DOI:10.5748/9788599693131-14CONTECSI/RF-4593

EVALUATION OF SOFTWARE DEVELOPMENT METHODOLOGIES AND SCRUM APPLICATION IN A SMALL BUSINESS

Márcio Aurélio Ribeiro Moreira (Faculdades Pitágoras de Uberlândia, Minas Gerais, Brasil) – marcio.moreira@pitagoras.com.br
Natália Rios Guimarães (Faculdades Pitágoras de Uberlândia, Minas Gerais, Brasil) – nataliariosguimaraes@gmail.com

This paper shows the importance of using a software development methodology, particularly agile methodologies for small software and teams. The next section presents a bibliographic study of cascade, RUP, XP and Scrum methodologies. For each methodology are highlighted: the ideals of each of them, a brief history, how they work, and their main characteristics, its positive and negative points. Following, the paper presents a case study of the implementation of Scrum is presented in a small software development company in Araxá - MG, which had problems of late submissions, rework, customer and staff dissatisfaction with the results. With the use of Scrum, the acceptance project delivery without rework improved from 40% to 70% and the delivery project on time improved from 10% to 60%.

Keywords: Scrum, XP, RUP, Cascade, Software Engineering.

AVALIAÇÃO DE METODOLOGIAS DE DESENVOLVIMENTO DE SOFWARE E ESTUDO DA APLICAÇÃO DO SCRUM EM UMA PEQUENA EMPRESA

Este trabalho mostra a importância da utilização de uma metodologia de desenvolvimento de software, principalmente de metodologias ágeis para softwares e equipes pequenas. O texto apresenta um estudo bibliográfico das metodologias Cascata, RUP, XP e Scrum. Para cada metodologia são destacados: os ideais de cada uma delas, um breve histórico, como elas funcionam, suas principais características, seus pontos positivos e negativos. Na sequência é apresentado um estudo de caso da implantação do Scrum em uma pequena empresa de desenvolvimento de software de Araxá - MG, que tinha problemas de entregas fora do prazo, retrabalho e insatisfação dos clientes e da equipe com os resultados. Com o Scrum a aceitação da entrega do projeto sem retrabalho melhorou de 40% para 70% e o cumprimento de prazo melhorou de 10% para 60%.

Palavras-chave: Scrum, XP, RUP, Cascata, Engenharia de Software.

1. Introdução

Como informado por Nuno Simões (2013, p. 1), diretor do Grupo de Software e Serviços para a America Latina da Intel, o Brasil atualmente é o sétimo maior no mercado de software mundial. A estimativa é que em 2015 existirão mais de quinhentos mil desenvolvedores no Brasil, o que transforamará o país na sexta comunidade de desenvolvimento de software do planeta. No mercado de software para equipamentos móveis (*smartphones* e *tablets*), as pequenas empresas disputam em condições de igualdade com as grandes empresas, mesmo no mercado internacional.

Os projetos de TI (Tecnologia da Informação) e de desenvolvimento de software ainda têm uma taxa de sucesso baixa, como mostra aFigura 1, feita com base na pesquisa de 50 mil projetos, de totos os continentes, concluídos entre 2004 e 2012, feita pelo Standish Group (2013, p. 1). O Standish Group é um instituto de pesquisa especializado em projetos de TI e de software. Note que em 2012 a taxa de sucesso chega a 39% dos projetos, mas ainda existem 18% que falham (não são concluídos)e outros 43% que extrapolam as previsões de custo e/ou prazo.

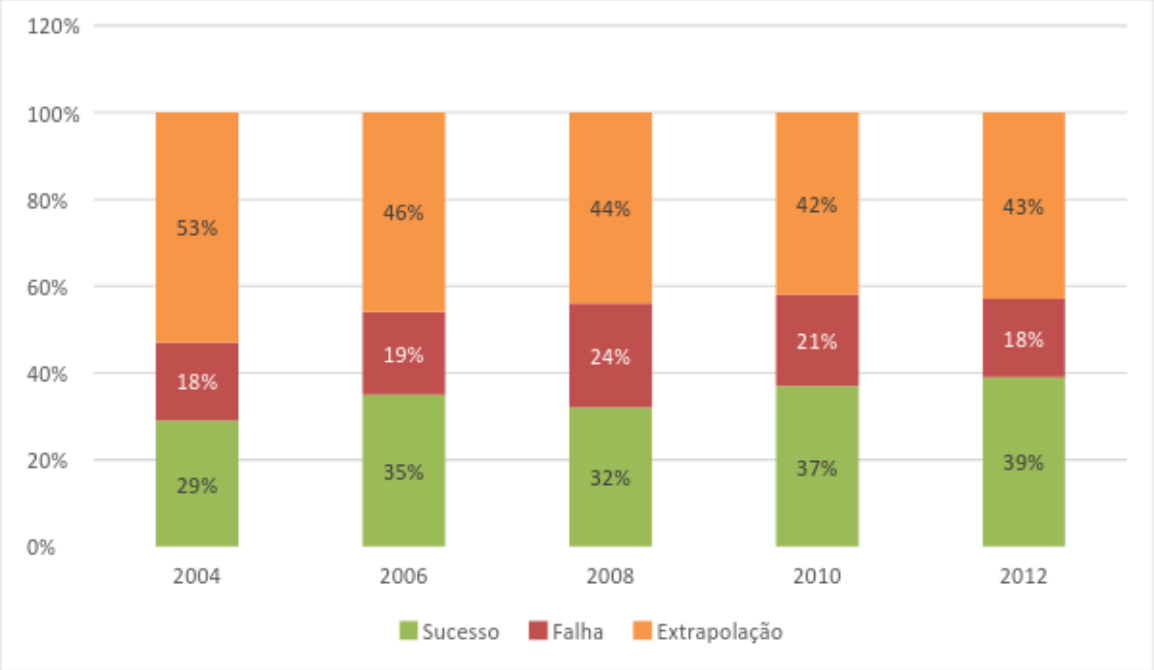

Figura 1: Resultados dos projetos de TI e software (adaptado de Standish Group, 2013, p.5)

Ainda segundo Standish Group (2013, p. 3), como mostrado na Figura 2, as principais causas de falhas dos projetos são: falta de suporte executivo (20%), falta de envolvimento dos usuários (15%), falta de otimização de recursos (15%), falta de perfil adequado dos recursos (13%), falta de expertise dos gerentes de projeto (12%), falta do uso de métodos ágeis (10%), falta de clareza dos objetivos (6%), falta de maturidade emocional da equipe (5%), falhas de execução (3%), ferramentas e infraestrutura (1%). As questões reportadas pelo Standish Group estão ligadas ao projeto e a complexidade relacionada à fabricação de software (engenharia de software).

