iv) estruturar treinamento e (v) avaliar os benefícios da metodologia ágil Scrum adotada e implantada. Durante a implantação, práticas como integração contínua e desenvolvimento baseado em teste (TDD - *Test Driven Development*) são oportunidades de melhorar a qualidade de software e facilitar o refatoração das aplicações (o que pode ser percebido no projeto em que o TDD foi utilizado).

Cabe ressaltar que o Scrum tem o objetivo de mostrar os problemas da organização. Os problemas são impedimentos, que no início, podem gerar desgastes desnecessários por falta de experiência até o completo entendimento do que é um real impedimento. Os maiores problemas encontrados foram relacionados à pró-atividade e à colaboração dos membros do time. Há pessoas que sentem a necessidade de uma outra para cobrar e solucionar os problemas que surgem, o que mostra falta de comprometimento, individualismo e iniciativa. Pessoas que pensam como indivíduo e não como equipe devem ser excluídas do time.

Pelo que pôde ser visto em “adaptações” no *framework*, não se deve tirar algumas de suas características essenciais. Por exemplo: i) o intenso ciclo de comunicação que garante o alinhamento das expectativas; ii) a constante revisão de escopo que garante a coesão no projeto; iii) as entregas intermediárias que buscam o atendimento da necessidade do cliente e a correção de falhas antes do final do projeto; e iv) o conceito de *timebox* que força a equipe a ter o que entregar evitando atrasos.

Esses benefícios são provenientes dos princípios embutidos da metodologia ágil Scrum e ao adaptá-la é preciso manter esses princípios de modo a trazer o que há de melhor no *framework*: i) forte interação entre as pessoas; ii) comunicação; iii) comprometimento; e iv) produto.

IV. Considerações Finais

O desenvolvimento de software utilizando metodologia ágil está se tornando uma realidade cada vez maior no cotidiano das empresas desenvolvedoras de software. A agilidade agrega qualidade no processo de desenvolvimento e da gerência de software. Para agregar valor ao software final, deve-se ter uma equipe bem estruturada e seguir as metodologias e as estratégias corretas.

Hoje, no Brasil, existe grande interesse no desenvolvimento de novos sistemas de softwares e patentes para que a produção tecnológica nacional possa ser compatível com a produção científica, medida em artigos nacionais e internacionais. Nesse sentido, a adoção de técnicas modernas de desenvolvimento e de gerenciamento de projetos de software, por exemplo a metodologia ágil Scrum, pode ajudar a diminuir esta diferença, estabelecendo uma ponte entre a ciência de qualidade e produtos que efetivamente resolvam problemas da realidade empresarial nacional.

Assim, o que a metodologia ágil Scrum traz é um processo de desenvolvimento iterativo e incremental para gerenciamento de projetos e desenvolvimento ágil de software sustentado por quatro pilares: i) os indivíduos e suas interações acima de procedimentos e ferramentas; ii) o funcionamento do software acima de documentação abrangente; iii) a colaboração dos clientes acima da negociação de contratos; e iv) a capacidade de resposta à mudanças acima de um plano pré-estabelecido.

No contexto do Laboratório de Desenvolvimento e Pesquisa, houve pleno apoio na implantação da metodologia ágil Scrum por parte dos pesquisadores seniores ("gerentes de projetos"), bem como a infraestrutura básica estava adequada e os participantes dos projetos (membros do time) entenderam a nova proposta e incorporaram nas suas tarefas. Cabe ressaltar que a utilização da metodologia ágil Scrum contribui(u) na formação dos membros do time os quais são estagiários (estudantes) e/ou profissionais autônomos (*freelancer*).

Quanto a formação dos times no Laboratório, eles costumam ser constituídos de quatro a sete membros, o que acarreta facilidade na comunicação. Uma adaptação significativa foi(é) a inversão da semântica do *product owner*, pois ele é um membro do Laboratório alocado no cliente. Isso acontece em decorrência da dificuldade de ter um cliente no Laboratório.

