UNIVERSIDADE ESTADUAL DO NORTE DO PARANÁ

***CAMPUS* LUIZ MENEGHEL - CENTRO DE CIÊNCIAS TECNOLÓGICAS**

BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ERICK CESAR SANCHES KLEIM

UM ESTUDO DO SCRUM EM UM PROJETO COM UMA EQUIPE PEQUENA

Bandeirantes - PR

2016

ERICK CESAR SANCHES KLEIM

UM ESTUDO DO SCRUM EM UM PROJETO COM UMA EQUIPE PEQUENA

Trabalho de Conclusão de Curso submetido à Universidade Estadual do Norte do Paraná, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Prof. Me. Thiago Adriano Coleti
Coorientador: Prof. Dr. Maurício Massaru Arimoto

Bandeirantes - PR
2016

ERICK CESAR SANCHES KLEIM

UM ESTUDO DO SCRUM EM UM PROJETO COM UMA EQUIPE PEQUENA

Trabalho de Conclusão de Curso submetido à Universidade Estadual do Norte do Paraná, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

COMISSÃO EXAMINADORA

_____
Prof. Me. Thiago Adriano Coleti
UENP – *Campus* Luiz Meneghel

_____
Prof. Dr. Maurício Massaru Arimoto
UENP – *Campus* Luiz Meneghel

_____
Prof. Dr. André Luís Andrade Menolli
UENP – Campus Luiz Meneghel

Bandeirantes, __ de_____ de 2016

RESUMO

Existem numerosos relatos sobre aplicações de métodos ágeis em empresas, estes apontam melhorias na comunicação, prazo, qualidade, métricas e motivação dos membros envolvidos. Contudo, estes relatos são, em sua maioria, relacionados ao estudo da implementação destes métodos em empresas de médio e grande porte. Observou-se então um deficit de produções no ambiente de empresas com escopo reduzido, e ainda identificou-se que estes métodos poderiam ser melhor configurados para se adequar às características destas empresas. Este trabalho consiste de um estudo sobre a utilização do método ágil *Scrum* em uma pequena equipe da empresa júnior pertencente a Universidade Estadual do Norte do Paraná em Bandeirantes, durante um projeto real de desenvolvimento de software. Esta empresa trabalha com carga horária reduzida e quantidade de membros flutuante. Durante o estudo de caso, a coleta de evidências deste trabalho foi realizada pela técnica de observação participativa e entrevistas, apontando complicações no comportamento do Scrum como: excesso de papéis e falta de necessidade na utilização das reuniões diárias do método *Scrum* tradicional para o porte desta equipe. Por meio da análise dos dados coletados foi possível elaborar uma proposta de customização do método *Scrum*, com o objetivo de mitigar os pontos negativos, como: falta de recursos humanos para completar os cargos, sobrecarga de tarefas e falta de necessariedade da reunião diária, levantados por esta análise. O objetivo deste estudo de casso foi levantar as problemáticas da utilização do *Scrum* tradicional em equipes deste porte e então propor uma solução, elaborando uma abordagem alternativa para este método, para em trabalhos futuros comprovar sua validade.

Palavras-chave: *Scrum*. Microempresa. Abordagem. Engenharia de Software. Gestão de projeto.

ABSTRACT

There are numerous reports on applications of agile methods in companies, these indicate improvements in communication, timing, quality, metrics and motivation of the members involved. However, these reports are mostly related to the study of the implementation of these agile methods in medium and large companies. A production shortfall was observed in the environment of companies with reduced scope, and it was also identified that these methods could be better configured to suit the characteristics of these companies. This work consists of a study on the use of the agile Scrum method in a team of three members of the junior company belonging to the State University of Northern Paraná in Bandeirantes, during a real software development project. This company works with reduced working hours and number of floating members. During the case study, the evidence collection of this work was accomplished through the technique of participatory observation and interviews, pointing out complications such as: excess of roles and lack of necessity in the use of events of the traditional method for the size of this team. Through the analysis of the collected data it was possible to elaborate a proposal of a new approach of the Scrum method, with the attempt to mitigate the negative points raised by this analysis. The objective of this study was to raise the problems of using traditional Scrum in teams of this size and then propose a solution, elaborating an alternative approach for this method, in future work to prove its validity.

Keywords: Scrum. Micro enterprise. Approach. Software Engineering. Project management.

LISTA DE SIGLAS

FDD	*Feature Driven Development*
MSF	*Microsoft Solutions Framework*
MPE	Micro e Pequena Empresa
PSP	*Personal Software Process*
RUP	*Rational Unified Process*
XP	*Extreme Programming*

SUMÁRIO

1. INTRODUÇÃO

O desenvolvimento de software se tornou um dos mais importantes setores da economia dos tempos modernos, e fornece produtos fundamentais ao nosso cotidiano (FONSECA, 2008).

Para se manter no mercado, as empresas devem oferecer um diferencial. Um modo de se destacar é no processo de desenvolvimento, entregando com rapidez algo de valor ao cliente (FONSECA, 2008). Em consequência disso, aumentou-se a importância das empresas de desenvolvimento no cenário nacional, intensificando a competitividade.

Ainda sobre a evolução da indústria de software, Berni (2010) afirma que existe uma crescente pressão competitiva no âmbito empresarial e a relação cliente-empresa exerce grande influência sobre ele. Isto indica que para permanecer no mercado empresas necessitam inovar nos processos. No cenário atual, desenvolvedores buscam por entregar ao cliente pequenos incrementos com qualidade em um período curto de tempo, com o objetivo de deixar o cliente seguro (SOUZA, 2014). Segundo Coelho (2011), uma forma de assegurar a satisfação do cliente e da equipe envolvida é utilizar ferramentas, técnicas e métodos, tendo em vista a melhoria de processos organizacionais e a gerência no desenvolvimento de software.