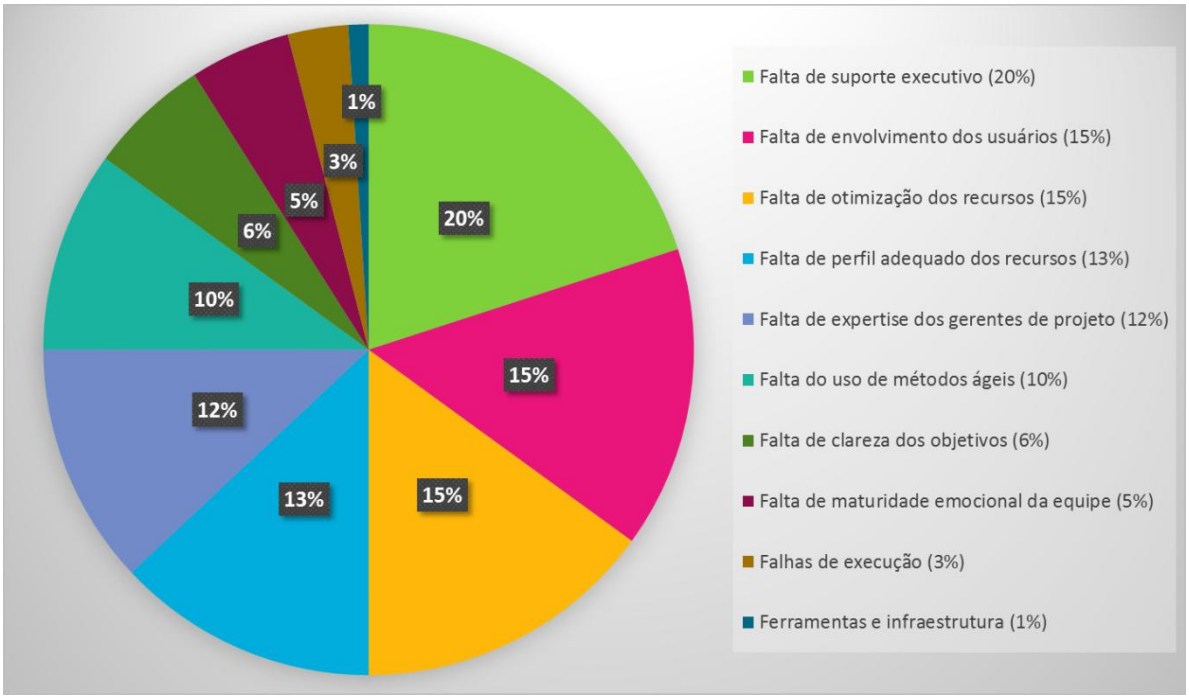

Figura 2: Causas de falhas dos projetos de TI e software (adaptado de Standish Group, 2013, p.5)

Para a fabricação de software, como mostra o trabalho de Moura & Moreira (2012, p. 2), existem no mercado várias metodologias que podem ser utilizadas, dentre elas podemos citar: Cascata, RUP, XP (*eXtreaming Programming*), Scrum, etc.

Este artigo visa investigar se o Scrumpode ser adequado para uma pequena empresade Araxá que tem uma pequena equipe de desenvolvedores. Para isto será investigado se o Scrum será eficiente para resolver os problemas da empresa e trará bons resultados para os clientes e para a equipe.

2. Metodologias de Engenharia de Software

O uso de metodologias modernas de desenvolvimento de software vem ganhando espaço para resolver a complexidade envolvida neste trabalho, conforme observado por Pressman (2011, p.52)ao realizar um processo e necessário seguir uma série de passos previsíveis para obter um resultado de qualidade e dentro do prazo estabelecido.

Ainda segundo Pressman (2011, p.52) esses passos são definidos por engenheiros e seus gerentes e os mesmos adaptam esses as suas necessidades e então o seguem.Pressman (2011, p.52) também afirma que isso faz com que o desenvolvimento tenha estabilidade, controle e organização para uma atividade evitando dessa forma falhas. Porém, uma metodologia de software moderna deve ser ágil. Além disso deve demandar apenas aquilo que é apropiado para o projeto que está sendo produzido.

O desenvolvimento ágil defende: entrega incremental, satisfação do cliente, equipes pequenas e motivadas, métodos informais e simplicidade no desenvolvimento geral. Para isso todos os envolvidos trabalham em conjunto em uma equipe que se auto organiza e controla seu próprio destino (PRESSMAN, 2011, p.81).

Ainda segundo Pressman (2011, p. 52 e 81), o desenvolvimento ágil é importante

porque o ambiente dos sistemas é dinâmico e as necessidades dos produtos de software estão em constante mundança.A metodologia deve ser escolhida dependo do software a ser desenvolvido. Pois um processo pode ser apropiado para a construção de um sistema de um avião, enquanto o mesmo processo poderia não ser indicado para criação de um site.

A seguir serão mostradas as principais metodologias de desenvolvimento de software.

2.1. Cascata

Essa foi a primeirametodologia de desenvolvimento de software. No modelo Cascata as atividades são estruturadas de forma que uma tarefa só possa ter início após o término da anterior (Moura &Moreira,2012, p. 2).

Segundo Pressman (2011, p.59) deve-se aplicar esse modelo quando os requisitos são bem compreendidos e não tendem a ter mudanças. Essa metodologia sugere uma abordagem clássica e sequencial, iniciando com o levantamento de requisitos, passando pelas fases de planejamento, modelagem, construção, utilização e por fim suporte e manutenção contínuado software.

