CENTRO PAULA SOUZA
GOVERNO DO ESTADO DE SÃO PAULO

Faculdade de Tecnologia de Americana

Curso de Bacharelado em Análise de Sistemas e Tecnologia da Informação

METODOLOGIA ÁGIL SCRUM:

Aplicação em uma empresa de desenvolvimento de sistemas integrados

SAMUEL VIEIRA DOS SANTOS **RA: 0912232**

AMERICANA/SP

2013

CENTRO PAULA SOUZA GOVERNO DO ESTADO DE SÃO PAULO

Faculdade de Tecnologia de Americana

Curso de Bacharelado em Análise de Sistemas e Tecnologia da Informação

METODOLOGIA ÁGIL SCRUM:

Aplicação em uma empresa de desenvolvimento de sistemas integrados

SAMUEL VIEIRA DOS SANTOS RA: 0912232

samuka.svs@gmail.com

Trabalho de Conclusão de Curso para obtenção do título de Bacharelado em Análise de Sistemas e Tecnologia da Informação da FATEC – Americana/SP.

Orientador: Me. Carlos Henrique Rodrigues Sarro

AMERICANA/SP

2013

BANCA EXAMINADORA

Aprovado em:

_____ ___/___/_____

Orientador (a): Mestre Carlos Henrique Rodrigues Sarro

FATEC – Americana/SP

_____ ___/___/_____

Professor (a):

FATEC – Americana/SP

_____ ___/___/_____

Professor (a):

FATEC – Americana/SP

À

Minha família que sempre me apoiou nos momentos difíceis.

AGRADECIMENTOS

Agradeço primeiramente a Deus por ter me dado força e coragem para enfrentar e superar os momentos difíceis até os dias de hoje.

Ao meu orientador Carlos Henrique Rodrigues Sarro, que sem duvida nenhuma teve um papel fundamental no desenvolvimento desta monografia.

Aos meus pais Guilherme e Marli que sempre me apoiaram e invitaram a nunca desistir dos sonhos e sempre me deram o suporte necessário em todos os momentos.

A minha irmã Gisele que sempre esteve presente me dando suporte durante o desenvolvimento deste trabalho e que com certeza tem grande parcela dos méritos desta monografia.

Ao meu grande amigo Rafael que fiz durante a faculdade, sem o qual com certeza não seria possível chegar ao fim deste curso.

A todos os professores da FATEC que de alguma forma contribuíram durante a faculdade e o desenvolvimento deste presente trabalho

O mundo não está ameaçado pelas pessoas más, e sim por aquelas que permitem a maldade.

(Albert Einstein)

RESUMO

O processo de desenvolvimento de projetos de *software* está em constante evolução, buscando aumentar a produtividade, se aproximar mais do cliente e entregar um produto final com mais valor. A partir destas ideias de evolução e das metodologias tradicionais existentes, que trazem uma documentação excessiva e não absorvem bem mudanças durante o projeto, surgiu então o Manifesto Ágil que permitiu o aparecimento de algumas metodologias ágeis de desenvolvimento de *software.* Dentre estas metodologias ágeis, o Scrum se apresenta como uma das mais populares e eficientes. Esta monografia tem como objetivo apresentar o que são as metodologias ágeis, assim como mostrar com mais detalhes a Metodologia Ágil Scrum, no que consiste, quais são suas regras, características e os requisitos para sua implantação. Para isto será apresentado um estudo de caso da aplicação do Scrum em uma empresa de desenvolvimento de sistemas integrados.

Palavras-chave: Metodologia Ágil; Scrum; Produtividade.

ABSTRACT

The process of developing software projects is constantly evolving, seeking to increase productivity, get closer to the customer and deliver a final product with more value. From these ideas of evolution and traditional methodologies exist that bring a excessive documentation and do not allow changes during the project, then came the Agile Manifesto which allowed the emergence of some agile software development. Among these agile methodologies, Scrum is presented as one of the most popular and efficient. This monograph aims to present what they are agile methodologies, as well as to show in more detail the Agile Scrum, what is, what are your rules, characteristics and requirements for your deployment. To this will be presented a case study of the implementation of Scrum in an enterprise system ERP development.

Keywords: Agile methodology, Scrum, Productivity.

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Muitas empresas de desenvolvimento de *software* ainda utilizam métodos tradicionais, como Cascata para o processo de desenvolvimento de *software*. Estas metodologias tradicionais são divididas em fases e após a conclusão de uma fase então é iniciada a próxima e geralmente o processo inverso não ocorre, pois estas metodologias não preveem mudanças dos requisitos durante o projeto. Teoricamente estas fases estão corretas, todo o fluxo do processo de desenvolvimento de *software* deveriam seguir as fases consecutivamente para que fosse produzido um produto final da forma que foi planejado e dentro do prazo. Mas, na prática muitas empresas perceberam que as etapas não fluíam como o esperado, tendo que muitas vezes voltar em etapas anteriores, que causava muitos problemas, prejuízos e atrasos, pois as metodologias tradicionais não preveem mudanças ao decorrer do projeto.

Aliado a esse problema também se percebeu que as metodologias tradicionais exigiam uma documentação muito detalhada, extensa e muitas vezes desnecessária. Esta documentação exagerada além de causar atraso no projeto, também dificulta manutenções futuras, impossibilitando análises simples de partes do sistema. Diante desse contexto, surgiram novas ideias para mudar e melhorar o processo de desenvolvimento de *software*.

Dentre estas novas ideias, surgiu o Manifesto Ágil que apresentava valores, princípios e padrões que tinham o objetivo de simplificar o processo de desenvolvimento eliminando burocracias desnecessárias; possibilitar o cliente estar mais próximo do desenvolvimento de *software* e criar uma arquitetura de desenvolvimento de software que permitisse mudanças frequentes nas funcionalidades do *software.* Todas estas características sugeridas pelo Manifesto Ágil tinha o objetivo final de ter um produto final com mais valor e consequentemente também aumentar a produtividade do desenvolvimento de software.

Inspirados no Manifesto Ágil surgiram às metodologias ágeis que seguiu os valores e princípios do Manifesto, porém cada uma destas tinha suas peculiaridades. Com o tempo algumas destas metodologias se destacaram mais e ganharam grande força no mercado de desenvolvimento de *software*. O Scrum foi uma destas

metodologias ágeis que mais se destacaram hoje é uma das mais usadas nas empresas de desenvolvimento de *software.*

A metodologia ágil Scrum segue todos os conceitos básicos do Manifesto Ágil, mas assim como todas as metodologias ágeis também tem suas características únicas. Serão apresentados detalhadamente todas as regras e características que envolvem a aplicação do Scrum em empresas de desenvolvimento de *software.* Também serão abordados todos os papeis e responsabilidades de cada pessoa envolvidas no projeto de software desde os desenvolvedores até o próprio cliente que tem um papel fundamental dentro da metodologia ágil Scrum.

Com o objetivo de mostrar na pratica o processo de desenvolvimento de *software* utilizando o Scrum será apresentado um estudo de caso mostrando a aplicação do Scrum em um projeto de melhoria do sistema em uma empresa de desenvolvimento de software ERP. Para isto foi utilizado a ferramenta BraZip Scrum que tem todas as funcionalidades necessárias para o processo de desenvolvimento de *software*. Neste estudo de caso será mostrado um passo a passo de como se deve proceder em um projeto utilizando o Scrum.

No fim da monografia é feito uma análise dos resultados obtidos a partir da utilização da Metodologia Ágil Scrum, mostrando quais são as principais vantagens e desvantagens da adoção desta metodologia. Para concluir a monografia é apresentado no último capítulo as considerações finais fazendo uma análise da metodologia ágil Scrum e sua aplicação em uma empresa de desenvolvimento de sistemas integrados.

2 SURGIMENTO DAS METODOLOGIAS ÁGEIS

Antes de aprofundar na Metodologia Ágil é necessário entender como surgiu tal metodologia e que valores inspiraram sua existência. Desta forma, será apresentado o Manifesto Ágil mostrando o contexto histórico de seu surgimento e detalhando seus valores e princípios. Também são analisadas algumas das metodologias ágeis mais importantes no mercado, evidenciando o Scrum.

2.1 História do Manifesto Ágil

O surgimento da metodologia Scrum, assim como outras metodologias ágeis, foi inspirado a partir da criação do Manifesto Ágil que tinha como principal filosofia agilizar o processo de desenvolvimento de *software*.

O Manifesto Ágil surgiu em fevereiro de 2001, quando um grupo de 17 pessoas que eram grandes pensadores sobre o processo de gerenciamento e desenvolvimento de *software,* se encontraram em uma estação de esqui chamada *The Lodge at Snowbird*, em Wasatch, nas montanhas Utah, para discutir melhorias no desempenho de seus projetos. Os 17 participantes trabalhavam com desenvolvimento de *software* em empresas diferentes, de modo que não existia uma única visão, entretanto o objetivo principal do grupo era o mesmo, aumentar a agilidade, diminuindo a pesada quantidade de documentação, porém sem perder qualidade no processo de desenvolvimento.

A partir deste encontro, devido ao senso comum sobre alguns valores e princípios da nova metodologia, foi criada a Aliança Ágil e estabelecido o Manifesto Ágil.

2.2 Valores do Manifesto Ágil

Segundo o Manifesto Ágil (2001), foi definido quatro valores:

- Indivíduos e interações são mais importantes do que processos e ferramentas

O primeiro valor se refere ao fato de que as pessoas envolvidas no projeto, sejam elas da equipe de desenvolvimento ou o próprio cliente, devem ser consideradas mais importantes do que as máquinas e programas. Ter uma boa interação entre os membros do projeto é um fator chave para que o processo de desenvolvimento de *software* alcance seus objetivos.

- *Software* funcionando é mais importante do que documentação extensa

O Manifesto Ágil coloca o funcionamento do *software* em primeiro plano, na qual mudanças rápidas na funcionalidade do *software* podem ocorrer durante o desenvolvimento do projeto, ficando para segundo plano os processos burocráticos, como a documentação.

Vale salientar que, o Manifesto Ágil não ignora a documentação, apenas não o trata como prioridade no processo de desenvolvimento de *software*. A documentação tem um valor importante no processo, porém só tem utilidade quando contribui para o funcionamento do *software*, pois na prática o cliente se interessa apenas em resultados e o resultado é o funcionamento do *software*. Deste modo, a documentação deve existir, entretanto de forma simplificada e apenas com o objetivo de auxiliar no funcionamento do *software*.

- O relacionamento com o cliente é mais importante do que a negociação do contrato

O Manifesto Ágil é favor do contrato, mas não o vê como prioridade, pois não há uma preocupação com prazos, custos e condições, tampouco os contratos não devem ser usados pela equipe de desenvolvimento como argumento para se proteger de reclamações do cliente, uma vez que o objetivo do projeto é deixar o cliente satisfeito.

O mais importante no desenvolvimento de um projeto de *software* é garantir que o produto final atenda as expectativas e necessidades do cliente. A equipe de desenvolvimento e o cliente devem ter o mesmo objetivo que é agregar valor ao *software*, para tanto é prezado uma interação constante do cliente com a equipe de desenvolvimento, cabendo ao cliente solicitar alterações no projeto a qualquer momento, em prol de agregar valores ao produto final, satisfazendo suas expectativas.

- Responder às mudanças é mais importante do que seguir o planejamento

Mudanças em um projeto de *software* são praticamente inevitáveis, então o mais importante é providenciar o mais rápido possível soluções para estas mudanças. Claro que no início do projeto deva existir um planejamento de todo o processo de desenvolvimento do *software*, entretanto ele não deve ser levado adiante se houver mudanças, sendo assim o plano deve ser alterado por mais que essa alteração cause atraso no desenvolvimento, afinal o mais importante é o projeto trazer valor ao negócio do usuário final e não seguir um planejamento que não irá atender as necessidades do cliente.

2.3 Princípios do Desenvolvimento Ágil

Para fundamentar a filosofia do Manifesto Ágil foi instituído 12 princípios:

“Nossa maior prioridade é satisfazer o cliente, através da entrega contínua e adiantada de software com valor agregado” (ÁGIL, 2001).

No Desenvolvimento Ágil não existe uma única entrega do produto que irá conter todas as funcionalidades, na prática o que acontece são várias entregas contínuas do *software*. Entretanto, a cada entrega o *software* deve agregar valor ao cliente, isto é, o *software* deve atender as necessidades do negócio do cliente mesmo que seja parcialmente.

Também é muito importante que essas entregas contínuas sejam feitas de forma adiantada para que o cliente não seja surpreendido negativamente, evitando prejuízos ao cliente.

Desta forma, o cliente ganhará confiança no desenvolvimento de *software* tendo a convicção de que em cada entrega do *software* irá agregar valor ao seu negócio.

“Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças, para que o cliente possa tirar vantagens competitivas“ (ÁGIL, 2001).

Este princípio menciona que no processo de Desenvolvimento Ágil é necessário estar preparado para mudanças simples e complexas mesmo que estas mudanças tenham um impacto grande no planejamento do projeto. O cliente não pode ser obrigado a fazer um planejamento tomando decisões antes de o projeto tomar uma forma, ou até mesmo ser privado de voltar atrás para fazer alterações e melhorias.

No desenvolvimento essas mudanças devem ser vistas como vantagens que irão melhorar o produto final e atenderá as necessidades do cliente com mais eficácia, dando a ele mais vantagens competitivas no mercado.

“Entregar *software* funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos.” (ÁGIL, 2001).

Um planejamento a longo prazo dificulta a equipe de desenvolvimento de *software*, assim como o cliente que tem que esperar um longo tempo para verificar o produto e identificar possíveis alterações que já poderiam ter sido verificados no decorrer do desenvolvimento. Então, com entregas em menor escala de tempo será evitado implementações desnecessárias, adiantando a identificação de erros. Além de ajudar a equipe de desenvolvimento de *software* ver com mais clareza e facilidade as implementações a serem realizadas.

Outro fator muito importante é que com entregas mais frequentes o cliente fica mais envolvido com o processo de desenvolvimento de *software*.

“Pessoas relacionadas à negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projeto” (ÁGIL, 2001).