Com base na análise da implantação da metodologia ágil

Scrum para o desenvolvimento dos projetos, foram perceptíveis melhorias no gerenciamento e no desenvolvimento dos projetos de software, o que garantiu maior visibilidade ao seu andamento. O envolvimento e o comprometimento dos membros da equipe perante os resultados aumentaram, permitindo trabalho mais colaborativo.

Percebeu-se, também, que os membros das equipes estavam motivados e abertos às mudanças no trabalho, o que facilitou o processo de implementação/adequação da metodologia ágil Scrum. Assim, permitiu-se amadurecimento e busca por melhorias no processo, de modo a atender as particularidades de cada projeto.

Próximos passos são a consolidação das práticas adotada, quando na implantação da metodologia ágil Scrum, por meio de adequações e correções de desvios identificadas ao longo do desenvolvimento dos sete projetos e utilização de métricas para avaliar formalmente o ganho obtido com a utilização da metodologia ágil Scrum.

REFERÊNCIAS

[1] Abrahamsson, P.; Salo, O.; Ronkainen, J.; Warsta, J. Agile Software Development Methods – Review and Analysis. VTT Publication 478. 107 p. 2002. Disponível em: <http://www.vtt.fi/inf/pdf/publications/2002/P478.pdf>. Acesso em: 21 jan 2009.

[2] Bona, C. Avaliação de Processos de Software: Um Estudo de Caso em XP e ICONEX. Dissertação (Mestrado em Engenharia de Produção) – Universidade Federal de Santa Catarina, Florianópolis – 2002. Disponível em: <ftp://www.ufv.br/dpi/mestrado/Gerais/TeseIconix.pdf>. Acesso em: abr.2011.

[3] Fowler, M. The New Methodology. 2005. Disponível em: <www.martinfowler.com/articles/newMethodology.html>. Acesso em: abr.2011.

[4] Leitão, M. V. Aplicação de Scrum em Ambiente de Desenvolvimento de Software Educativo. Monografia (Trabalho de Conclusão de Curso) – Universidade Federal de Pernambuco, Recife – 2010. Disponível em: <http://dsc.upe.br/~tcc/20101/TCC_final_Michele.pdf>. Acesso em: abr.2011.

[5] Schwaber, K.; Sutherland, J. The Scrum Guide. 2010. Disponível em: <http://www.scrum.org/scrumguides/>. Acesso em: mar. 2011.

[6] Hiranabe, K. Kanban Applied to Software Development: from Agile to Lean. 2008. Disponível em: <http://www.infoq.com/articles/hiranabe-lean-agile-kanban>. Acesso em: abr. 2012.

[7] Murphy, C. Adaptive Project Management Using Scrum. In: Methods & Tools - Software Development Magazine - Programming, Software Testing, Project Management, Jobs. 2004. Disponível em: <http://www.methodsandtools.com/archive/archive.php?id=18>. Acesso em: abr. 2012.

[8] Beck, K.; Beedle, M.; Bennekum, A. van; 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.; Thomas, D. Manifesto for Agile Software Development. 2001. Disponível em: <http://www.agilemanifesto.org/>. Acesso em: 16 abr.2012.

[9] Vijayasarathy, L. R.; Turk, D. Agile Software Development: A Survey of Early Adopters. In: Journal of Information Technology Management, v. XIX, n. 2, p. 1-8. 2008.

[10] Sutherland, J.; Viktorov, A; Blount, A. Distributed Scrum: Agile Project Management with Outsourced Development Teams. In: Proceedings of the 40th Hawaii International Conference on System Sciences, 40, 2007.

[11] Catunda, E.; Nascimento, C.; Cerdeiral, C.; Santos, G.; Nunes, E.; Schots, N. C. L.; Schots, M.; Rocha, A. R. Implementação do Nível F do MR-MPS com Práticas Ágeis do Scrum em uma Fábrica de Software. In: X Simpósio Brasileiro de Qualidade de Software (SBQS). 2011. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/sbqs/2011/SBQS2011-RE10_82940.pdf>. Acesso em: 23 abr 2012.