Segundo Moura & Moreira (2012, p. 3), o modelo Cascata é composto pelas seguintes fases:

a) Levantamento de requisitos: essa é a primeira fase onde será criado um documento que terá definido quais objetivos, regras, restrições, funcionalidades e necessidades do sistema.
b) Análise: onde é produzido um documento que especificará a descrição completa do sistema.
c) Projeto: nessa fase e definido como o software deve ser implementado. Além de se produzir documentos de especificação técnica.
d) Codificação e testes: o sistema é desenvolvido numa linguagem de programação. Após essa etapa as funcionalidades são testadas e os ajustes necessários são realizados.
e) Entrega e manutenção: nessa faseo sistema é entregue ao cliente, os usuários envolvidos são treinados e inicia-se a utilização do sistema. Qualquer alteração ou ajuste após a entrada em produção recomenda-se tratar via projeto de manutenção.

Segundo Moura & Moreira (2012, p. 2), apud A. Carlos, os principais benefícios do modelo Cascata são:

> O fato de ser dividido em etapas traz organização e orienta melhor as equipes de desenvolvimento antes de iniciar a codificação dos programas. Este modelo preza também o planejamento, principalmente no início, reduzindo assim a necessidade de planejamento contínuo conforme o andamento.

Por outro lado, segundo Moreira (2010, p. 16), o modelo Cascata apresenta os seguintes problemas: o cliente só participa no início (fase de levantamento de requisitos) e no final (fase de testes e de entrega) do processo; os erros de especificação são detectados somente durante a fase de testes (tardiamente); como os requisitos da maioria dos sistemas

não são estáticos (mudam ao longo do tempo), os riscos são mitigados somente no final do projeto. Consequentemente, se o projeto é longo o cliente se sente à margem do processo e geralmente quando vai testar o software aparecem os erros de especificação e/ou os requisitos de negócio já mudaram.

2.2. RUP

Segundo Sommervile (2007, p.54) o RUP (*Rational Unified Process*) é um exemplode processo de desenvolvimento de software moderno que foiderivado do trabalho da UML[1] e do Processo Unificado de desenvolvimento de software[2].

Essa metodologia foi criada por Grady Booch, James Rumbaugh e Ivar Jacobson, na Rational Software Corporation, que foi vendida para a IBM entre dezembro de 2002 e fevereiro de 2003. Esta metodologia utiliza o modelo iterativo e incremental, evitando os problemas do modelo Cascata. Ela é composta de 4 fases, cada fase com uma, duas ou mais iterações (conforme o tamanho do projeto), a cada iteração o conteúdo do software cresce de forma incremental, o que reduz o impacto de mudanças dos requisitos. O RUP pode ser aplicado a qualquer projeto seja ele pequeno, médio ou grande, desde que o método seja adaptado ao projeto (MOREIRA, 2010, p. 26).

As fases do RUP são as apresentadas na Figura 3, são elas: concepção, elaboração, construção e transição. Segundo Sommervile (2007, p. 54-55), os objetivos das fases do RUP são:

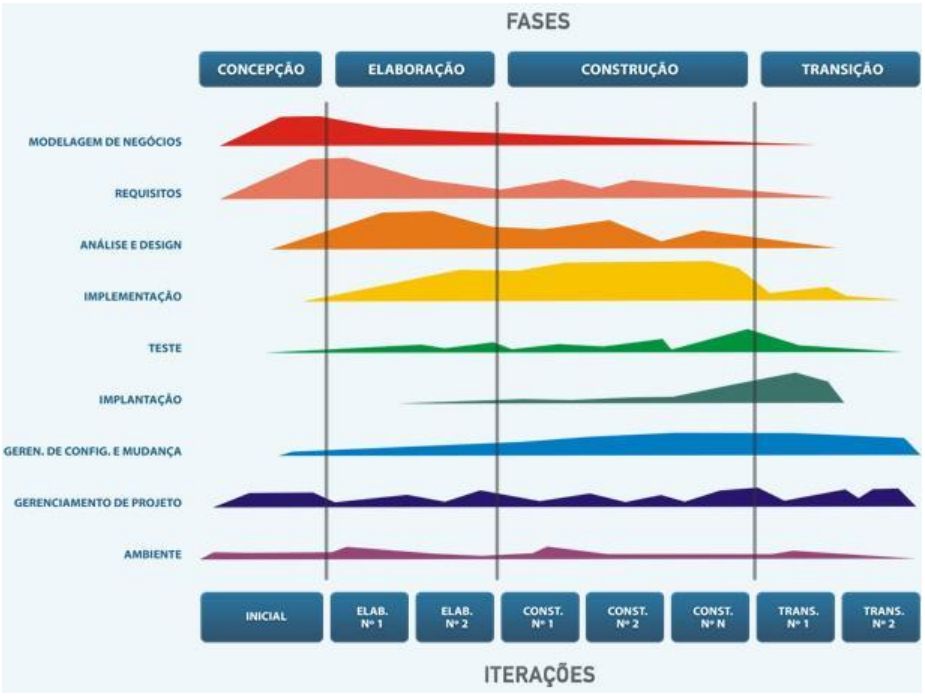

Figura 3: Fases do RUP (Adaptado de MOREIRA, 2010, p. 27)

1. Concepeção: o objetivo dessa fase é fornecer um caso de negócio para o sistema. O

[1]A UML (*Unified Modelling Language*) é uma linguagem ou notação de diagramas para especificar, visualizar e documentar modelos de 'software' orientados a objetos.

[2] É o conjunto de atividades necessárias para transformar requisitos do usuário em um sistema de software.

caso de negócio é um estudo de retorno, ou seja, ele avalia quanto será gasto no projeto de desenvolvimento e quanto o projeto trará de benefícios. Para isso é necessário identificar todos os envolvidos com o sistema e definir as iterações que cada um terá com o mesmo.

2. Elaboração: essa fase tem como foco entender o problema que está sendo exposto, estabelecer um *framework*[3]de arquitetura para o sistema, desenvolver o plano do projeto e identificar os fatores de risco. Ao final dessa fase deve existir um modelo de requisitos para o sistema (casos de uso da UML são especificados), uma descrição da arquitetura e um plano de desenvolvimento de software.
3. Construção: essa fase está relacionada ao projeto, programação e teste de sistema. O desenvolvimento do sistema é realizado paralelamente e integrado durante essa fase. Ao encerrar esse ciclo é necessário ter um sistema funcionando assim como a documentação associada a ele para ser liberada para o usuário.
4. Transição: é nesta fase que o sistema é transferido do desenvolvimento para os usuários, ou seja, o software é colocado em produção. Portanto, no final dessa fase deve-se ter um sistema documentado e funcionando corretamente em produção.