A equipe de desenvolvimento de *software* tem todo o domínio técnico para a implementação do projeto, por outro lado as pessoas que entendem do negócio, que estão representando o cliente, detém todo o conhecimento de como o projeto deve ser desenvolvido para atender as necessidades do cliente. Então, é de extrema importância que ambos trabalhem constantemente em conjunto para que o projeto

chegue no seu objetivo final que é agregar valor ao negócio do cliente da melhor maneira possível.

“Construir projetos ao redor de indivíduos motivados. Dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho.” (ÁGIL, 2001).

As pessoas envolvidas no desenvolvimento devem estar motivadas e para isso os desenvolvedores recebem responsabilidades e devem ter a confiança para tomar decisões de acordo com sua função na equipe. Desta forma, os integrantes passam a se sentir importantes no processo de desenvolvimento e realmente passam a se sentir parte do projeto.

Para isso é necessário que toda a equipe tenha um suporte e um ambiente adequado que permita que cada pessoa envolvida consiga resolver os problemas constantes no desenvolvimento de *software*.

“O Método mais eficiente e eficaz de transmitir informações para, e por dentro de um time de desenvolvimento, é através de uma conversa cara a cara” (ÁGIL, 2001).

Em um desenvolvimento baseado no Manifesto Ágil, um dos pontos mais relevantes é a importância em existir reuniões com o cliente, inclusive estas reuniões devem ocorrer de forma frequente. Nestas reuniões, o cliente e a equipe de desenvolvimento tem a possibilidade de expressar através de linguagem verbal e não verbal o que querem realmente dizer.

Deste modo, a equipe de desenvolvimento saberá exatamente o que o cliente precisa e providenciará a melhor solução para atender as necessidades do mesmo.

“Software funcional é a medida primária de progresso” (ÁGIL, 2001).

A cada entrega do *software*, este precisa atender as necessidades do negócio do cliente e agregar valor ao mesmo. Independente se o *software* esteja organizado, realizando vários processos e tenha sido entregue com muita agilidade, ele somente será considerado um progresso quando for validado pelo cliente.

“Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter indefinidamente, passos constantes” (ÁGIL, 2001).

O processo de desenvolvimento de *software* deve ter uma constante produção em um mesmo ritmo em um longo período de tempo. Desta forma, é importante ter muitas entregas do *software* em pequenas escalas de tempo, permitindo assim que ocorra vários testes e validações, dando a possibilidade de que problemas sejam corrigidos e mudanças possam ser identificadas e realizadas de acordo com a necessidade do cliente.

Essas medidas devem ser seguidas no Desenvolvimento Ágil para evitar que no início do desenvolvimento a intensidade seja baixa e que no final seja necessário aumentar a intensidade para entregar na data combinada, senão a empresa terá prejuízos com horas extras, pois o *software* realizado muito rápido é provável que no final haja grande chance de ser entregue com problemas.

“Contínua atenção à excelência técnica e bom design, aumenta a agilidade” (ÁGIL, 2001).

O Desenvolvimento Ágil deve ser pautado por várias práticas técnicas e devem ter um design muito eficiente e eficaz, sendo simples para que não seja perdido tempo de forma desnecessária.

Estes fatores simples, porém muito eficientes permitirão um aumento considerável de agilidade no processo de desenvolvimento de *software*.

“Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito” (ÁGIL, 2001).

Os projetos desenvolvidos de forma ágil devem ser o mais simples possível, o que vai facilitar a implementação, a manutenção e até mesmo o entendimento por parte do cliente.

Para que isso se torne uma realidade, as funcionalidades que não tem muita importância na agregação de valor para o negócio do cliente devem ser evitadas. O mesmo deve ocorrer no processo de desenvolvimento do *software*, devem ser evitado utilizar ferramentas complexas e realizar implementações que não sejam de suma importância para as funcionalidades do sistema.

“As melhores arquiteturas, requisitos e designs emergem de times auto-organizáveis” (ÁGIL, 2001).

Para que o projeto alcance seu objetivo final com eficiência e agilidade, a equipe de desenvolvimento precisa ser auto-organizável. Muitos problemas aparecerão no processo de desenvolvimento, porém os integrantes não podem ficar presos em suas funcionalidades básicas. Quando as pessoas envolvidas no desenvolvimento estão motivadas, elas buscam resolver os problemas mesmo que necessitem sair de suas funções e cargo para isso.

Um integrante da equipe não conseguirá mostrar e explorar todo o seu potencial se o mesmo ficar preso em seu cargo, a partir do momento que todos exercem várias funções, a equipe consegue explorar mais seu potencial e dar mais qualidade no processo de desenvolvimento de *software*.

“Em intervalos regulares, o time reflete em como ficar mais efetivo, então, se ajustam e otimizam seu comportamento de acordo.” (ÁGIL, 2001).

Frequentemente, a equipe precisa se reunir e refletir sobre os processos e trabalho em equipe, evidenciando pontos positivos que estão melhorando o desempenho da equipe como também, indicando pontos de melhorias que precisam ser tratados para que o processo de desenvolvimento e o comportamento da equipe melhorem.

2.4 Metodologias Ágeis

A partir dos valores e princípios do Manifesto Ágil, surgiram algumas Metodologias Ágeis para gerenciamento de desenvolvimento de *software*. Dentre essas metodologias, quatro tiveram um destaque maior no mercado.

- *Dynamic Systems Development Method* (DSDM)[1]

O DSDM é uma metodologia de desenvolvimento de sistema dinâmico e tem como característica principal ser baseada em uma metodologia de desenvolvimento interativo e incremental.

Basicamente, a estrutura do desenvolvimento de *software* baseado na Metodologia Ágil DSDM consiste em três fases: Pré-projeto, Ciclo de vida e Pós-

[1] Metodologia de Desenvolvimento de Sistema Dinâmico (DSDM)

projeto. O ciclo de vida é ainda subdividido e mais cinco estágios: análise de viabilidade, análise de negócio, iteração do modelo funcional, iteração de elaboração e construção e implementação.

O DSDM tem como objetivo principal entregar ao usuário o produto final de acordo com o que foi solicitado e de acordo com o tempo estimado, para isso conta com ajustes de requisitos ao longo do tempo do processo de desenvolvimento (STAPLETON, 2003).

- *Feature Driven Development* (FDP)[2]

Seu desenvolvimento de *software* é guiado por funcionalidades e apesar de ser uma Metodologia Ágil ainda possui algumas características da metodologia tradicional. É uma metodologia intermediária se comparada às metodologias tradicionais e ágeis.

O gerenciamento de desenvolvimento de *software* baseada no FDP é classificado em cinco processos: Desenvolvimento de um modelo abrangente, Construção de uma lista de funcionalidades, Planejamento com base nas funcionalidades, Detalhamento das funcionalidades e Construção das funcionalidades (PURY, 2004).

- Programação Extrema (XP)

Conhecido como XP, ela normalmente é utilizada em projetos onde os requisitos estarão em constantes mudanças. A XP é uma Metodologia Ágil que tem como objetivo o controle da qualidade do *software* a ser entregue ao cliente, mesmo que demore mais para ser implementado, pois acredita-se que as perdas da qualidade para ganho de produtividade não são compensadoras.

Para aplicação da XP no gerenciamento de desenvolvimento de *software* são considerados cinco valores: Comunicação, Simplicidade, *Feedback*, Coragem e Respeito. Baseados nesses valores são considerados mais cinco princípios de suma

[2] Desenvolvimento guiado por Funcionalidades (FDP)

importância na XP: *Feedback* rápido, Presumir simplicidade, Mudanças incrementais, Abraçar mudanças e Trabalho de alta qualidade (BECK, 2004).

- Scrum

Baltzan e Phillips (2012) dizem que “[...] a Metodologia Scrum, utiliza pequenas equipes para a produção de pequenas partes de *software* a serem entregues, utilizando *sprints* ou intervalo de 30 dias para alcançar uma meta determinada [...]”. Diante disso, neste TCC, o estudo de caso de projeto de melhorias de um sistema integrado (ERP) será baseado na Metodologia Ágil Scrum. No próximo capítulo será mais bem explicado as características da Metodologia Ágil Scrum.

3 METODOLOGIA ÁGIL SCRUM

A Metodologia Ágil Scrum será analisada de forma bem detalhada e profunda, mostrando suas principais características e quais são as regras que devem ser seguidas para extrair o máximo dos benefícios desta metodologia. Também serão apresentados todos os papéis e responsabilidades de cada pessoa envolvida no projeto de *software*. E por fim serão apresentadas algumas das ferramentas mais usadas no mercado, destacando a ferramenta Brazip Scrum que será utilizada no estudo de caso no capítulo 4.

3.1 Conceito básico da Metodologia Ágil Scrum

Scrum é um *framework*[3] que define processos de desenvolvimento interativo para gerenciar de forma simples projetos que geralmente são complexos. Tem o foco voltado para o negócio especificamente, desta forma as funcionalidades mais importantes são entregues primeiro, tendo como objetivo principal agilizar o gerenciamento e o desenvolvimento do projeto, deixando em segundo plano as regras e documentações, entretanto mantendo a qualidade exigida pelo cliente.

O Scrum permite alterações complexas no projeto a qualquer momento de forma eficiente, diferindo assim em mais um aspecto do método tradicional que não tem como planejamento voltar em etapas anteriores para fazer reajustes.

O projeto é dividido em blocos e a cada bloco finalizado o cliente faz a sua análise e indica os erros e reajustes que devem ser realizados para passar para a próxima etapa, deste modo o projeto evolui gradativamente tendo uma interação constante do cliente com o projeto, evitando assim problemas grandes no fim do projeto.

Diferentemente de outras metodologias, inclusive as ágeis, o Scrum divide todas as responsabilidades para a equipe de desenvolvimento, como por exemplo, a responsabilidade de estimar a quantidade de horas que será necessário para concluir um bloco do projeto (MARTINS, 2007).

[3] É um conjunto de conceitos usado para resolver um problema de um domínio específico.

3.2 História da Metodologia Ágil Scrum

O termo Scrum é oriundo do jogo *rugby*, onde quando ocorre um incidente no jogo, todo o time se reúne, recomeçando a partida e dando continuação ao jogo. Com base nisso, a Metodologia Ágil Scrum é baseada na ideia de sempre reunir os desenvolvedores para que o projeto continue rolando de forma eficiente.

> “No rugby, o scrum é uma parte da equipe em que todos trabalham em conjunto para movimentar a bola pelo campo. Segundo essa metodologia, cada dia termina ou começa com uma reunião em pé diária para monitorar e controlar a atividade de desenvolvimento.” (BALTIZAN e PHILLIPS, 2012, p. 284).

Jeff Sutherland trabalhava na Easel Corporation em 1994, e devido à necessidade de seu time de *software* precisar de uma versão melhorada de rápido desenvolvimento de aplicação, buscava desenvolver uma nova ferramenta. Durante mais ou menos o mesmo período, Ken Schwaber procurava criar meios para desenvolver a produtividade de sua equipe, na empresa em que trabalhava, Advanced Development Methods, Inc. (ADM).

Já em 1995, Jeff Sutherland e Ken Schwaber trabalharam juntos e formalizaram um processo com estilo de gerenciamento voltado para a indústria de *software* e então, publicaram o primeiro artigo sobre Scrum. Apesar de ter sido criado para gerenciar projetos em empresas de fabricação de automóveis e produtos de consumo, o Scrum ganhou força no gerenciamento de processos de desenvolvimento de *software*, pois tem características muito compatíveis com os processos complexos da maioria dos projetos de *software*.

3.3 Regras e características da Metodologia Ágil Scrum

O Scrum possui regras e características que são baseadas no Manifesto Ágil, estas regras e características são basicamente definições de Papéis e responsabilidades, Artefatos e Reuniões. Para que o Scrum seja implantado em uma

equipe de desenvolvimento de *software* é necessário que todas essas regras sejam seguidas e respeitadas (ORTH; PRIKLADNICK, 2009).

3.3.1 Principais Papéis e Responsabilidades

Cada pessoa envolvida no projeto terá seu papel e suas responsabilidades na utilização do Scrum. Os três papéis principais são:

- ***Product Owner* (PO)[4]**

O PO é a pessoa que representa a empresa-cliente nas reuniões de planejamento do projeto, desenvolvimento e entrega dos blocos de *software*. Ele deve ter um conhecimento básico do processo de desenvolvimento para que possa compreender melhor o andamento do desenvolvimento do projeto.

> "[...] o dono do produto é normalmente o cliente ou alguém interessado no resultado no projeto. Ele tem como principais responsabilidades: definir as funcionalidades, ajustar funcionalidades e prioridades, aceitar ou rejeitar o resultado dos trabalhos, resolver dúvidas da equipe quanto ao entendimento dos requisitos." (ORTH; PRIKLADNICK, 2009, p. 152)

Nas reuniões de planejamento o PO tem a responsabilidade de indicar o que deverá ser realizado pela equipe de desenvolvimento. Nas reuniões de entrega do *software* o *Product Owner* é quem valida o *software* com o objetivo de verificar se o mesmo atende as expectativas da empresa-cliente.

O *Product Owner* tem como principal responsabilidade garantir que a empresa-cliente terá um produto final que agregue valor ao seu negócio, ou seja, ele tem que garantir que a empresa-cliente tenha o retorno esperado perante o investimento que foi feito para o desenvolvimento do projeto de *software*.

- **Equipe de Desenvolvimento (Time)**

[4] Dono do produto.

A equipe de desenvolvimento é composta por todas as pessoas que irão ajudar de alguma forma a desenvolver o *software*. Cada membro da equipe deve ter uma especialidade que geralmente são nas áreas de programação, teste, arquitetura ou análise. Na metodologia Scrum recomenda-se que todos os integrantes busquem além de suas especialidades, ajudar o time trabalhando em outras funções com o objetivo de entregar um *software* de valor.

Em consequência, é alcançado um dos grandes objetivos da metodologia Scrum que é o compartilhamento de conhecimento entre os integrantes da equipe. Cada pessoa da equipe deve saber o que os outros estão fazendo para que, se necessário, possam dar continuidade no desenvolvimento de outro integrante. Por este motivo é importante que toda a equipe tenha uma boa comunicação e assim, os processos de desenvolvimento tenham a interação de todos os membros equipe.