[12] Salgado, A.; Melcop, T.; Acchar, J.; Rego, P. A.; Ferreira, A. I. F.; Katsurayama, A. E.; Montoni, M.; Zanetti, D. Aplicação de um Processo Ágil para Implantação de Processos de Software baseado em Scrum na Chemtech. In: IX Simpósio Brasileiro de Qualidade de Software (SBQS). 2010. Disponível em: <http://www.lbd.dcc.ufmg.br/colecoes/sbqs/2010/RL10_alex_salgado.pdf >. Acesso em: 23 abr 2012.

[13] Cockburn, A. Agile Software Development. Addison-Wesley Professional. 304 p. 2001.

[14] Cockburn, A. Crystal Clear – A Human-Powered Methodology for Small Teams, including The Seven Properties of Effective Software Projects. 2004. Disponível em: <http://st-www.cs.uiuc.edu/users/johnson/427/2004/crystalclearV5d.pdf>. Acesso em: 17 fev 2009.

[15] Highsmith, J. (2002); Agile Software Development Ecosystems; Publisher: Addison Wesley; Pub Date: May 26, 2002; ISBN: 0-201-76043-6; Pages: 448.

[16] Paetsch, F., Eberlein, A., and Maurer, F. Requirements Engineering and Agile Software Development. In Proceedings of the Twelfth international Workshop on Enabling Technologies: infrastructure For Collaborative Enterprises (June 09 - 11, 2003). WETICE. IEEE Computer Society, Washington, DC, 308.

[17] Filho, D. L. B. Experiências com desenvolvimento ágil, Instituto de Matemática e Estátistica da Universidade São Paulo, 170p, Dissertação de mestrado.

[18] Portela, C. S. Uma proposta de gerenciamento ágil dos projetos de desenvolvimento de software do CTIC / UFPA, Instituto de Ciências Exatas e Naturais – Faculdade de Computação – Universidade Federal do Pará; 94p, Dissertação de mestrado.

[19] Coram, M; Bohner, S. The Impact of Agile Methods on Software Project Management. 2005.

[20] Awad, M. A. A Comparison between Agile and Traditional Software Development Methodologies.

Technical Report. University of Western Australia. 77p. 2005.

[21] Palmer, S. R.; Felsing, J. M. A Practical Guide to Feature-Driven Development. Prentice-Hall. 304 p. 2002.

[22] Hunt, J. Agile Software Construction. Springer. 254 p. 2005.

[23] Lindstrom, L.; Jeffries, R. Extreme Programming and Agile Software Development Methodologies. In: Information Systems Management, v. 21, Issue 3, pp 41-52. 2004.

[24] Beck, K. Embracing Change with Extreme Programming. In: Computer, v. 32, Issue 10, pp 70-77. 1999.

[25] Costa Filho, E.; Penteado, R. A. D.; Silva, J.; Braga, R. Padrões e Métodos Ágeis: Agilidade no Processo de Desenvolvimento de software. In: 5th Latin American Conference on Pattern Languages of Programming. August 16-19, 2005.

[26] Schwaber, K. SCRUM Development Process. In: Object-Oriented Programming, Systems, Languages, and Applications – Workshop on Business Object Design and Implementation. October 15-19, 1995. Disponível em: <http://www.jeffsutherland.com/oopsla/schwapub.pdf>. Acesso em: 17 fev 2009.

[27] Schwaber, K.; Beedle, M. Agile Software Development with SCRUM. Prentice-Hall. 158 p. 2001.

[28] Sutherland, J.; Schwaber, K. The Scrum Papers: Nut, Bolts, and Origins of an Agile Framework. 224p. 2011. Disponível em: <http://jeffsutherland.com/ScrumPapers.pdf>. Acesso em: 24 abr 2012.

[29] Tekool.net. Printable Agile Planning Poker. Disponível em: < http://www.tekool.net/blog/2009/07/21/printable-agile-planning-poker/>. Acesso em: abr. 2012.