Segundo Sommervile (2007, p. 55), as iterações no RUP podem ocorrer de duas maneiras: por fase ou por projeto, conforme mostrado na Figura 4. Cada fase pode ser realizada de forma iterativa, neste tem-se uma ou mais iterações por fase com os resultados alcançados incrementalmente. Outra possibilidade é que o conjunto total de fases pode ser realizado de forma iterativa e incremental.

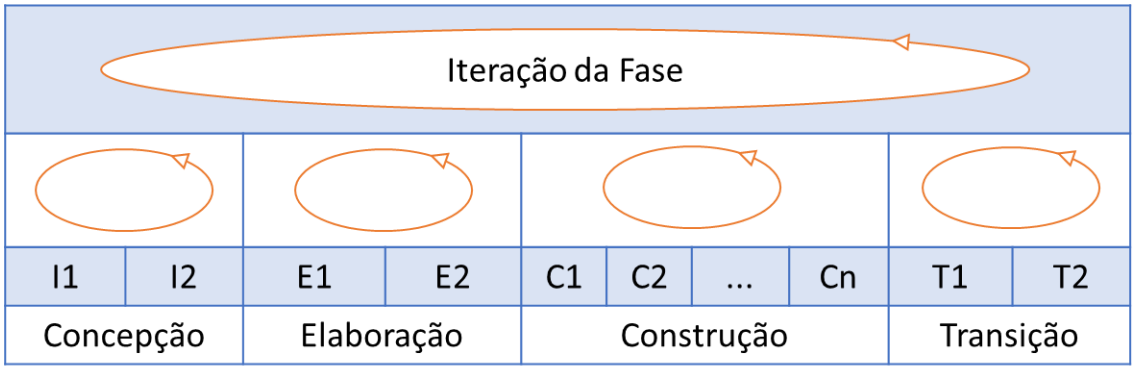

Figura 4: Iterações do RUP (Adaptado de SOMMERVILE, 2007, p. 57)

Segundo Moreira (2010, p. 31), o RUP é composto das seguintes disciplinas: modelagem de negócios (compreender a engenharia do negócio), requisitos (explicitar e coletaros requisitos), análise e projeto (projetar como os requisitos serão implementados), implementação (desenvolver, organizar, testas unitariamente e integrar os componentes do software), testes (testarde forma sistêmica a qualidade do software desenvolvido), distribuição (planejar e executar a distribuição e instalação do software), gestão de configuração e mudanças (cuidar do controle e sincronização dos componentes do software), gestão do projeto (planejar, executar, monitorar, controlar e encerrar o projeto de software) e ambiente (adequar o processo ao projeto, preparar e manter os ambientes

[3]*Framework* é um conjunto de classes que incorpora um projeto abstrato de soluções para uma família de problemas relacionados

necessários ao desenvolvimento).

Sommervile (2007, p. 55), cita seis boas práticas de desenvolvimento de software recomendadas pelo RUP:

1. Desenvolver o software iterativamente: planejar a construção do sistema com base na prioridade do cliente para que essas sejam entregues primeiro.
2. Gerenciar requisitos: documentar os requisitos e acompanhar sempre as mudanças. Além de analisar o impacto delas antes de aceitá-las.
3. Arquiteturas baseadas em componentes: pois assim é possível reaproveitar algo pronto economizando portanto tempo no desenvolvimento.
4. Modelar o software visualmente: usar modelos gráficos como a UML para visualizar o sistema estático e dinamicamente.
5. Verificar a qualidade do software: garantir que o sistema atenda os padrões de qualidade da organização.
6. Controlar as mudanças no software: utilizar um sistema de gerencimento de mudanças para controlar as mesmas.

Ainda segundo Sommervile (2007, p.56), o RUPé um processo que pode ser adaptado a todos os tipos de desenvolvimento. Entretanto, para projetos de menor envergadura este processo pode ficar pesado. Mas representa uma nova geração de processos sendo a mais importante a separação de fases e disciplinas e o reconhecimento que a implantação também faz parte do processo.

2.3. XP (Extreme Programming)

Segundo Pressman (2011, p.87) os primeiros trabalhos relacionados ao XP ocorreram no final dos anos oitenta, o trabalho que originou o tema foi escrito por Kent Beck. Nesse trabalho foi estabelicido os valores do XP comunicação, simplicidade, *feedback*, coragem e respeito. Segundo Sommervile (2007, p. 263), do grupo de metodologias ágis, o XP é um dos mais conhecidos e utilizados.

Para Pressman (2011, p.88-90), o XP tem uma abordagem orientada a objetos que é composto por quatro atividades metodológicas: planejamento, projeto, codificação e testes que são explicadas a seguir. A Figura 5 mostra estas atividades. Em planejamento os requisitos são coletados, em projeto eles são preparados para a codificação, em codificação o código fonte do software e finalmente em testes, o softwre é testado.

Figura 5: Atividades do XP (Adaptado de PRESSMAN, 2011, p. 88)

No planejamento é realizado o levantamento de requisitos, onde os membros técnicos são capacitados a entender o ambiente de negócio para ter uma percepção sobre os resultados solicitados, fatores principais e funcionalidades. Cada requisito criado é

chamado de histórias de usuários e, a partir disso, o cliente atribui uma prioridade para a construção dessa funcionalidade. Depois disso os membros da equipe XP avaliam cada história e atribuem um custo à mesma que é medido em semanas. Caso a história tenha mais que três semanas é solicitado ao cliente para realizar a divisão desta em histórias menores.

Após todas as histórias estarem listadas, clientes e desenvolvedores trabalham em conjunto para decidir como agrupar as histórias para o próximo incremento além de definirem prazos e custos.