Na equipe não existe um líder técnico, e sim o *Scrum Master* que é o responsável pelo andamento das *Sprints*, que será descrito no próximo capítulo. Isto é, cada pessoa fica responsável por suas funções técnicas o que ajuda a motivar cada integrante da equipe (MARTINS, 2007).

Segundo Schwaber e Sutherland (2013), afirmam que “O tamanho ideal da Equipe de Desenvolvimento é pequeno o suficiente para se manter ágil e grande o suficiente para completar uma parcela significativa do trabalho”. Isto é, o time deverá ter poucos integrantes, cerca de 5 a 10 pessoas, o que facilita a interação da equipe. Entretanto, existem outros fatores importantes que explicam o fato das equipes de desenvolvimento ter poucas pessoas, como por exemplo, em equipes grandes os integrantes acabam deixando de fazer alguma tarefa, pois como há várias pessoas no time, existe a impressão que outro pode fazer; em equipes menores é difícil isso ocorrer já que todos sabem o que cada integrante está fazendo. Outro fator importante, devido à equipe ser menor, é a maior adesão dos integrantes do time nas reuniões, uma vez que no Scrum é recomendado que todos participem.

No Scrum os testes são realizados pelos próprios desenvolvedores. Recomenda-se que cada integrante assim que concluir um a implementação deverá verificar se existe alguma implementação concluída por outro integrante da equipe, e se existir, deverá realizar os testes necessários. Caso seja encontrado algum problema na implementação do outro integrante, o defeito deverá ser registrado, e

caso a implementação tenha sido realizada corretamente, deverá ser feito um registro de implementação realizada com sucesso. Posteriormente, o integrante deverá iniciar uma nova implementação, seguindo essa rotina até que todas as tarefas sejam realizadas e testadas. Assim que não existirem tarefas a serem realizadas, então todos os defeitos ora registrados deverão ser corrigidos, não necessariamente pelo desenvolvedor que a implementou, mas por qualquer membro da equipe, pois todos os integrantes deverá ter o conhecimento de todos as implementações que estão em andamento (SCHWABER; SUTHERLAND, 2013).

Basicamente, as principais funções da equipe são estimar o tempo de cada implementação a ser realizada e entregar todas essa implementações funcionando corretamente, no tempo previsto pelo próprio time.

- ***Scrum Master***

O *Scrum Master* é o líder da equipe de desenvolvimento, porém ele não é um líder técnico e sim um líder das pessoas integrantes do time de desenvolvimento, ele também tem como função gerenciar todo o processo de desenvolvimento da equipe. Segundo Martins (2007, p. 271), ele menciona que “o Scrum Master ocupa uma posição similar à ocupada pelo gerente de projeto [...]”.

Ele tem como principal responsabilidade entregar todas as implementações que foram definidas para serem realizadas em um determinado período de tempo, que foi estimado pela própria equipe de desenvolvimento juntamente com o *Scrum Master*. Deste modo, o *Scrum Master* se torna um interlocutor entre o negócio e a tecnologia.

Outra grande responsabilidade do *Scrum Master* é garantir que todas as regras da metodologia Scrum sejam seguidas durante o processo de desenvolvimento de *software*, sendo responsável também por marcar as reuniões e liderar as mesmas (SUTHERLAND; SCHWABER, 2013).

Recomenda-se que o *Scrum Master* não participe do desenvolvimento e nem dos testes realizados pela equipe, para que o mesmo fique exclusivamente na função de acompanhar todo o processo de desenvolvimento e propor soluções rápidas para as dificuldades que a equipe encontrará durante o período de

desenvolvimento, reduzindo os riscos de problemas na entrega do *software* na data definida.

O *Scrum Master* não deve resolver o problema sozinho ou delegar alguém para resolver, mas sim ajudar a equipe a encontrar uma solução para o problema encontrado (ORTH; PRIKLADNICK, 2009).

3.3.2 Artefatos

A metodologia Scrum não exige uma técnica específica para a sua utilização, porém há um conjunto de regras e práticas, que devem ser seguidas, as quais são chamadas de artefatos que serão apresentadas a seguir.

- ***Product Backlog***

Para Sutherland e Schwaber (2013), o *Product Backlog* é uma listagem de todas as funcionalidades que um determinado produto deve ter para que este agregue valor ao negócio do cliente, conforme foi planejado no projeto.

Essa lista é gerada em uma reunião onde estarão presentes todos os envolvidos no projeto, e será de responsabilidade do *Product Owner* indicar quais serão as funcionalidades e quais terão maior prioridade para serem desenvolvidas pela equipe de desenvolvimento. Nenhuma funcionalidade pode ser inserida fora da reunião, pois todos os envolvidos devem estar de acordo com cada item do *Product Backlog*.

O *Product Backlog* pode ser visto como uma coleta de requisitos e a lista são as tarefas que serão desenvolvidas durante o projeto. Essas tarefas são descritas na forma de *User Stories,* ou seja, serem feitas de forma simples e até mesmo um pouco superficial, pois o objetivo não é se aprofundar em cada funcionalidade, mas sim apontar as necessidades e os requisitos técnicos do negócio do cliente.

Como a ideia da metodologia Scrum é baseada no Manifesto Ágil, todos os requisitos coletados podem sofrer alterações a qualquer momento, então seria desperdício de tempo se aprofundar muito no início do planejamento.

- ***Sprint Backlog***

O *Sprint Backlog* é uma lista de atividades selecionadas da lista de tarefas do Product Backlog, que serão desenvolvidas em um período de tempo. Essas tarefas retiradas do *Product Backlog* são bem genéricas, então elas devem ser decompostas em várias atividades bem especificadas que formarão o *Sprint Backlog.*

O grande responsável por definir o *Sprint Backlog* é *Product Owner*, porém ele pode contar com a ajuda da equipe de desenvolvimento e do *Scrum Master*. Desta forma, deve ser gerado o *Sprint Backlog* de acordo com a prioridade de cada funcionalidade e com a estimativa de tempo para a execução de cada tarefa pela equipe de desenvolvimento (ORTH E PRIKLADNICK, 2009), conforme mostra a Figura 1.

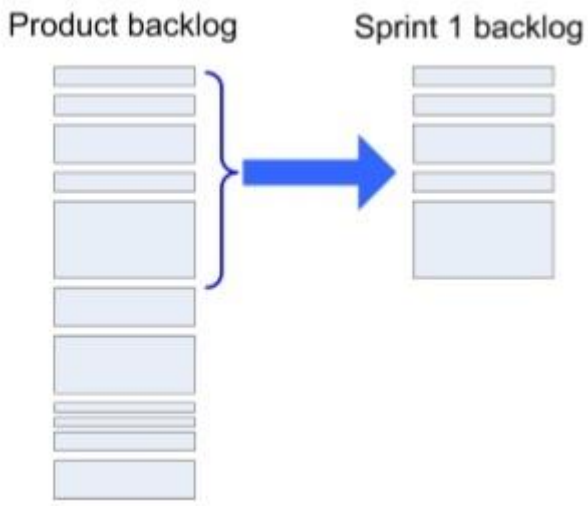

Figura 1 Seleção do *Sprint Backlog* (KNIBERG, 2007).

- ***Sprint***

Para Sutherland e Schwaber (2013), a Sprint é um período de tempo curto, geralmente limitado a um mês corrido. Este período pode ser considerado um ciclo

de trabalho que tem por objetivo entregar uma porção do *software* ou um incremento de *software* que agregue valor ao cliente, mesmo que ainda não contenha todas as funcionalidades que o produto final exige.

A lista de tarefas ou *Sprint Backlog*, que será desenvolvida durante a *Sprint*, será definida e estimado o tempo de cada tarefa em reunião conjunta entre o *PO*, *Scrum Master* e a equipe de desenvolvimento.

3.3.3 Reuniões

As reuniões são apresentadas pelo *Scrum Master,* nelas são definidas um planejamento das *Sprints*. A seguir, constam relacionados os tipos de reuniões.

- **Reunião de Planejamento da *Realease* (*Realease Planning Meeting*)**

A reunião de planejamento da *Realease*[5] acontece no início do projeto. Nessa reunião é realizado um planejamento inicial do projeto juntamente com o PO, *Scrum Master* e toda a equipe de desenvolvimento. O objetivo principal dessa reunião é criar planos e metas que estejam em comum senso entre a equipe de desenvolvimento e o cliente, estabelecendo metas e planos, onde todas as pessoas envolvidas no projeto possam se organizar e se comunicar de forma eficiente durante todo o projeto.

O *Product Owner* deve gerar todas as *User Stories* que vão formar o P*roduct Backlog*, que são as funcionalidades básicas do *software*, porém nessa reunião as funcionalidades devem ser colocadas de forma simples somente para que a equipe possa entender as funcionalidades básicas do projeto, haja vista que posteriormente as *User Stories* serão mais detalhadas em futuras reuniões.

- **Reunião de Planejamento da *Sprint* (*Sprint Planning Meeting*)**

[5] Texto contendo *User Stories*, definida na referida reunião.

Nesta reunião, que ocorre sempre antes de cada *Sprint*, devem estar presente o *Product Owner*, o *Scrum Master* e toda a equipe de desenvolvimento. O objetivo dessa reunião é criar a *Sprint Backlog* e fazer todo o planejamento da *Sprint*

que vai ser iniciada. O *Product Owner* deve então analisar o *Product Backlog* e selecionar quais são as *User Stories* que tem maior prioridade para serem desenvolvidas e avaliar se o tempo de execução das mesmas é compatível com o tempo de duração da *Sprint*.

Depois que a *Sprint Backlog* for gerada pelo *PO*, a equipe de desenvolvimento deverá analisar essas tarefas e tirar as dúvidas referente às implementações, buscando mais detalhes para que durante o desenvolvimento não seja necessário perguntar novamente.

Com todas as *User Stories* definidas e discutidas, então o time de desenvolvimento deverá juntamente com o *Scrum Master* analisar essas funcionalidades a serem desenvolvidas e dividi-las em várias tarefas para que possam ser implementadas por vários integrantes da equipe, a fim de facilitar na estimativa de tempo de cada tarefa. A forma como essas tarefas serão desenvolvidas é de responsabilidade da própria equipe de desenvolvimento, o que significa que o time pode decidir como fazer as implementações (SUTHERLAND; SCHWABER, 2013).

Para estimar as tarefas é utilizado um método denominado *Planning Poker*. Esse método é mais conhecido como um jogo e é utilizado na maiorias das equipes de desenvolvimento orientado pelo Scrum.

Nesse jogo, todos os integrantes do time recebem um conjunto de cartas que geralmente é baseada na sequência de Fibonacci[6]: 1, 2, 3, 5, 8, 13, 21 e 34. Essa numeração corresponde ao grau de dificuldade e são chamados de pontos. Deste modo, depois que as tarefas estão decompostas cada membro da equipe de desenvolvimento mostra ao mesmo tempo uma carta, se todos mostrarem a mesma carta, isso significa que todos entendem que aquela tarefa tem o mesmo grau de dificuldade, sendo então definido o grau de divindade da tarefa. Entretanto, não é

[6] Sequência de números naturais.

comum os integrantes da equipe mostrarem a mesma carta, então aqueles que mostraram cartas diferentes dos demais devem explicar o motivo e tentar convencer toda a equipe, e por fim é novamente jogado as cartas na mesa até que exista um consenso de todos os integrantes da equipe.

Antes mesmo de votar o grau de dificuldade das tarefas, a própria equipe deve fazer um cálculo e decidir quantos pontos a equipe consegue fazer durante uma *Sprint*, deste modo a equipe só poderá colocar na *Sprint* uma quantidade de tarefas que não passem da quantidade de pontos que a equipe suporta para aquela *Sprint*.

Com o uso da *Planning Poker* a equipe ganha mais interação e o time como um todo sabe todas as tarefas que serão desenvolvidas, mesmo que ele próprio não faça o desenvolvimento. Cada integrante da equipe também consegue entender vários pontos de vista para se realizar uma implementação, fazendo com que o desenvolvimento fique mais eficiente e o produto final seja mais valoroso para o cliente (PEREIRA; TORREÃO, MARÇAL, 2007).

- **Reunião Diária (*Daily Scrum Meeting*)**

Esta reunião é mais curta e simples, ela deve ocorrer todos os dias como o próprio nome indica. Ela deve acontecer no mesmo local onde é realizado o desenvolvimento do projeto e deve ser realizada com os integrantes da equipe, cujo o *Scrum Master* deve estar em pé, para que alcance seu objetivo de ser rápido e eficaz. Segundo Sutherland e Schwaber (2013), o tempo da reunião deve ser em média de 15 minutos, dependendo da quantidade de membros do time.

Cada integrante tem a oportunidade de dizer, sem muitos detalhes, o que está fazendo no momento, o que pretende fazer posteriormente e expor algum problema que está bloqueando seu desenvolvimento, se assim existir. O *Scrum Master* deve conduzir a reunião, e se for necessário, indicar quais tarefas que devem ser priorizadas na *Sprint*.

O objetivo principal dessa reunião é tão somente verificar o que cada integrante está fazendo e o que ele pretende fazer, além de fazer um levantamento de todos os problemas encontrados. Nessa reunião, não deve ser discutido

nenhuma solução para os problemas, apenas o *Scrum Master* registra os problemas e posteriormente, ele buscará alguma solução rápida (MARTINS, 2007).

- **Reunião de Revisão da *Sprint* (*Sprint Review*)**

Nesta reunião o principal objetivo é apresentar ao cliente o que foi desenvolvido durante a *Sprint*. Devem estar presente o PO, o *Scrum Master* e toda a equipe de desenvolvimento. O *Scrum Master* então mostrará detalhadamente cada funcionalidade que será desenvolvida e explicar de que maneira estes desenvolvimentos podem agregar valor ao negócio do cliente.

Ao fim da apresentação de cada funcionalidade desenvolvida, o *PO* tem que fazer uma análise e dizer se realmente aquele desenvolvimento atende às necessidades do cliente e se for o caso, indicar os ajustes necessários que devem ser realizado. Logo, devem ser geradas novas tarefas e incluídas no *Product Backlog* para serem desenvolvidas na próxima *Sprint* (SUTHERLAND; SCHWABER, 2013).

- **Retrospectiva da *Sprint* (*Sprint Retrospective*)**

Esta reunião ocorre geralmente após a reunião de Revisão da *Sprint*, pode-se dizer que esta reunião faz parte da reunião de Revisão. Esta reunião tem como objetivo verificar o que ocorreu de bom na Sprint e buscar soluções para problemas que ocorreram.

Basicamente cada integrante da equipe de desenvolvimento, o *Scrum Master* e o *PO* falam quais foram os pontos positivos e os pontos de melhorias da *Sprint*. O *Scrum Master* deve registrar todas as informações, juntamente com as possíveis soluções dos pontos de melhoria.

Sutherland e Schwaber (2013), diz que “[...] A Retrospectiva da Sprint fornece um evento dedicado e focado na inspeção e adaptação, no entanto, as melhorias podem ser adotadas a qualquer momento”.

3.4 Gráfico *Burndown*

O *Scrum* se utiliza desse gráfico para que seja visualizado com mais facilidade o andamento da *Sprint* e do Projeto de *software*. Este é um gráfico simples, porém a partir dele é possível verificar quanto já foi desenvolvido e quanto deveria ter sido desenvolvido até o momento, de acordo com a Figura 2.

Normalmente, no eixo Y é mostrado a quantidade que falta a ser desenvolvido, que geralmente está representado na unidade Horas ou Pontos, que são o grau de dificuldade, definidos a partir da *Planning Poker*. No eixo X, fica o período que geralmente está representado em Dia. Sendo assim, o gráfico terá uma linha que indicará a média de desenvolvimento, que deverá ser seguida para que se alcance o objetivo sem problemas, e também terá uma linha que indicará exatamente o quanto a equipe desenvolveu até um determinado período.

No *Scrum* existem dois gráficos *Burndown*, o primeiro é chamado de *Burndown Realease*, que mostra o andamento do *Product Backlog* e o segundo é o *Burndown Sprint* que indica o desenvolvimento na *Sprint* (PEREIRA, TORREÃO e MARÇAL, 2007).

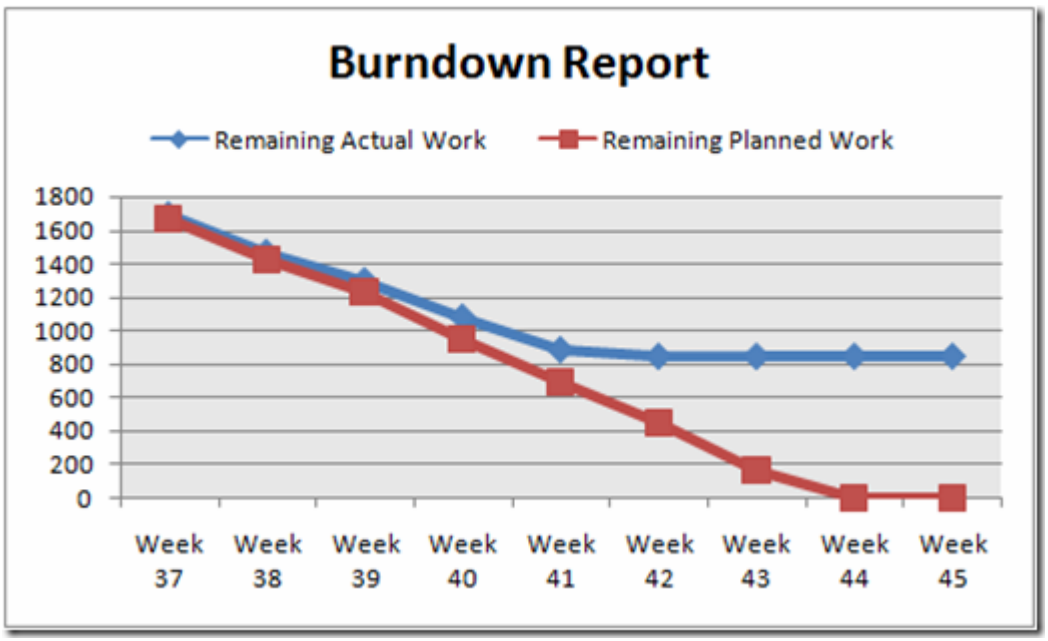

Figura 2 – Gráfico *Burndown* (GOOGLE, 2013).

3.5 Ferramentas de gerenciamento de Scrum

Existem várias ferramentas de controle de Scrum, algumas gratuitas e outras pagas. As empresas de menor porte podem usar sem problemas uma ferramenta gratuita para gerenciar as equipes de Scrum, tendo em vista que os *softwares* são muito parecidos e seguem em geral um mesmo padrão. Entretanto, para empresas de grande porte o ideal são ferramentas pagas, porque são mais consistentes e permitem mais usuários, além de agregar outras ferramentas, como por exemplo, uma ferramenta de gerenciamento de suporte pode ser agregada a uma ferramenta de gerenciamento de *Scrum*.

Dentre essas ferramentas de gerenciamento de *Scrum* gratuitas e pagas, existem algumas que se destacam no mercado, tais como, *Scrumrf*, *Scrumwise*, *Scrum Half*, Ferramentas exclusivas e a BraZip Scrum.

3.5.1 Scrumrf

Essa é uma ferramenta que oferece um serviço online gratuito para gerenciamento de projetos de *software*, seu objetivo é a grande facilidade de uso, onde as tarefas são direcionadas de forma simples e rápida e o acompanhamento é objetivo e visual, com gráficos automáticos e eficientes. O grande diferencial dessa ferramenta é a simplicidade, todas as suas funcionalidades são objetivas e de fácil usabilidade (www.scrumrf.com).

3.5.2 Scrumwise

Este *software* é mais recente, foi desenvolvido em 2009, e é mais completo, porém suas funcionalidades são simples de usar. Até o momento, ainda não existe uma versão em português, contudo é muito utilizado pelas empresas de desenvolvimento de *software*. Atualmente, é cobrado US$ 6,00 por usuário e tem 30 dias de uso grátis para se fazer uma avaliação (www.scrumwise.com).

3.5.3 Scrum Half

É uma ferramenta desenvolvida pela empresa brasileira especializada em gestão ágil, a GPE. Como é uma ferramenta brasileira é muito mais fácil o entendimento de suas funcionalidades, que são bem completas. Esta ferramenta não necessita de instalação, pois é inteiramente online. Até três pessoas podem utilizar a ferramenta de forma gratuita, no entanto a partir de quatro pessoas a ferramenta se torna paga com vários planos interessantes (www.myscrumhalf.com).

3.5.4 Ferramentas exclusivas

Nas empresas de pequeno e médio porte é muito comum que ferramentas de gerenciamento de Scrum sejam desenvolvidas pelas próprias empresas ou até mesmo por seus funcionários.

3.5.5 BraZip Scrum

Neste TCC, a ferramenta de gerenciamento de Scrum escolhida para realizar o estudo de caso é o BraZip Scrum. Esta é uma ferramenta da empresa BraZip, que tem o foco em soluções de TI e atua em todo o território nacional. No capítulo 4, esta ferramenta será mais detalhada na apresentação do estudo de caso.

4 ESTUDO DE CASO: APLICAÇÃO DA METODOLOGIA SCRUM EM UM PROJETO DE DESENVOLVIMENTO DE SISTEMA ERP

Com o objetivo de mostrar a aplicação prática da Metodologia Ágil Scrum, foi realizado um estudo de caso na empresa *Supply Integration Demand* (S2D), que desenvolve sistemas ERP. Anteriormente, a S2D trabalhava com metodologias tradicionais, mas resolveu adotar a metodologia Scrum em seu novo projeto.

Para mostrar o desenvolvimento de *software* deste novo projeto, foi escolhida o BraZip Scrum como ferramenta de gerenciamento Scrum.

4.1 Empresa de desenvolvimento de sistema ERP

A empresa S2D foi criada em 1996, quando duas empresas do ramo têxtil sentiram a necessidade de renovar seu sistema de informação e então contratou três pessoas para desenvolver um produto diferenciado, que atendesse todas as áreas da empresa em um único sistema.

O grande diferencial da S2D é a linguagem de programação exclusiva AKB, que foi desenvolvida pela própria S2D, e trabalha em alto nível de programação o que possibilita uma alta velocidade na produção de *software*.

Depois de desenvolvido todo o sistema para as duas empresas têxtil, a S2D se tornou independente e obteve novos clientes, que também queriam um sistema integrado (ERP) para controlar o processo de suas empresas.

Hoje, a S2D é uma empresa de pequeno porte que tem como cliente, na sua grande maioria, empresas do ramo têxtil, porém algumas empresas metalúrgicas e contábeis também se tornaram clientes.

4.2 Motivação da escolha da Metodologia Ágil Scrum

A S2D utilizava as metodologias tradicionais para o gerenciamento do desenvolvimento de *software*, porém percebeu que muitas empresas de desenvolvimento de *software* estavam adotando metodologias ágeis para o

gerenciamento de desenvolvimento, e estas empresas estavam tendo uma melhora significativa em sua produção.

Diante disso, a S2D resolveu implantar na empresa uma Metodologia Ágil, então foi feito uma pesquisa sobre as metodologias ágeis existentes e verificou-se que a metodologia Scrum se encaixava melhor no perfil da S2D; e também o Scrum era a metodologia mais utilizada e comprovadamente trazia melhores resultados para as empresas de desenvolvimento de *software*.

4.3 Ferramenta de Desenvolvimento Ágil Scrum: BraZip Scrum

Depois de definir a Metodologia Ágil Scrum, foi então feito uma pesquisa para verificar qual seria a melhor ferramenta com o melhor custo benefício do mercado. Então, a ferramenta BraZip Scrum surgiu como a melhor opção por ter algumas características que a S2D achou muito importante.

Esta ferramenta pertence à empresa BraZip que oferece soluções em TI. É uma ferramenta paga, mas comparado com os valores de outras ferramentas é relativamente barata. Primeiramente, existe uma versão teste para verificar as funcionalidades do *software,* que é gratuita. Assim que a ferramenta é contratada é cobrado um valor de R$ 20,00 por cada usuário.

O BraZip Scrum tem todas as funcionalidades necessária para gerenciar os projetos com a metodologia Scrum. Todas as suas funcionalidades são objetivas e de simples compreensão e usabilidade (www.brazip.com.br/brazipscrum).

4.4 Descrição do projeto

O projeto que será desenvolvido pela S2D, que terá o nome ERP, terá como objetivo atender as necessidades de melhorias no sistema integrado (ERP) desenvolvido pela própria S2D para uma empresa têxtil há alguns anos atrás.

Este projeto deverá verificar todas as melhorias necessárias e propor uma solução adequada que atenda às necessidades do cliente e agregue valor ao negócio do cliente. Como se trata de um sistema integrado (ERP), todas as áreas da empresa que tem sistema de informação deverão ser tratadas. Deste modo, como a

empresa é do ramo têxtil o projeto vai implementar melhorias nos módulos administrativo, financeiro, comercial, industrial, desenvolvimento de produto e logística.

4.5 Fases do desenvolvimento de sistema aplicando a metodologia Scrum e utilizando a ferramenta BraZip

Para ser demonstrado a aplicação da Metodologia Ágil Scrum no projeto que a empresa S2D vai desenvolver, foi selecionado a ferramenta BraZip Scrum. Dessa forma, para melhor explicar a utilização do Scrum em um projeto de *software*, o desenvolvimento foi dividido em quatro fases: Planejamento do projeto, Planejamento da *Sprint*, Desenvolvimento do *software* e Entrega do bloco de *software* que foi desenvolvido na *Sprint*.

4.5.1 Planejamento do projeto

A primeira etapa do projeto consiste em uma reunião inicial onde todas as pessoas envolvidas no projeto devem estar presentes. Esta reunião tem o nome de Reunião de Planejamento da *Realease*, nela deve estar presente o PO, que é o representante do cliente, o *Scrum Master,* que é a pessoa responsável pelo gerenciamento e desenvolvimento do *software*, e a equipe de desenvolvimento, que neste caso terá seis integrantes.

No início da reunião o PO faz uma explicação objetiva e simples do projeto. Neste caso, o projeto se trata de melhorias em um sistema integrado que foi desenvolvido pela própria S2D, então o PO explica para todos que o projeto vai abranger os módulos administrativo, financeiro, comercial, industrial, desenvolvimento de produto e logística. Diante disso, será necessário criar pelo menos um *Product Backlog* para cada um destes módulos.

Então, o PO solicita para o *Scrum Master* abrir a ferramenta BraZip Scrum e criar um novo projeto com o nome ERP, como pode ser visto na Figura 3.

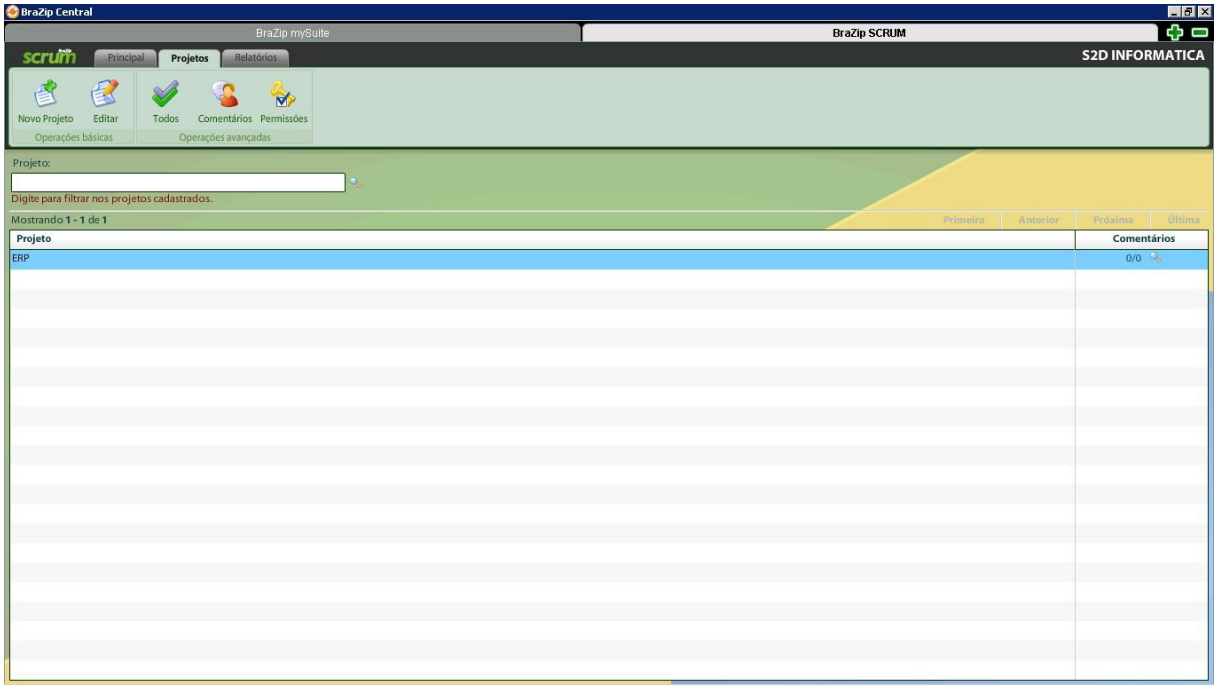

Figura 3 – Projeto ERP (AUTORIA PRÓPRIA, 2013).