Tendo os itens citados acima definidos, Pressman (2011, p.89) afima que a equipe XP pode ordenar as histórias em uma das três formas:

1. Todas serão implementadas imediantemente em um prazo de poucas semanas.
2. As histórias de maior valor serão implementadas primeiro.
3. As histórias de maior risco serão implementadas primeiro.

Após o desenvolvimento das primeirashistórias terem sido concluídas é entregue a primeira versão do software. Assi é realizado um cálculo da velocidade do projeto. Isso é realizado para ajudar estimar prazos para as próximas versões e determinar se foi assumido um comrpomisso exagerado ou subestimado para todas as histórias ao longo de todo o projeto de desenvolvimento. Se ocorrer um exagero, o conteúdo das versões e modificado e os prazos para entrega alterados. (PRESSMAN, 2011, p.89).

Conforme afirma Pressman (2011, p.89) ao desenvolver do trabalho o cliente pode acrescentar histórias, mudar o valor de uma existente, dividir algumas ou eliminá-las. Consequentemente a equipe XP reconsidera todas as versões remanescentes e altera seus planos.

Na atividade de projeto segundo Pressman (2010, p.64) é importante manter a simplicidade. Além disso, o projeto fornece diretrizes de implementação para uma história apenas no que foi exigido nada mais nada menos.

Na metodologia XP quando uma história é considera complexa é feito um protótipo da mesma demostrando a solução pronta. Isto é feito para reduzir os riscos em relação a implementação.

O XP apoia também o uso de refabricação que é o processo de modificar um sistema não alterando o comportamento externo do código, mas aperfeicoando a estrutura interna. É uma forma disciplinada e simplificada de modificar o sofware diminuindo assimas chances de introdução de defeitos. (PRESSMAN, 2010, p.65).

Pressman (2010, p.65) assegura que os projetos no XP podem ser continuamente modificados a medida que a construção prossegue. A refrabicação tem como intenção controlar essas modificações sugerindo pequenas alterações de projeto que podem aperfeiçoar muito o projeto, ou seja, ocorre tanto antes quanto depois que a codificação é iniciada.

Outra atividade é a codificação, é recomendado que essa atividade tenha inico após a equipe projetar uma série de testes unitários[4] que exercitarão cada uma das histórias que devem ser incluidas na versão atual (incremento de software). Quando a codificação estiver concluída, os testes unitários deverão ser executados. (PRESSMAN, 2010, p. 65).

O XP segundo Pressman (2010, p.65) recomenda programação em pares que nada mais

[4] São testes aplicados para verificar se os retornos oferecidos pelo software são corretos.

é do que dois desenvolvedores trabalharem em apenas uma estação de trabalho. Isso porque duas cabeças trabalhando são melhores do que uma. Além disso, desta forma os desenvolvedores ficam focados no problema. Esses dois programadores apresentam atividades ligeiramente distintas enquanto um poderia pensar nos detalhes do código o outro garante que as normas de codificação estão sendo seguidas.À medida que os pares de programadores completam seu trabalho, o trabalho desenvolvido por eles é integrado ao trabalho dos outros.

A última atividade metodológica são os testes onde são realizados vários tipos de testes. Um deles é o teste unitário, porém nessa etapa ele deve ser realizado de forma automatizada, visto que na codificação eles podem ser realizados de forma manual. Outros tipos de testes que são realizados são: integração e validação, que podem ocorrer diariamente, isso auxilia no acopanhamento da qualidade do sistema. Existe também o teste de aceitação, ele é realizado pelo cliente onde o mesmo busca verificar se as funcionalidades foram atendidas ou não. Esses testes são derivados das histórias de usuários. (PRESMMAN, 2010, p. 66)

Segundo Moura & Moreira (2012, p. 14), o XP é mais para equipes pequenas, onde não há uma grande necessidade de documentação, as pessoas estão próximas e existe uma certa indefinição dos requisitos. Por outro lado, quando o cliente exige algum outro método ou não existe a possibilidade de trabalho cooperativo, devido a limitações de ferramentas, o XP torna-se inadequado.

2.4. Scrum

De acordo com Pressman (2010, p. 69), o Scrum é uma metodologia de desenvolvimento ágil, essa expressão é derivada de uma atividade que o ocorre durante o jogo de rúgbi. O Scrum foi desenvolvido por Jeff Sutherland e por sua equipe na década de noventa.Os princípios do Scrum, listados abaixo, são condizentes com o manifesto ágil:

- Pequenas equipes de trabalho são organizadas de modo aumentar a comunicação, minimizar a supervisão e maximizar o compartilhamento de conhecimento.
- Processo adaptavel tanto a alterações técnicas quanto de negócios para garantir o melhor produto possível.
- O processo produz constantemente incrementos de software que pode ser inspecionados, ajustados, testados, ducumentados e expandidos.
- O trabalho de desenvolvimento e o pessoal que o realiza é divido em participações de baixa dependência, ou em pacotes.
- Os teste e documentação são realizados a medida que o produto é desenvolvido.
- Pode se declarar o produto como pronto a qualquer momento sempre que necessário.

Ainda segundo Pressman (2010, p.69) os príncipios são utilizados para guiar as atividades de desenvolvimento dentro de um processo que incorpora as atividades dentro de uma estrutura: requisitos, análise, projeto, evolução e entrega. Em cada atividade da estrutura, as tarefas de trabalho ocorrem dentro de um padrão de processo chamado Sprint.

A quantidade de Sprints para cada atividade varia de acordo com a complexidade e o tamanho do produto. O Projeto pode ser constantemente modificado em tempo real pela equipe Scrum. (PRESMMAN, 2010, p.69).

Conforme afirma Pressman (2010, p.69), o Scrum enfatiza o uso de padrões de processo de sofware para projetos onde os prazos são pequenos, os requisitos mutantes e existe uma criticidade de negócio. Cada um desses padrões de processo de software define um conjunto de atividades de desenvolvimento.

Pressman (2010, p.70) define o Sprint como atividades de trabalhos que são necessárias para atender um requisito e que precisa ser realizada em um determinado tempo (tipicamente trinta dias). Durante o Sprint não pode ser inseridas modificações. Assim é possível trabalhar com um ambiente estável de curto prazo. As prioridades dos requisitos são inseridas em uma lista de pendências. Nela os itens podem serem introduzidos a qualque momento e o gerente de produto avalia a pendência e atualiza a lista quando necessário.

Ainda segundo Pressman (2010, p. 70), as reuniões do Scrum são são feitas diariamente e devem ser de curta duração, normalmente quinze minutos. Nelas, três perguntas são respondidas por todos os membros da equipe:

1. O que você fez desde a última reunião de equipe?
2. Que obstáculos você está encontrando?
3. O que você planeja realizar até a próxima reunião de equipe?

O Scrum Master é o líder da equipe e por isso gerencia a reunião avaliando as respostas de cada pessoa. Essas reuniões são muito importantes pois ajudam a equipe a descobrir problemas potenciais o mais cedo possível. Além disso, através delas é possível obter a socialização do conhecimento, promovendo assim uma estrutura de equipe auto organizada. (PRESSMAN, 2010, p. 70)

No Scrum existe as Demos (demonstrações de funcionalidade implementadas) para mostrar as funcionalidades para avaliação do cliente. A demo talvez não contenha toda a funcionalidade planejada, mas, em vez disso, as funções que podem ser entregues dentro do intervalo de prazo previsto. (PRESSMAN, 2010, p.70).

A Figura 6 mostra o fluxo de processo do Scrum. Note que existem dois *bakclogs* (lista de pendências), o do produto (software a ser desenvolvido) e do Sprint. O Sprint tem duração média de 30 dias, neste período ocorrem reuniões diárias, no final do Sprint ocorre a Demo do trabalho do Sprint.

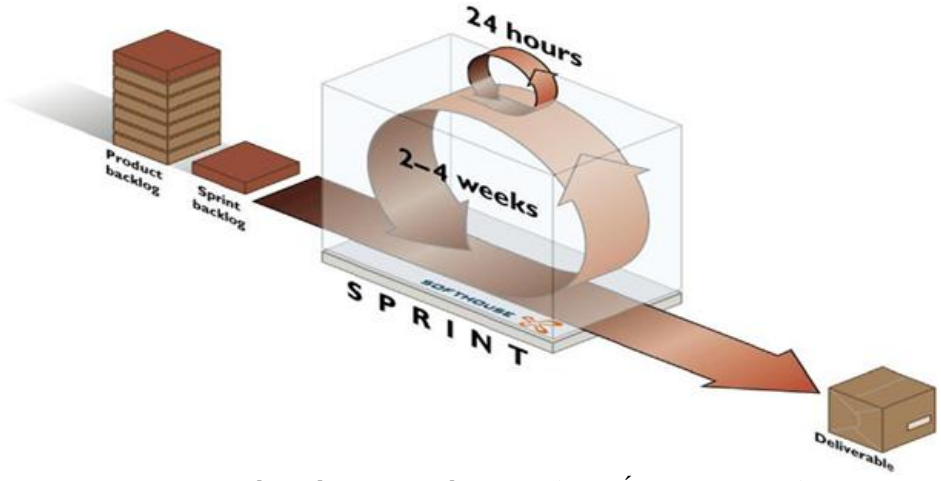

Figura 6: Fluxo de processo do Scrum (ARAÚJO, 2009, p. 88)

Segundo Araújo (2009, p. 42), a equipe Scrum é dividida da seguinte forma:

- Product Owner*:* ele é o responsável por definir os requisitos do produto e priorizar os mesmos, decidir a data de entrega e o que cada entrega deve abordar. Além disso ele pode alterar os requisitos e as prioridades de cada Sprint, além de aceitar ou não o Sprint.
- Scrum Master: ele é responsável por garantir a produção da equipe, facilitar o processo, eliminar os obstáculos e garantir que o processo seja seguido.
- Scrum Team: são as pessoas que irão desenvolver as tarefas do Sprint e ao final realizar a Demo do produto finalizado.

O Scrumé bem simples de ser utilizado, além disso, ele se encaixa muito bem em projetos onde as mudanças são constantes, e os problemas são difíceis de serem mensurados. Outro ponto positivo é que o Scrum preza muito a comunicação, facilitando assim estimar prazos, definir funções e criar soluções. Por ser simples, ele é indicado para pequenas equipes e pode ser utilizado não apenas para desenvolvimento de software.

As tarefas no Scrum podem ser representadas através de um quadro como mostrado na Figura 7. Nele é possível visualizar o andamento de cada item do Sprint. Separando por situações destes itens, por exemplo: item do *backlog*, a fazer, fazendo e feito.

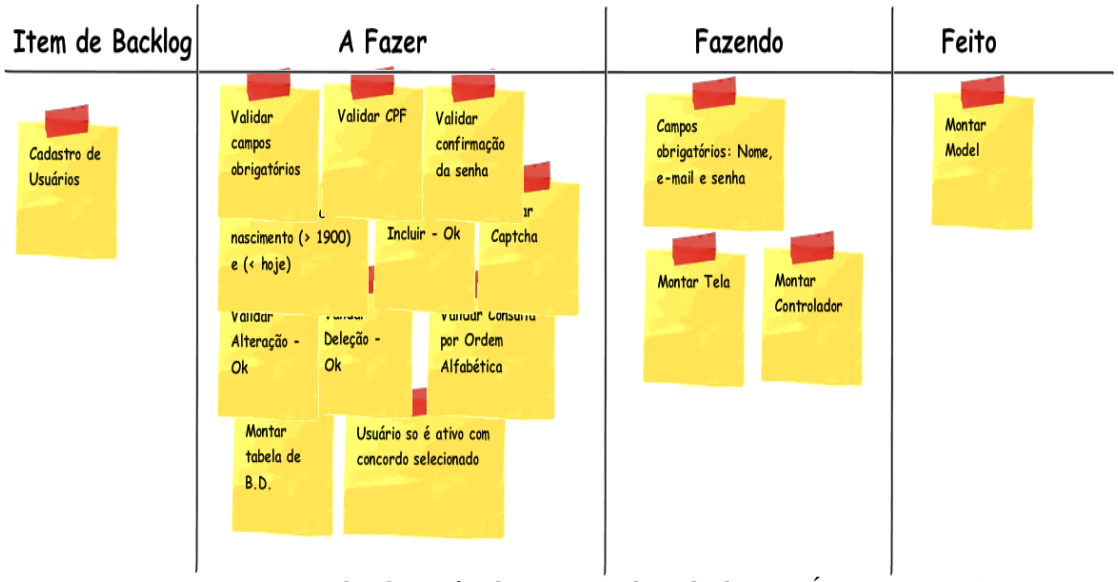

Figura 7: Quadro de tarefas do Scrum (Adaptado de ARAÚJ0, 2009, p. 51)