Depois que o projeto ERP foi criado, ele deve ser selecionado no BraZip Scrum e deve então ser gerado os *Product Backlog* necessários para o projeto. Cada módulo do sistema integrado terá um *Product Backlog*, então o *Scrum Master,* a partir da solicitação do PO, gera todos os *Product Backlog* necessário para o projeto na ferramenta BraZip Scrum, como pode ser visto na Figura 4.

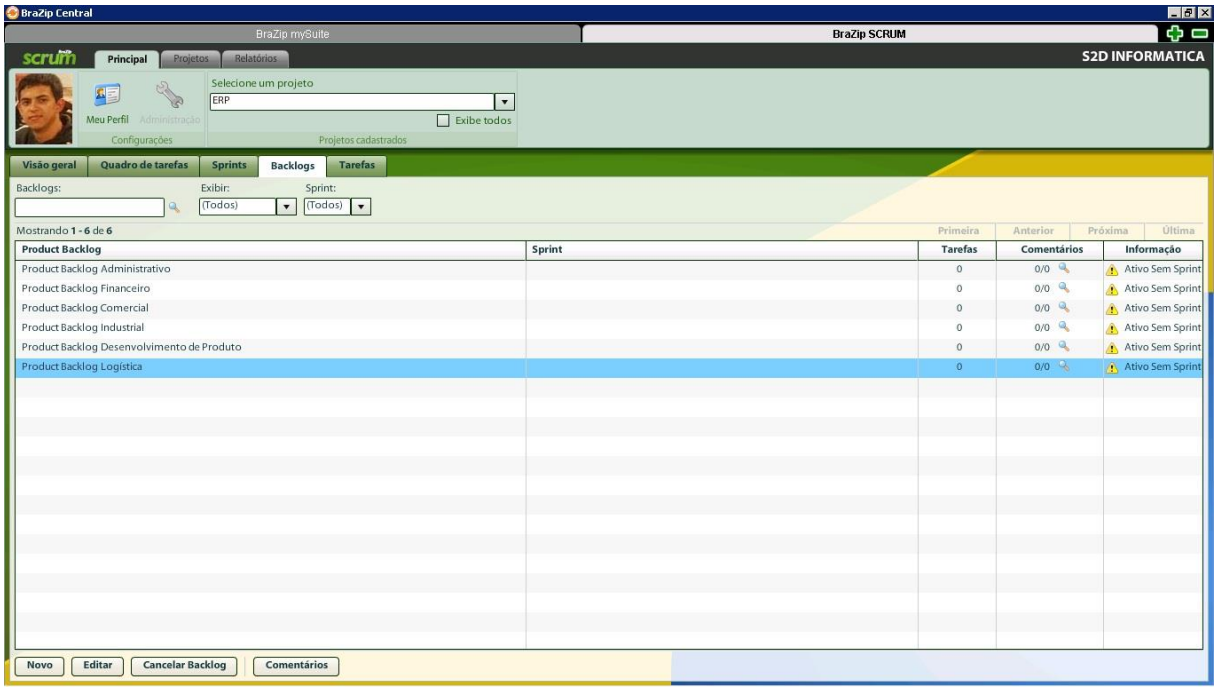

Figura 4 – *Product Backlog* (AUTORIA PRÓPRIA, 2013).

Após todos os *Product Backlogs* serem gerados, as *User Stories* de cada *Product Backlog* devem ser criadas. No caso da ferramenta BraZip Scrum, essas *User Stories* são chamadas de tarefas. Deste modo, o PO explica as *User Stories* e informa qual é a prioridade de cada uma delas e por sua vez, o *Scrum Master* tem o dever de criar cada uma delas no BraZip Scrum e associá-la a algum *Product Backlog*.

As prioridades na ferramenta BraZip Scrum são controladas pela cor. Isto posto, cada cor determina um grau de dificuldade. A cor Verde representa a prioridade Baixa, a cor Amarela representa a prioridade Média e por fim a cor Vermelha representa a prioridade Alta.

Como exemplo, temos o *Product Backlog* Logística, neste módulo será criada uma *User Story*, que solicita que seja criado uma tela para cadastrar as localizações de todos os produtos que estão dentro da empresa têxtil, com prioridade alta, como pode ser visualizado na próxima Figura. Nesta funcionalidade é possível perceber que ela é descrita de forma bem abrangente e sem muitos detalhes, pois está *User Story* será decomposta posteriormente e dividida em várias tarefas que serão mais bem detalhadas nos *Sprint Backlogs*, vide Figura 5.

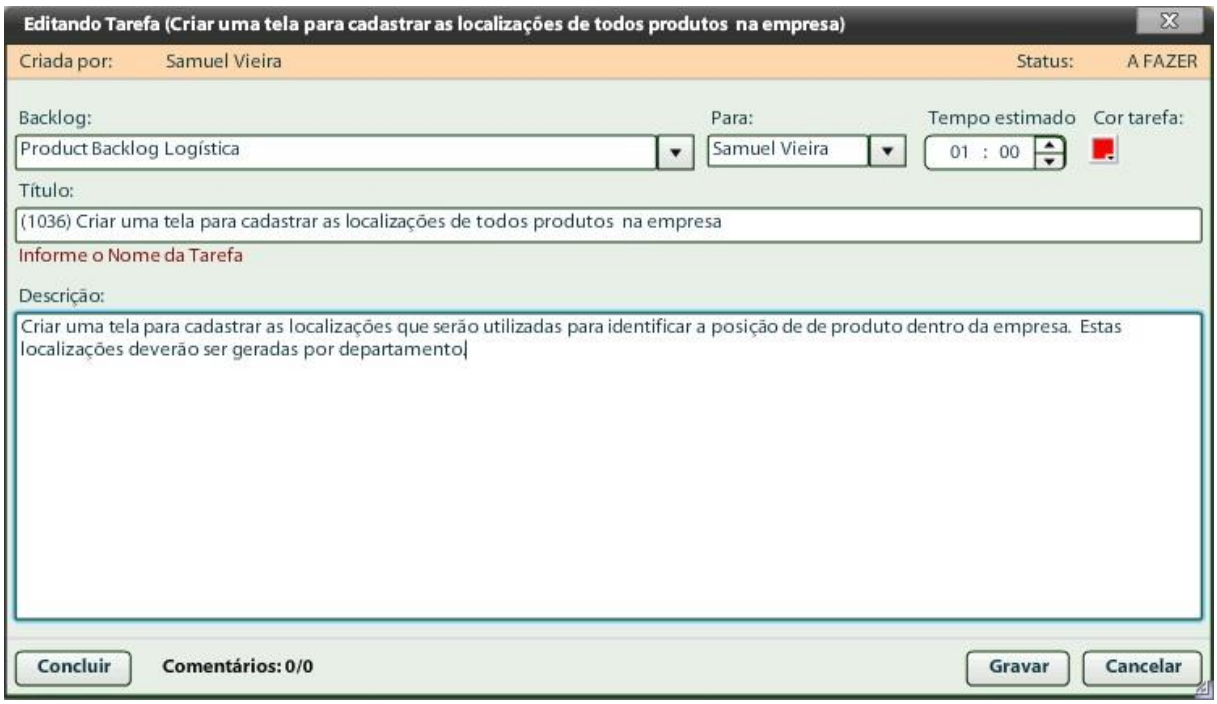

Figura 5 – *User Story* (AUTORIA PRÓPRIA, 2013).

Assim que as *User Stories* forem geradas, o PO deve explicá-las ao *Scrum Master* e a equipe de desenvolvimento, e o time juntamente com *Scrum Master* devem sanar todas as suas dúvidas em relação a cada *User Story* que for gerada, possibilitando que todos envolvidos no projeto estejam em comum acordo e falando a mesma língua.

Seguindo estes conceitos, todas as *User Stories* de cada *Product Backlog* devem ser geradas e discutidas, assim ao findar a reunião será estipulado um prazo para se concluir o projeto, entretanto esse mensuramento é muito difícil de fazer, uma vez que os requisitos extraídos e apresentados na reunião de planejamento da *Realease* podem ser alterados pelo próprio cliente, durante o processo de desenvolvimento do projeto.

Nesta reunião também deve ser discutido quanto tempo vai durar cada *Sprint* durante o projeto. Neste caso, como as equipes são de tamanho menor o recomendado é que a *Sprint* tenha duração de 10 dias úteis.

4.5.2 Planejamento da *Sprint*

Na segunda fase do desenvolvimento do projeto, chamado de Planejamento da *Sprint*, deve ser realizada uma reunião intitulada “Reunião de planejamento da *Sprint*”, onde o PO, o *Scrum Master* e a equipe de desenvolvimento devem estar presentes. Nesta reunião, o PO deve verificar quais são as *User Stories* de cada *Product Backlog* que tem maior prioridade de serem desenvolvidas, para serem incluídas no *Sprint Backlog*.

Assim que as *User Stories* forem selecionadas e discutidas,o *Scrum Master* deve gerar todos os *Sprint Backlogs* que serão associadas e desenvolvidas na próxima *Sprint*, que neste caso será o primeiro, conforme Figura 6.

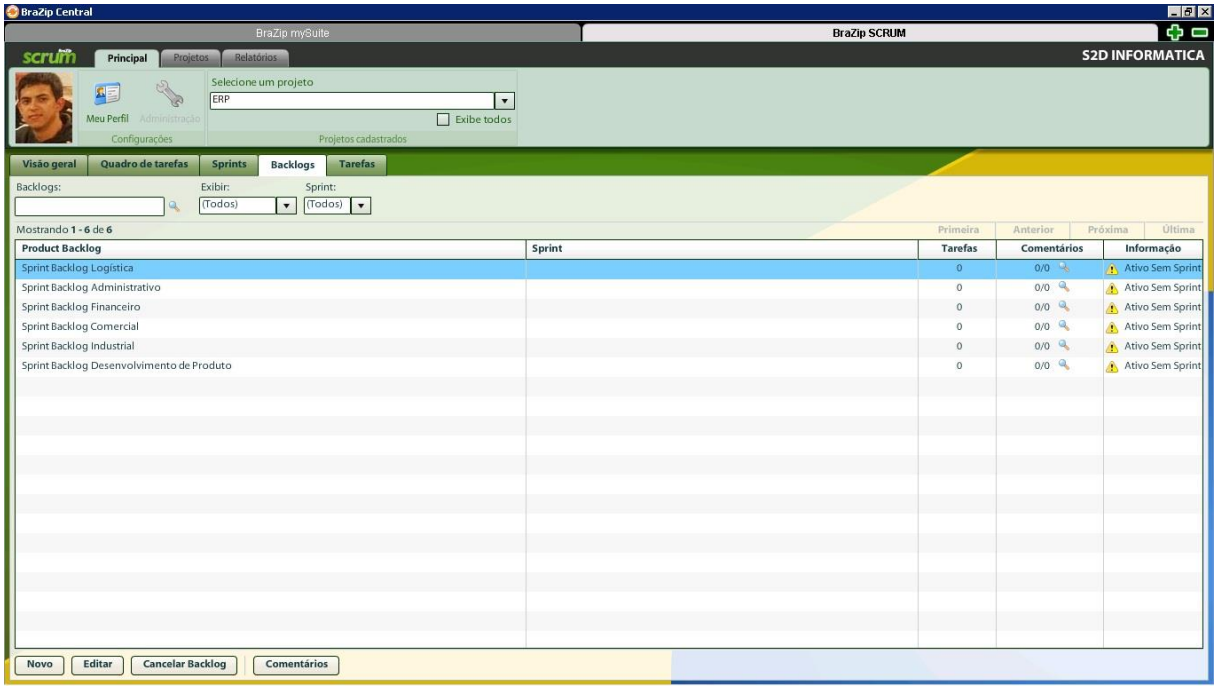

Figura 6 – *Sprint Backlog* (AUTORIA PRÓPRIA, 2013).

Após todos os *Sprint Backlog* serem gerados, o *Scrum Master* juntamente com a equipe devem analisar cada uma das *User Story* e dividí-las em várias tarefas. Como exemplo, temos a *User Story* 1036 que está no *Product Backlog* Logística, a partir desta são geradas mais quatro tarefas, como mostra a Figura 7, em que foi gerada a primeira tarefa que solicita que seja criada as tabelas necessárias para o funcionamento da tela Cadastro de localização. Todas essas tarefas que forem geradas devem ser associadas em algum *Sprint Backlog*.

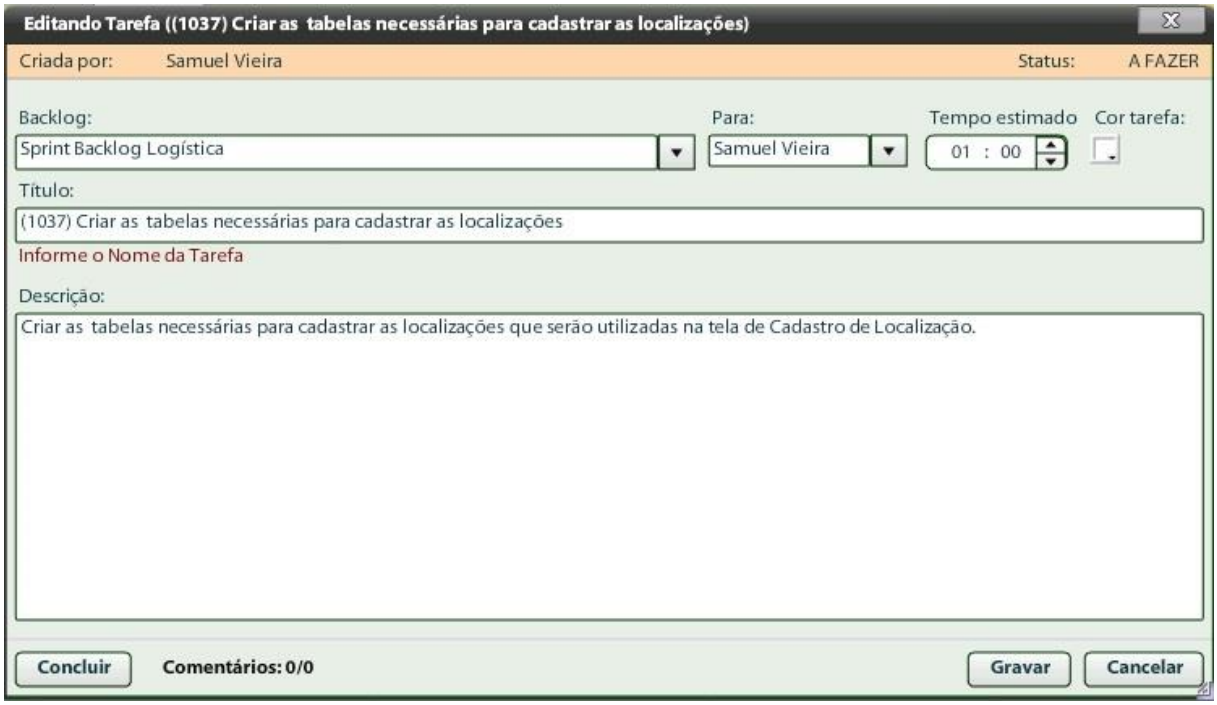

Figura 7 – Tarefas geradas pela *User Story* 1036 (AUTORIA PRÓPRIA, 2013).

No momento que for sendo geradas estas tarefas, a funcionalidade que antes era bem generalizada e sem muitas especificações, agora estão muito mais detalhadas. Então, todos da equipe de desenvolvimento e o *Scrum Master* devem sanar todas as dúvidas sobre cada tarefa, para que fique bem explicado o que deverá ser feito em cada funcionalidade solicitada pelo cliente.

Em seguida, a equipe precisa juntamente com o *Scrum Master* verificar qual será a velocidade da equipe durante a *Sprint*. No Scrum, o recomendado é que essa velocidade seja mensurada pela unidade de medida em pontos, que são baseados na sequência de Fibonacci. Basicamente, deve-se chegar a uma pontuação base, porém pode ser alterado nas próximas *Sprints*. No caso da S2D, foi estipulado que cada funcionário tem a capacidade de fazer cerca de 20 pontos em uma *Sprint* de 10 dias úteis, pois como a equipe é composta de seis integrantes o time terá uma velocidade de 120 pontos durante a primeira *Sprint*.

Com a velocidade da equipe definida, todas as tarefas necessárias cadastradas em seu respectivo *Sprint Backlog* e todas as dúvidas sanadas, o *Scrum Master* e a equipe de desenvolvimento devem jogar o *Planning Poker,* a fim de mensurar cada uma das tarefas.

As tarefas são discutidas e votadas uma a uma, ordenadas por sua prioridade. Como a velocidade da equipe é de 120 pontos, devem ser votadas as velocidades das tarefas que somando, não ultrapassem a velocidade da equipe. Deste modo, depois que as tarefas estiverem com suas pontuações definidas, o *Scrum Master* deve criar o *Sprint* 1, conforme Figura 8, e inserir todas as tarefas votadas nesta *Sprint* como pode ser visualizado na Figura 9.

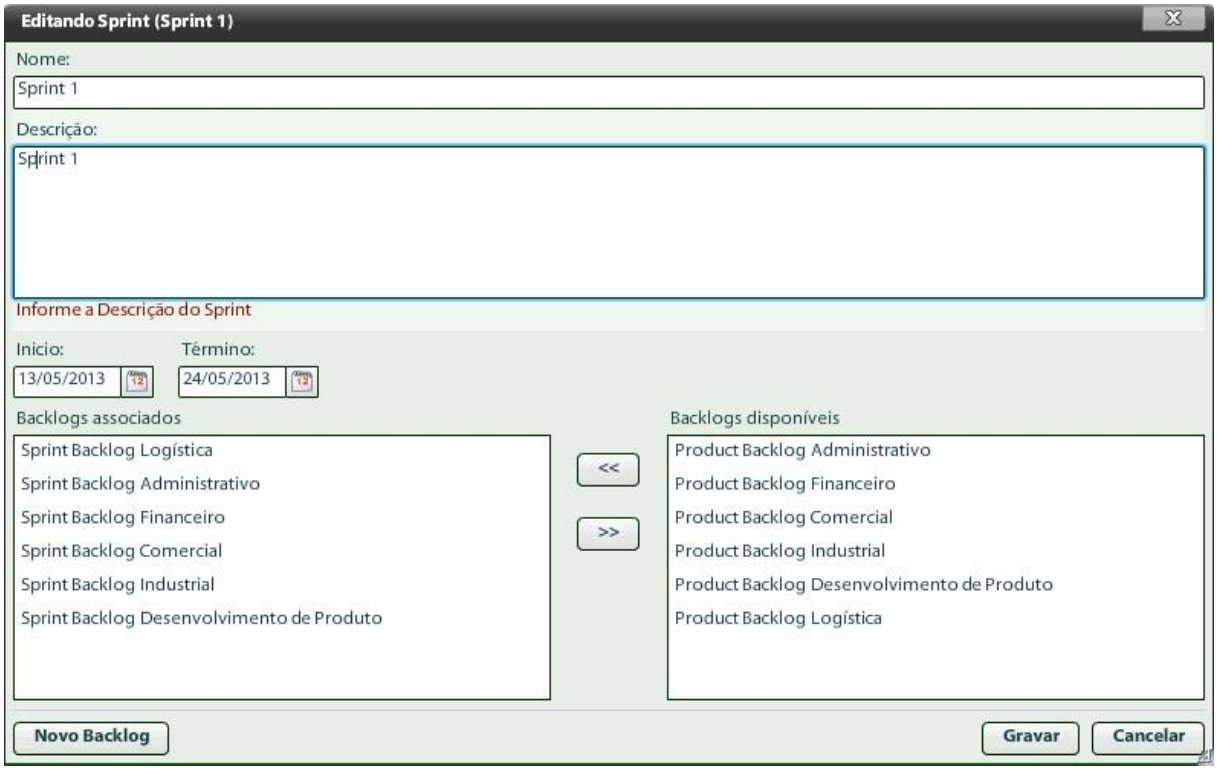

Figura 8 – *Sprint* 1 (AUTORIA PRÓPRIA, 2013).

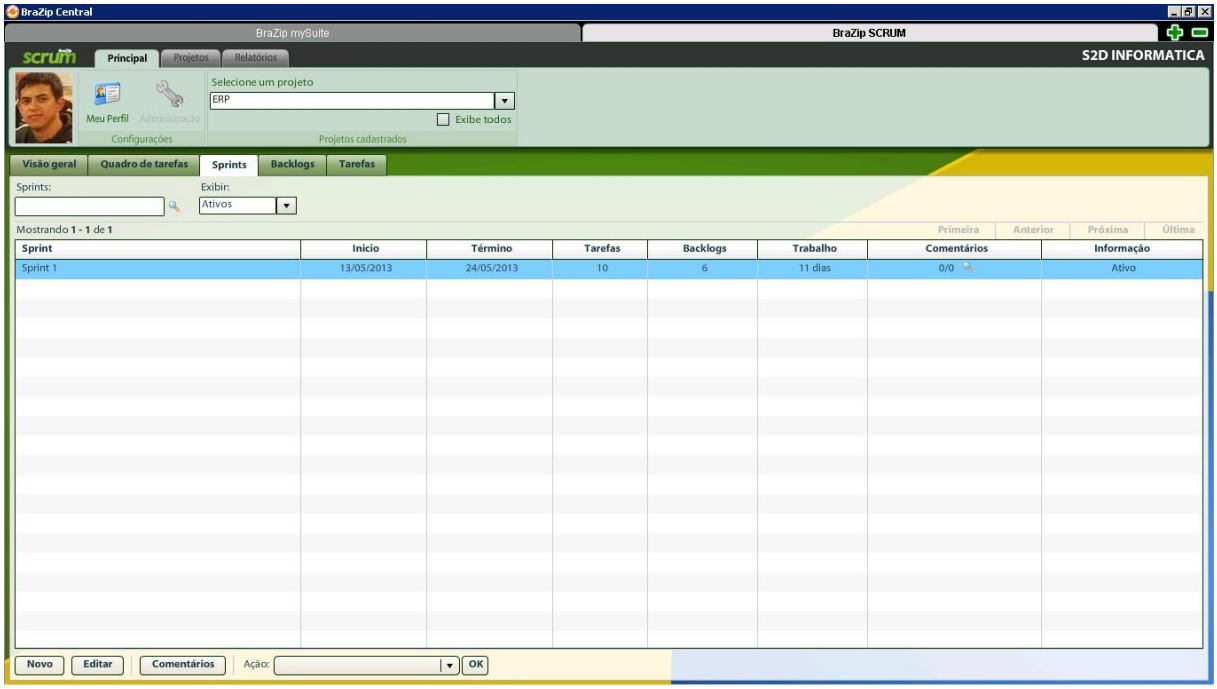

Figura 9 – Tarefas inseridas na *Sprint* 1 (AUTORIA PRÓPRIA, 2013).

4.5.3 Desenvolvimento do *software*

Depois que as reuniões de planejamento da *Realease* e da *Sprint* forem encerradas, deve ser iniciado o desenvolvimento do *software*. A equipe de desenvolvimento utilizará o quadro de tarefa na ferramenta BraZip Scrum, conforme Figura 10, neste quadro é possível visualizar rapidamente todas as tarefas que estiverem A Fazer, Em Execução, A Verificar e Concluídas.

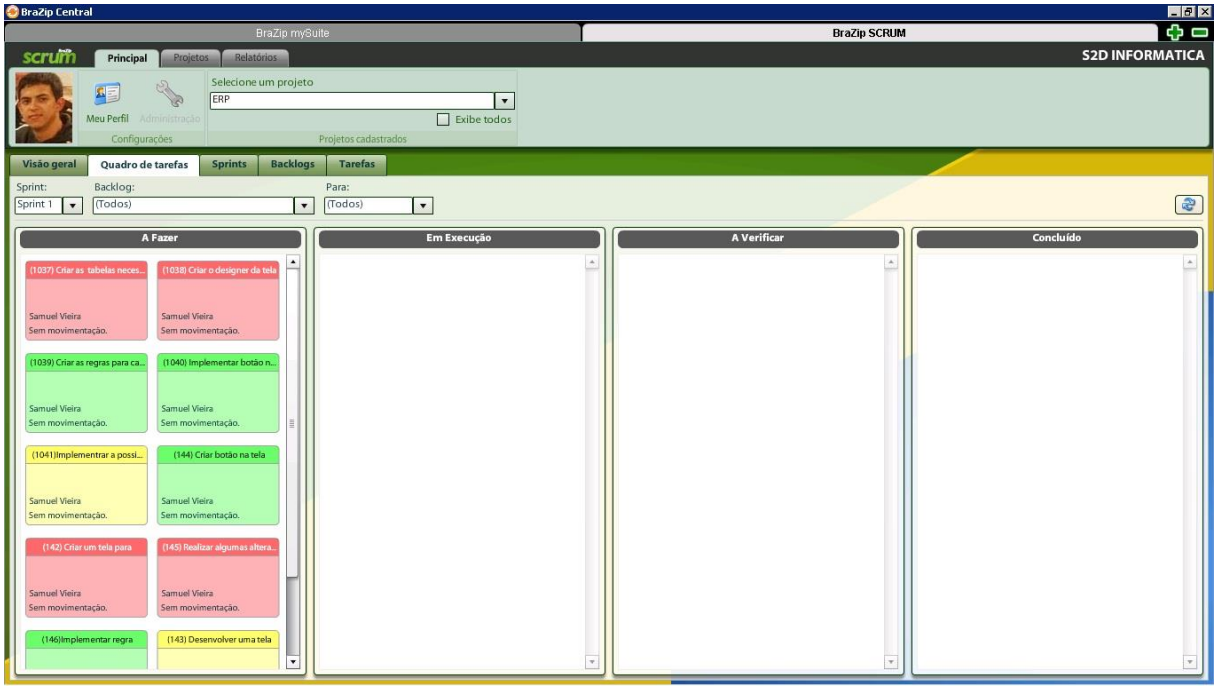

Figura 10 – Quadro de tarefas (AUTORIA PRÓPRIA, 2013).

Inicialmente, todas as tarefas estarão no primeiro bloco, denominado A Fazer, então o desenvolvedor deve pegar a primeira tarefa e arrastá-la para o bloco ao lado, cujo nome é Em Execução, automaticamente esta tarefa vai ficar alocada com o nome deste desenvolvedor e então outro desenvolvedor não poderá alocar esta mesma tarefa. Depois que a tarefa estiver implementada, o desenvolvedor deverá arrastar esta tarefa ao bloco do lado direito, denominado A Verificar, uma vez que a tarefa estiver nesse bloco, outro desenvolvedor poderá selecionar e testá-la.

Conforme o desenvolvedor for realizando as implementações necessárias de uma tarefa, ele deverá fazer todas as anotações necessárias, para que outro desenvolvedor possa entender rapidamente o que foi implementado naquela tarefa. No exemplo da tarefa 1037, de acordo com a Figura 11, pode ser visualizada uma observação que foi inserida nessa tarefa.