O acompanhamento de cada Sprint no Scrum, normalmente é feito com um gráfico chamado de Sprint Burndown, como mostrado na Figura 8. Todos os pontos (ou horas) dos itens do Sprint Backlog (itens a fazer) são somados, sendo este o ponto de partida do Sprint. No final do Sprint, a quantidade de pontos será zero, pois todo os itens do Sprint Backlog devem estar prontos. Assim é montada a curva do previsto, mostrada como tracejada na figura. À medida que o Sprint vai sendo realizado, os pontos dos itens realizados vão sendo abatidos do valor original, montando assim a curva do realizado, mostrada em verde na figura.

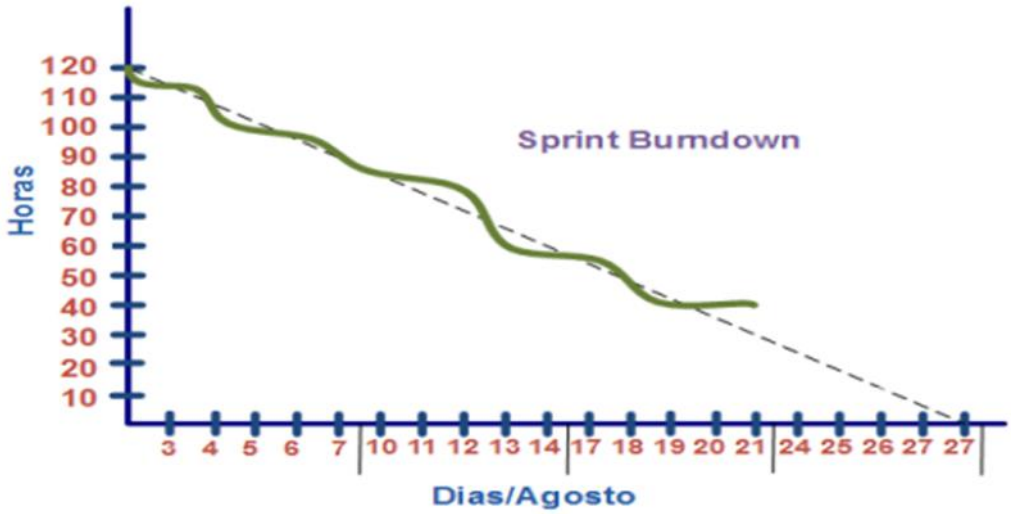

Figura 8: Gráfico de acompanhamento do Sprint (Sprint Burnsdown) (ARAÚJ0, 2009, p. 48)

3 Avaliação da Aplicabilidade

Em uma empresa de tecnologia de informação localizada em Araxá, MG, com clientes de vários portes,onde existia um problema importante no desenvolvimento de aplicações, no final do primeiro semestre de 2013. Praticamente todos os projetos atrasavam e/ou não atendiam a necessidade do cliente, causando insatisfação do cliente, da gestão da empresa e dos desenvolvedores. A empresa utilizava um modelo de desenvolvimento parecido com o modelo Cascata, ou seja, ela tentava levantar todos os requisitos, depois fazia uma análise parcial e partia para o desenvolvimento. Diante do problema, uma das possíveis causas identificadas foi a ausência de metodologia de desenvolvimento.

Como a equipe era pequena (seis pessoas), os projetos curtos (de dois a três meses de duração), geralmente os clientes não tinham os requisitos muito claros e a empresa precisava de uma solução rápida, o Scrum foi apontado como uma proposta de solução para a causa identificada, devido ao conhecimento do método que um analista da equipe já tinha. As pessoas acharam que o aprendizado e a adequação do RUP seria mais complexo e lento para a empresa, além da impressão que as pessoas tinham de que o RUP demandaria muita documentação. No caso XP, além do aprendizado, a falta de incorporação da gestão de projetos no método, fez com que a empresa optasse por não correr o risco de testá-lo. A correção do método Cascata não foi considerada, pois a empresa já tinha uma visão que a forma sequencial do Cascata não seria adequada para a agilidade demandada pelos clientes. Diante disto, optou-se pela implantação do Scrum, como metodologia de desenvolvimento e gestão dos projetos da empresa.

Para implantação do Scrum foram utilizadas as etapas mostradas na Figura 9. No início, um analista mais experiente e que já havia trabalhado com o Scrum antes, fez um rápido *workshop* de treinamento para a equipe da empresa, a ideia foi apenas apresentar o método para que o aprendizado visse realmente no dia a dia do projeto piloto. Em seguida, um dos projetos da empresa foi escolhido como piloto e o Scrum aplicado. Como o resultado foi satisfatório para o cliente, para a equipe e para a direção da empresa, o Scrum

passou a ser aplicado para todos os demais projetos da empresa.

.

Figura 9: Implantação do Scrum na empresa.

Um dos principais riscos enfrentados durante a implantação, foi o risco de perda do direcionamento devido à saída do analista mais experiente que conhecia o Scrum, no final do projeto piloto. Este risco foi mitigado devido à facilidade de utilização do Scrum, a equipe aprendeu rapidamente e começou a utilizar de forma natural, tornando o impacto da saída bem menor do que o esperado.

Após o início da utilização do Scrum (após o projeto piloto), os resultados dez projetos foram avaliados quanto ao cumprimento de prazos e a aceitação da entrega final do escopo sem mudanças adicionais. Estes dez projetos, foram comparados com os dez últimos projetos anteriores ao projeto piloto de implantação do Scrum. Os resultados desta avaliação estão mostrados na Figura 10, note que a aceitação do escopo sem mudanças adicionais (após a entrega) melhorou de 40% para 70% e o cumprimento de prazo melhorou de 10% para 60% dos projetos.