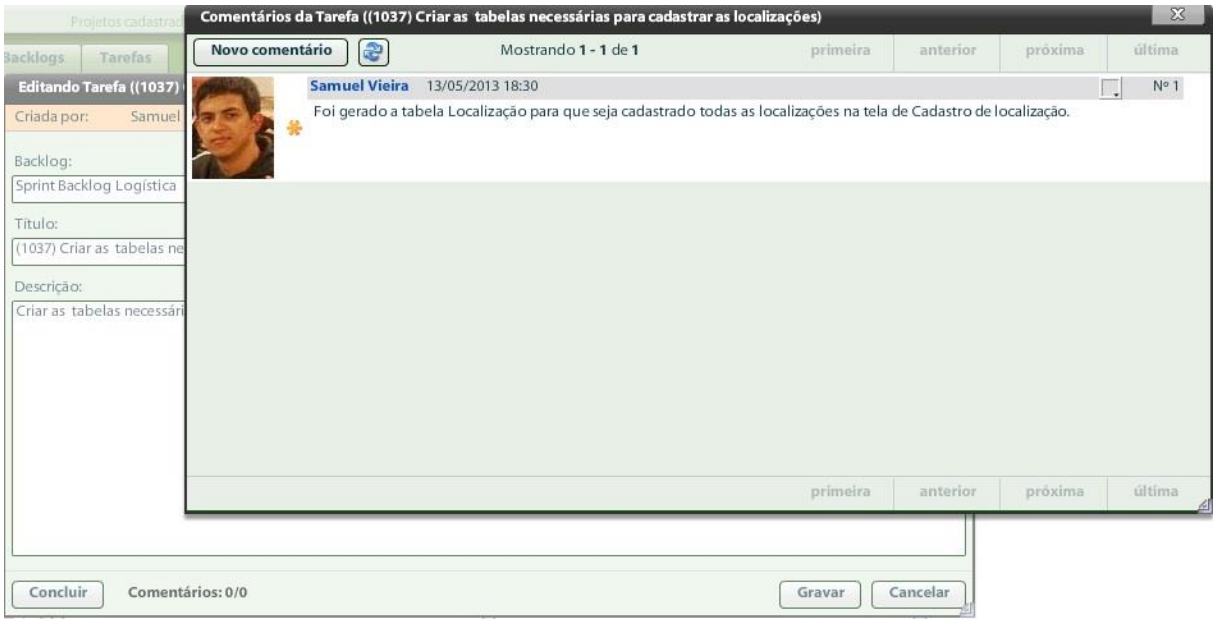

Figura 11 – Tarefa 1037 (AUTORIA PRÓPRIA, 2013).

Assim que o desenvolvedor concluir as implementações de uma tarefa, ele deverá observar o quadro de tarefas e selecionar a tarefa com maior prioridade do bloco A Verificar e realizar os testes necessários. Se a tarefa estiver correta, o desenvolvedor deverá arrastá-la para o bloco Concluído. Entretanto, se a tarefa apresentar algum erro ou não estiver corretamente implementada, ela deverá ser arrastada para o lado esquerdo, de volta para o bloco Em Execução e deverá ser informado todos os erros apresentados nas observações. Esta tarefa que apresentou erros em sua implementação não necessariamente deve ser corrigida pelo próprio desenvolvedor que a implementou, mas sim por qualquer desenvolvedor da equipe que for selecionar a próxima tarefa a ser implementada.

Após testar uma tarefa, o desenvolvedor deverá selecionar a próxima tarefa, conforme a prioridade e implementá-la. Este procedimento deve ser seguido por todos os desenvolvedores da equipe, até que todas as tarefas estejam no bloco Concluído.

Para que o *Scrum Master*, o PO e a equipe de desenvolvimento possam verificar o andamento estatístico da *Sprint* existe o gráfico Burndown, conforme Figura 12. Neste gráfico é possível analisar rapidamente o andamento da *Sprint* e verificar se está conforme o previsto ou se está atrasado, ou até mesmo verificar se a *Sprint* está adiantada. Este gráfico deve ser visualizado todos os dias pela equipe, principalmente pelo *Scrum Master* que é o responsável pela entrega dentro do prazo que foi estabelecido no planejamento da *Sprint*.

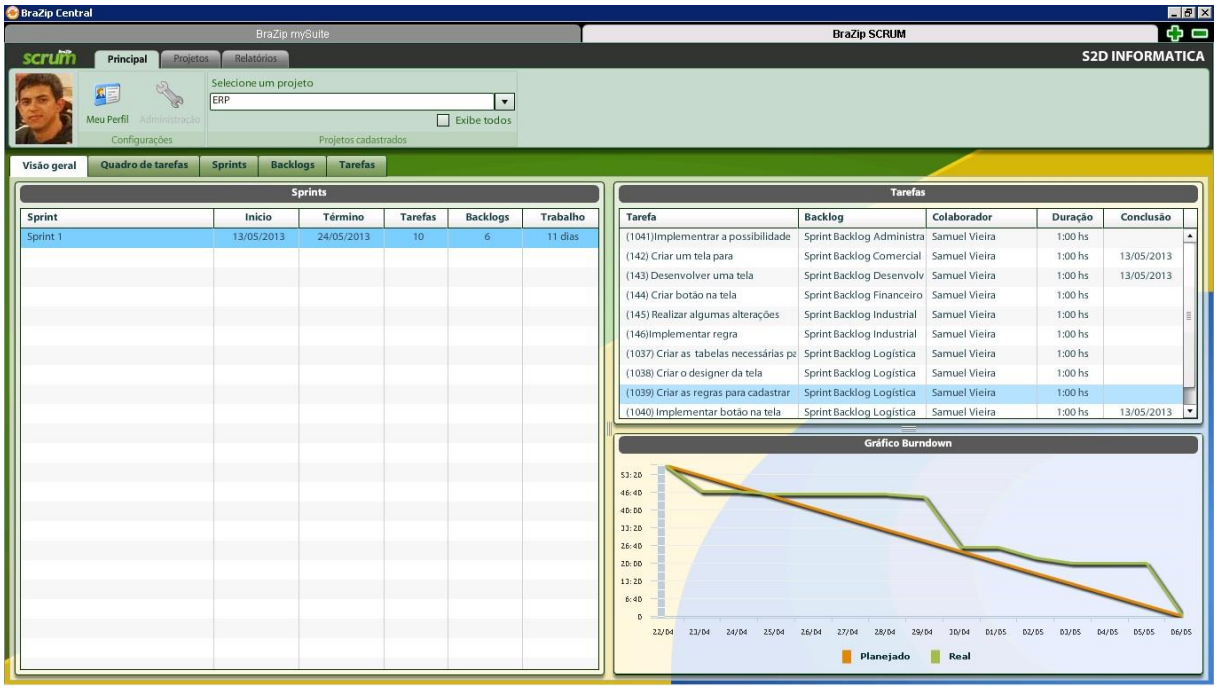

Figura 12 – Gráfico *Burndown* (AUTORIA PRÓPRIA, 2013).

Durante o desenvolvimento da *Sprint*, o *Scrum Master* deve realizar as reuniões Daily Meeting[7] juntamente com a equipe de desenvolvimento todos os dias. Nestas reuniões diárias, cada desenvolvedor explicará em poucas palavras o que está fazendo, o que vai fazer e se existe algum impedimento em sua implementação. Se existir algum impedimento, após a reunião, o *Scrum Master* deverá auxiliar a equipe de desenvolvimento a solucionar o problema em questão.

[7] Tradução: Reunião Diária

4.5.4 Entrega do bloco de *software* que foi desenvolvido na *Sprint*

Após os 10 dias úteis de desenvolvimento da *Sprint,* a reunião de revisão deve ser realizada, onde o PO, o *Scrum Master* e a equipe de desenvolvimento estarão presentes. Nesta reunião, o *Scrum Master* deve apresentar na prática todas as funcionalidades desenvolvidas durante a *Sprint* para que o PO possa analisar e verificar se está funcionando corretamente, conforme foi solicitado.

Como no exemplo da tela de cadastro de localização, ela deve ser apresentada para o PO pelo *Scrum Master*, com todas as funcionalidades solicitadas no planejamento da *Sprint*.

Todas as tarefas que forem consideradas corretas permanecerão no bloco Concluído da mesma *Sprint* que foi desenvolvida, entretanto todas as tarefas que forem encontradas algum problema pelo PO, estas também ficará no bloco Concluído da *Sprint*, mas será criada uma nova tarefa, detalhando o que falta para ser concluído efetivamente a funcionalidade que apresentou problemas. Estas novas tarefas que forem geradas deverão ser inseridas no *Product Backlog* para serem incluídas na próxima *Sprint*.

Após todas as funcionalidades serem validadas pelo PO, a próxima reunião pode ser iniciada. A reunião de retrospectiva ocorre logo em seguida à reunião de revisão e conta com os mesmos integrantes. Nesta reunião, deve ser analisado o que ocorreu de bom e o que precisa ser melhorado durante a *Sprint*.

O PO, o *Scrum Master* e cada integrante da equipe de desenvolvimento devem falar quais foram os pontos positivos e de melhoria durante o desenvolvimento, todas estas observações deverão ser registradas pelo *Scrum Master*. Deste modo, todos os pontos deverão ser refletidos por todos os integrantes da reunião, para que seja absorvido todos os pontos positivos e para que seja encontrado soluções para os pontos de melhorias.

4.6 Conclusão do projeto

Todas as fases do processo de desenvolvimento do projeto: Planejamento da *Sprint*, Desenvolvimento do *software* e Entrega do bloco de *software* que foi desenvolvido na *Sprint,* deverão ser realizado até que todas as *User Stories* forem concluídas. Mesmo tendo uma previsão de término do projeto, ora determinada na fase de planejamento do projeto, nem sempre é possível entregar na data prevista, pois ao longo do projeto as funcionalidades e requisitos podem ser alterados.

No entanto, seguindo os valores e regras da metodologia Scrum todas as funcionalidades solicitadas pelo cliente serão realizadas independentemente se foi solicita no início ou no meio do projeto, pois o importante é o sistema atender as necessidades do negócio do cliente e o mesmo ficará satisfeito com o produto final e se tornará um cliente fiel da empresa S2D.

5 RESULTADOS OBTIDOS COM A APLICAÇÃO DA METODOLOGIA ÁGIL SCRUM

Com a aplicação da Metodologia Ágil Scrum verificou-se várias mudanças no processo de desenvolvimento de *software* e percebeu-se que os resultados obtidos foram bem diferentes de quando era utilizada a metodologia tradicional de desenvolvimento.

Os resultados obtidos podem ser analisados em duas vertentes, a primeira são os inúmeros benefícios e vantagens que o Scrum proporcionou para a empresa S2D e também para seus clientes.

Entretanto, é claro que todas as mudanças tem lados positivos e negativos e no caso do Scrum, isso ocorre também, existindo algumas desvantagens em relação a metodologia tradicional.

5.1 Benefícios da utilização do Scrum

São vários os benefícios obtidos na aplicação da Metodologia Ágil Scrum tanto para a empresa de desenvolvimento quanto para as empresas-clientes. Serão detalhadas as vantagens mais importantes da utilização do Scrum, a partir da visão da empresa S2D na aplicação em um projeto ERP.

5.1.1 Cliente fica mais presente durante o projeto

Na empresa S2D antes da utilização do Scrum, quando ainda eram utilizadas metodologias tradicionais de desenvolvimento, o cliente tinha um papel muito importante no início do projeto, porém, após os requisitos serem recolhidos, o cliente não era mais consultado durante o desenvolvimento do projeto, e geralmente, apenas no fim, o cliente teria acesso ao sistema, que neste momento já estaria pronto.

Essa mudança foi bastante evidente, pois com a utilização do Scrum, o cliente tem papel ativo desde os requisitos inicias até a entrega do produto final. O que permite que o cliente possa realizar mudanças nas funcionalidades, se assim achar

necessário e importante, possibilitando que o cliente faça melhorias durante o desenvolvimento e consequentemente, seja alcançado no fim do projeto um produto que atenda completamente as necessidades do cliente.

Outra vantagem bastante importante com a presença contínua do cliente é a identificação de erros na funcionalidade do *software*, pois quanto mais rápido o cliente verificar um erro, mais rápido e fácil será a alteração do *software*.

5.1.2 Cliente recebe produto com maior valor de agregação ao seu negócio

O principal objetivo da empresa S2D e provavelmente para a grande maioria das empresas de desenvolvimento de *software* é fazer um produto que agregue valor ao cliente e que atenda suas necessidades. Com a utilização do Scrum a empresa S2D pode perceber que seus *softwares* passaram a ser mais qualificados e mais eficientes para seus clientes.

As vantagens do Scrum foram facilmente perceptíveis, tais como, o produto final ter agradado ainda mais o cliente, devido a possibilidade do mesmo fazer alterações contínuas nas funcionalidades, gerando assim um produto final mais próximo do desejado pelo cliente.

5.1.3 Arquitetura preparada para mudanças durante o projeto

Um dos principais diferenciais das metodologias ágeis é a equipe estar preparada para mudanças durante o projeto e o Scrum tem esse valor muito bem evidenciado em suas características e regras. No Scrum é recomendado que a equipe de desenvolvimento receba as mudanças de forma positiva, pois significa que o produto final está cada vez mais próximo de se tornar um produto com grande valor.

Com a aplicação do Scrum, a empresa S2D pode perceber que o desenvolvimento pode estar em andamento e os requisitos podem estar definidos, mas, se houver mudanças a equipe de desenvolvimento apenas verifica o que deve ser alterado, faz o planejamento durante as reuniões e executa as alterações. Isso faz com o Scrum tenha um diferencial em relação às metodologias tradicionais, onde

era necessário voltar uma etapa do desenvolvimento do projeto, o que causava grandes transtornos e atrasava consideravelmente o projeto.

5.1.4 Aumento da produtividade do desenvolvimento

A produtividade é vista por muitas empresas de desenvolvimento como um fator de extrema importância, assim como para a S2D. A partir da utilização do Scrum pode-se observar que a S2D teve um aumento significativo em sua produtividade de *software*.