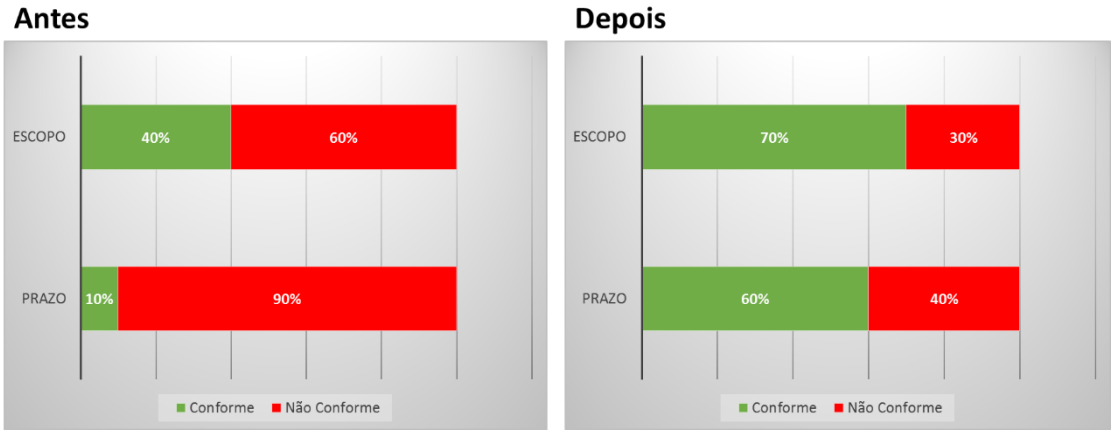

Figura 10: Resultado da aplicação do Scrum na empresa.

Os resultados mostram claramente que todos passaram a ter um maior controle sobre os projetos. A equipe concluiu que a melhora na comunicação interna e com os clientes foi um dos fatores mais importantes para o resultado. Outro fator que contribui para a melhoria foi a facilidade de visualizar o andamento do projeto, permitindo assim traçar estratégias e evitar o atraso na entrega para o cliente.Mesmo com a saída da pessoa mais experiente, a empresa percebeu que a metodologia foi muito útil, pois trouxe organização e disciplina para toda a equipe deixando-a mais integrada e responsável pelos

projetos.

O único ponto que ainda estava gerando algum desconforto nos clientes e nos gestores da empresa, era o fato de que no início do projeto existe uma falta de previsibilidade de quando o projeto será concluído e quanto de esforço é necessário para isto. Alguns clientes conseguem se adaptar a isto, por outro lado, outros clientes não ficam tão à vontade com isto. A empresa estava pensando em tentar calcular todos os pontos de todos os itens do Product Backlog. Porém, avaliando o resultado desta ação na literatura, observa-se um bom grau de ineficiência desta ação, pois no início do projeto os requisitos não estão tão claros assim e tentar explicitá-los antes do projeto começar é praticamente voltar ao modelo anterior (Cascata). Diante disto, a empresa pretende continuar trabalhando com o Scrum da forma que está. A única melhoria prevista para os próximos projetos é a implantação do gráfico de acompanhamento Sprint Burndown, como apresentado na Figura 8, para facilitar ainda mais o acompanhamento de cada Sprint pela equipe, pela gestão e pelos clientes.

4 Conclusão

A empresa não teve tempo de testar o RUP, o XP e nem de corrigir a implementação do método cascata que era utilizado. A empresa implantou o Scrum como metodologia de gestão e desenvolvimento. Com o Scrum a aceitação da entrega do escopo do projeto sem mudanças adicionais melhorou de 40% para 70% e o cumprimento de prazo melhorou de 10% para 60%.

Os resultados mostraram que o Scrum foi satisfatório para os objetivos da empresa: entregar no prazo, com menos retrabalho, com mais controle e mais satisfação do cliente e da equipe. Diante disto, a implantação do Scrum se mostrou satisfatória para a empresa em questão.Apesar dos resultados, a falta de previsibilidade de quando o projeto vai terminar e de quanto esforço é necessário para isto, causa desconforto em alguns clientes.

Como trabalho futuro, a empresa poderia investir melhorias no uso do Scrum ou separar alguns projetos e testar o uso do XP ou do RUP para pequenos projetos, para saber se estes métodos poderiam melhorar ainda mais os resultados.

5. Referências

ARAÚJO, Leandro. **Estudo comparativo da compatibilidade entre as melhores práticas do PMI e Scrum.** 2009. Faculdade de Informática e Administração Paulista. Disponível em: <http://www.gestaopm.com.br/documentos/PMI_Scrum.pdf>. Acesso em 25 fev 2014.

MOREIRA, Márcio. ***Metodologia de Desenvolvimento de Software: Apresentação e Introdução***. 2010. Pitágoras. Disponível em: <http://www.teraits.com/pitagoras/marcio/rup/Marcio_rup_0_Apresentacao_Introducao.pptx>. Acesso em 24 fev 2014.

MOURA, Ellen; MOREIRA, Márcio. Metodologias de desenvolvimento de software. 2012. Pitágoras. Disponível em:<

http://www.teraits.com/pitagoras/marcio/ori_p/20120428_daw_1_EllenMoura_MetodologiasDesenvolvimentoSoftware.pdf>. Acesso em 11 fev 2014.

PRESSMAN, Roger. ***Engenharia de Software***. 6ª Ed. São Paulo: AMGH Editora LTDA, 2010. 709p.

PRESSMAN, Roger. ***Engenharia de Software***. 7ª Ed. São Paulo: AMGH Editora LTDA, 2011. 720p.

SIMÕES, Nunes. ***Grandes oportunidades para a indústria brasileira de software***. Canaltech. 2013. Disponível em: <http://corporate.canaltech.com.br/noticia/mercado/Grandes-oportunidades-para-a-industria-brasileira-de-software/>. Acesso em: 09 fev 2014.

SOMMERVILE, Ian. Engenharia de Software. 8ª Ed. São Paulo: Pearson Addison Wesley, 2007. 551p.

Standish Group. ***Chaos Manifesto 2013. Thing Big, Act Small***. Stadinsh Group. 2013. Disponível em: <http://versionone.com/assets/img/files/ChaosManifesto2013.pdf>. Acesso em 20 fev. 2014.