O processo de desenvolvimento passou a ser mais objetivo e eficiente, eliminando todas as burocracias e etapas desnecessárias do desenvolvimento do projeto. Como exemplo, a documentação extensa e desnecessária foi deixada de lado, a equipe de desenvolvimento passou a trabalhar em várias áreas do desenvolvimento, o cliente passou a verificar os erros com maior antecedência, e isso gerou o aumento na produtividade do desenvolvimento dos projetos.

5.1.5 Gerenciamento do desenvolvimento simplificado e objetivo

Uma das palavras chaves e importante no Scrum é a simplicidade, uma vez que simplificando o processo de desenvolvimento, ele tende a ser mais eficiente e rápido. A S2D pode verificar que quanto menos burocracia, mais rápido o projeto será entregue ao cliente.

A simplicidade no desenvolvimento agiliza o próprio desenvolvimento do projeto e também facilita futuras mudanças de requisitos e manutenções no *software*. Além desses fatores que facilitam o desenvolvimento, o cliente terá vantagens, como a facilidade em controlar as funcionalidades do *software*.

5.1.6 Equipe de desenvolvimento mais motivada

Os integrantes da equipe de desenvolvimento da empresa S2D que antes tinham suas funções limitadas e restritas em uma única função, agora passaram a ser mais dinâmicos dentro do projeto.

Todos os membros da equipe de desenvolvimento não tem um papel específico dentro do projeto, os integrantes participam da extração dos requisitos, análise das funcionalidades, implementação, testes e entrega do projeto. Deste modo, os integrantes passam a ter um conhecimento mais amplo do sistema o que facilita que o membro da equipe possa desenvolver e fazer alterações no *software* com mais facilidade e eficiência.

Os membros da equipe de desenvolvimento também passam a ter mais responsabilidade dentro do projeto, pois esta metodologia ágil permite que cada integrante tenha a capacidade de resolver problemas sem precisar envolver muitas pessoas, porém se for necessário, o *Scrum Master* tem o dever de dar o suporte adequado para solucionar o problema. Estes fatores fazem com que toda a equipe de desenvolvimento tenha mais motivação durante o projeto, sentindo-se mais valorizados.

5.1.7 Maior compartilhamento de conhecimento entre o time

Quando a empresa S2D utilizava metodologias tradicionais no desenvolvimento, cada desenvolvedor era responsável por uma área do projeto, o que muitas vezes causava atrasos no desenvolvimento. Já com a utilização do Scrum, a S2D pode perceber que o conhecimento dos integrantes passou a ser um conhecimento geral do projeto, pois mesmo que um integrante esteja desenvolvendo um módulo específico durante a *Sprint*, este terá conhecimento sobre o que está sendo desenvolvido nos outros módulos também, isso ocorre através da reunião de planejamento do projeto, das reuniões diárias, do planejamento das *Sprints* e dos testes que geram um conhecimento amplo ao desenvolvedor.

Com um compartilhamento de conhecimento maior os integrantes da equipe se tornam mais eficientes, o que permite que o desenvolvedor aumente sua produtividade.

5.1.8 Melhor relacionamento entre pessoas de negócio e os desenvolvedores

A partir da aplicação do Scrum na empresa S2D pode-se perceber uma evidente aproximação entre o desenvolvedor com as pessoas de negócio. Nas

metodologias tradicionais o desenvolvedor fica mais isolado e tinha apenas a missão de programar, já no Scrum o desenvolvedor passou a ter mais contato com o cliente. Este novo relacionamento entre o desenvolvedor e a pessoa de negócio retira o intermediário que havia entre o cliente e o desenvolvedor, que possibilita que as solicitações do cliente possam ser atendidas com mais eficiência.

5.2 Algumas desvantagens consequentes da utilização do Scrum

Apesar da metodologia Scrum trazer consigo muitos benefícios, também apresenta algumas desvantagens que afetam a empresa de desenvolvimento e também as empresas-clientes.

5.2.1 Ausência ou escassez de documentação

As regras e características do Scrum não ignoram a documentação, mas recomenda uma documentação simplificada e objetiva, deixando o maior foco no funcionamento do *software*. Entretanto, deve-se tomar muito cuidado para que a documentação não seja deixada de lado completamente.

Na empresa S2D verificou-se que existe uma grande tendência na equipe de desenvolvimento de não documentar os fatos durante o desenvolvimento, o que causará transtornos e dificuldade no momento de dar manutenção ou fazer alguma melhoria no *software*. Desta forma, é muito importante que o *Scrum Master* analise com frequência o andamento da documentação do projeto e cobre da equipe uma documentação simplificada e objetiva do *software*.

5.2.2 Dificuldade em definir prazos para projetos

As desvantagens de não ser possível dar um prazo exato para a entrega do projeto, é totalmente prejudicial ao cliente. Esta desvantagem ocorre devido ao fato de que o cliente apenas descreve as funcionalidades do sistema de forma superficial no planejamento inicial, e somente posteriormente, no decorrer do projeto, estas funcionalidades serão refinadas, não viabilizando prever uma data precisa da entrega do projeto que atingirá imediatamente ao cliente. Contudo, essa é uma

desvantagem que de fato ocorre, mas é necessário ponderar, pois há outras vantagens que se apresentam ao cliente, tal qual a possibilidade do cliente alterar as funcionalidades do *software* a qualquer momento.

5.2.3 Equipe de desenvolvimento não tem funções definidas

Com a aplicação do Scrum, a S2D notou que os desenvolvedores passaram a não ter apenas uma funcionalidade específica dentro do projeto, porque no Scrum os integrantes passam a ter mais funções e responsabilidades. Por outro lado, este acúmulo de funções e responsabilidades pôde ser visto como um ponto negativo, pois quando um desenvolvedor é especialista em uma determinada área, este teoricamente tem amplo conhecimento e por consequência, desenvolve mais rápido, trazendo maior produtividade.

Em contrapartida, essa desvantagem acaba se convertendo em vantagem, pois se verificou na S2D que o fato dos desenvolvedores terem um conhecimento amplo de todas as áreas há um melhor resultado, quando visto em longo prazo comparado a metodologia tradicional.

5.3 Conclusão dos resultados obtidos

Na empresa S2D ficou muito evidente que as alterações no processo de desenvolvimento mudaram de forma significativa a dinâmica da empresa, e como de costume, todas as mudanças trazem consigo vantagens e desvantagens.

Observando as desvantagens, pode-se perceber que apesar de serem desvantagens, elas acabam sendo vantagens, trazendo pontos de melhoras no desenvolvimento do projeto. Por outro lado, existem as diversas vantagens que fazem do Scrum uma metodologia revolucionária, pois consegue cumprir tudo o que seus valores prometem, além de sobressair às desvantagens. Na empresa S2D ficou bem claro que os benefícios da Metodologia Ágil Scrum foram bem compensadores.

6 CONSIDERAÇÕES FINAIS

Com os conceitos apresentados, desde o surgimento do Manifesto Ágil até todas as características e regras da Metodologia Ágil Scrum, foi possível elucidar a importância de uma nova metodologia no processo de desenvolvimento de *software* na atualidade.

O Manifesto Ágil possui valores e princípios que sugerem um trabalho em equipe e de muita colaboração, a existência de uma boa comunicação entre o cliente e os desenvolvedores, a busca da excelência técnica e sanar as necessidades do sistema de informação do cliente com mais agilidade. E como a Metodologia Ágil Scrum é baseada no Manifesto Ágil, por meio desta monografia, pôde ser verificado que a grande promessa do Scrum, que é entregar um produto final de valor ao cliente e com agilidade é realmente cumprida.

O mercado de empresas desenvolvedoras de software é competitivo e o grande diferencial são as empresas que tem o foco no cliente. Logo, evidenciar o cliente faz com que ele se sinta importante, haja vista que o mesmo interage diretamente no desenvolvimento, palpitando maneiras para que seu produto final fique o mais próximo do esperado. Isso o torna um cliente participativo, bem como uma peça fundamental no desenvolvimento de seu software.

Para mostrar a aplicação do Scrum em uma empresa de desenvolvimento de *software* foi realizado um estudo de caso mostrando detalhadamente todos os processos que envolvem a utilização do Scrum. Com este estudo de caso pode-se verificar que as mudanças causadas pelo Scrum são bem evidentes e consequentemente podem-se analisar muitas vantagens e como em qualquer mudança também foram notadas algumas desvantagens.

Como uma de suas vantagens pode-se ser verificado que a Metodologia Ágil Scrum mostrou estar preparada para o mundo real dos negócios, onde nem sempre é possível estabelecer todos os requisitos no início do projeto, dando ao cliente a possibilidade de alterar e acrescentar funcionalidades no decorrer do projeto sem maiores problemas, pois o Scrum foi desenvolvido com o propósito de haver muitas mudanças no transcorrer do desenvolvimento do projeto.

Outro fator relevante é o salto de qualidade que a Metodologia Ágil Scrum propicia aos desenvolvedores, pois eles ganham mais responsabilidade e autonomia para exercer suas funções, que na grande maioria, são muito variadas. Os desenvolvedores, a partir do Scrum, também sabem o que cada integrante da equipe está desenvolvendo o que facilita o compartilhamento da base de conhecimento entre a equipe.

Porém para as empresas de desenvolvimento de *software* um dos fatores mais importante é o aumento da produtividade e o Scrum tem como uma de suas principais prioridade aumentar o desempenho da produtividade de desenvolvimento de *software,* uma vez que o processo de desenvolvimento fica mais simples, objetivos e eficiente, dando mais importância para as pessoas envolvidas no projeto e deixando de lado processos desnecessários que não contribuem efetivamente para a entrega do produto final para o cliente.

Em contrapartida, existem algumas desvantagens que surgem durante todo o processo, tal qual a documentação mínima que é registrada pelos desenvolvedores, uma vez que a documentação não é tratada com prioridade no processo de desenvolvimento do software. Nota-se que devido à ocorrência de tantas mudanças no planejamento do desenvolvimento do software, subentende-se que documentar passo-a-passo dos procedimentos é perda de tempo, então há apenas uma documentação minimamente necessária, entretanto deve ter muito cuidado para que a documentação não seja completamente ignorada, causando problemas em manutenções futuras. Cabe ao Scrum Master e aos gestores exigirem uma documentação simples e objetiva para ganhar em produtividade, porém sem perder a qualidade do produto final.

É válido ressaltar que devido ao Scrum possibilitar o cliente fazer mudanças constantes dos requisitos e funcionalidades durante o projeto, a estimativa do prazo de entrega do produto final fica mais difícil de ser realizada, uma vez que depende muito das mudanças e do andamento do projeto, entretanto essa é uma desvantagem que para o cliente deve ser visto pelo lado bom, já que o produto que será entregue ao cliente com mais qualidade e mais próximos do que o próprio cliente gostaria de receber.

Contudo, analisando os resultados obtidos com a utilização da metodologia ágil Scrum é possível perceber que mesmo havendo desvantagens os benefícios trazidos são mais compensadores e possibilitará que a empresa de desenvolvimento de software consiga entregar um produto final com mais valor, mais próximo do que o cliente gostaria e com mais um menor prazo de entrega.

Todavia é necessário que todas as pessoas envolvidas com o projeto, desde a equipe de desenvolvimento, o Scrum Master e o próprio cliente, devem sempre respeitar seus papéis, responsabilidades, regras e características dentro do Scrum, para que possam ser extraídos todos os benefícios de gerenciamento, propiciado pela Metodologia Ágil Scrum.

7 REFERÊNCIAS BIBLIOGRÁFICAS

AMBLER, S. W. Modelagem ágil: práticas eficazes para a programação extrema e o processo unificado. Bookman, 2004.

BALTIZAN, P; PHILLIPS, A. Sistema de informação. ed. 1. Rio Grande do Sul: AMGH Editora Ltda., 2012 (p. 284 - 299).

BECK, K; Programação extrema (XP) explicada. Ed. 1. Rio Grande do Sul: Bookman, 2004 (p. 123 - 160)

BRAZIP SCRUM. Disponível em: < http://www.brazip.com.br/brazipscrum/ >. Acesso em: 13/04/2013.

HIGHSMITH, J. Agilidade em gerencia de projetos Scrum. 2. Ed. Rio de Janeiro: Campus/Elsevier, 2004 (p. 170-215).

KNIBERG, H. Scrum e XP direto das trincheiras: como nós fazemos Scrum. C4 Media, 2007.

MANIFESTO ÁGIL. Disponível em: <http://agilemanifesto.org/>. Acesso em: 25/04/2013.

MARTINS, J. C. C. Técnica para gerenciamento de projetos de software. ed. 1. Rio de Janeiro: Brasport, 2007.

ORTH, A. I; PRIKLADNICKI, R. Planejamento e Gerencia de Projetos. ed. 1. Rio Grande do Sul: Edipucrs, 2009 (p. 149 – 168).

PEREIRA P; TORREÃO P, MARÇAL A S. Entendendo Scrum para Gerenciar Projetos de Forma Ágil. MundoPM; 2007. Disponível em <http://www.siq.com.br/DOCS/EntendendoScrumparaGerenciarProjetosdeFormaAgil.pdf>. Acesso em: 15/04/2013.

PICHLER, R. Gestão de Produtos com Scrum: Implementando Métodos Ágeis na Criação e Desenvolvimento de Produtos. Ed. 1. Rio de Janeiro: Elsevier Editora Ltda., 2010.

PURI, C. P; Agile Management Feature Driven Development. 1. ed. Global India Publications Pvt Ltd, 2004 (p. 3-18).

SCRUM HALF. Disponível em: <http://www.myscrumhalf.com/pt/>. Acesso em: 10/05/2013.

SCRUMRF. Disponível em: < http://scrumrf.com/>. Acesso em: 10/03/2013.

SCRUMWISE. Disponível em: <http://www.scrumwise.com/>. Acesso em: 10/05/2013.

STAPLETON, J; Dynamic Systems Development Method. 2. ed. Boston: Addison - Wesley Professional, 2003 (p. 4-55).

SUTHERLAND, J; SCHWABER, K. SCRUM. Disponível em: < http://scrum.org/>. Acesso em: 15/04/2013.

S2D. Disponível em: < http://www.s2d.com.br/>. Acesso em: 20/03/2013.