Tanto as empresas maiores quanto menores têm a necessidade de satisfazer o cliente entregando produtos de qualidade. Porém no âmbito das microempresas as necessidades diferem das demais organizações produtoras de software, pois grande quantidade de especificações de processos e burocracias podem afetar a produtividade pelo desvio de foco no desenvolvimento e resultar em fracasso do projeto (BERNI, 2010).

As afirmações anteriores refletem aos valores do manifesto ágil[1] de desenvolvimento de software: interação entre indivíduos, software em funcionamento, colaborar com o cliente e responder a mudanças.

[1] http://www.agilemanifesto.org

Estes fundamentos estão mais próximos da maneira de trabalho das micro e pequenas empresas (SOARES, 2004). Porém, no manifesto ágil ainda existem formalidades, eventos papeis e artefatos que podem ser tratados de uma outra maneira ou até mesmo desconsiderados, dependendo da característica de determinada equipe como, por exemplo, a quantidade de pessoas pertencentes a ela.

No contexto deste trabalho observa-se a necessidade de classificar equipes pelo seu tamanho. Schwaber (1997) cita que uma equipe pequena tem entre três a seis membros, enquanto Souza (2007) considera uma equipe pequena com até dez membros. Sendo assim, neste trabalho é abordado equipes de dois a cinco membros, consideradas pequenas. Salientando que o estudo de caso foi feito com uma equipe de três membros.

As afirmações feitas nesta seção revelam que a gestão no processo de desenvolvimento é crucial para a qualidade do software e qualidade é quesito incisivo na competitividade entre empresas não importando seu tamanho. Entretanto, Carvalho e Mello (2012) afirmam que ainda existe uma baixa quantidade de pesquisas sobre os processos de desenvolvimento de software direcionadas a micro e pequenas empresas de desenvolvimento. Fonseca, (2008) também dificuldade para se encontrar métodos de gerência de projetos que se encaixem em equipes pequenas de desenvolvimento de software.

Segundo Thiry *et al.* (2006) as micro e pequenas empresas empregam métodos de trabalho informais e com recursos escassos. No presente trabalho é dado enfoque ao recurso humano reduzido, dentro de pequenas equipes nas empresas.

Este trabalho apresenta uma adaptação de um conhecido método do manifesto ágil chamado *Scrum*. O *Scrum* funciona de forma iterativa e incremental, utilizando-se de ciclos de desenvolvimento, conhecidos como *Sprint*, para que ao final de cada *Sprint* seja entregue partes do todo, e estas partes estejam finalizadas e testadas. Estas partes são construídas por ordem de prioridade, sendo definida por um membro da equipe que interage diretamente com o cliente, tem a visão do produto como um todo e conhece os interesses do cliente (PAGOTTO *et al.* 2016).

Apesar do *Scrum* ser um método ágil, existem pontos que podem ser modificados para aumentar a dinamicidade do desenvolvimento, como por exemplo a facilidade de comunicação entre membros de uma equipe deste porte, não necessitando de formalidades tais como, as reuniões diárias fixas as quais poderiam ser utilizadas por equipes maiores.

Diante do exposto, um estudo de caso foi conduzido envolvendo uma pequena equipe, composta por três integrantes, pertencente a empresa júnior da Universidade Estadual do Norte do Paraná em Bandeirantes (UENP–CLM).

Para medir os efeitos da utilização do método *Scrum* tradicional[2] nesta equipe foi utilizada a técnica de observação participante e entrevistas com os membros da equipe foco do estudo. Os pontos positivos e negativos levantados durante a implantação ressaltam a necessidade de uma abordagem alternativa à tradicional. Como principal resultado deste trabalho, uma abordagem do método *Scrum* para equipes pequenas foi definida. Esta abordagem deve ser validada por trabalhos futuros e espera-se atingir os objetivos do manifesto ágil, na qual Da Silva, Pires e Neto (2015) apontam que são: qualidade no produto final, entrega de projeto dentro do prazo, flexibilidade a mudanças, transparência no projeto, redução de riscos e satisfação do cliente.

1.1 OBJETIVOS

Nesta seção do trabalho será apresentado o objetivo geral e objetivos específicos.

1.1.1 Objetivo Geral

O objetivo deste trabalho é propor uma adequação do método *Scrum* tradicional para pequenas equipes de desenvolvimento de software.

1.1.2 Objetivos Específicos

Com o propósito de alcançar o objetivo geral, objetivos específicos foram identificados como importantes, dentre eles:

- Realizar um estudo de caso envolvendo a aplicação do Scrum no desenvolvimento de um projeto de software, dentro do contexto de uma pequena empresa;

[2] A nomenclatura Scrum tradicional é utilizado durante este trabalho para referir-se ao método ágil Scrum conforme definido por Schwaber e Sutherland (2013).

- Analisar o comportamento do *Scrum* tradicional em uma pequena equipe; e
- Propor uma abordagem alternativa mediante os resultados da análise dos dados coletados.

1.2 JUSTIFICATIVA

Vários autores como Coelho (2011), Thiry *et al.* (2006) e Berni (2009) relatam que no cenário de desenvolvimento de software, em que melhorias são claramente necessárias, não importando o tamanho da equipe ou empresa, um caminho para se alcançar maior qualidade do software visando a satisfação do cliente é modificando ou implementando uma forma de gerência de projeto específica para cada contexto.

Este trabalho justifica-se pela necessidade de suprir a dificuldade em encontrar método para gestão de projetos de desenvolvimento de software no âmbito de equipes pequenas. A indústria de software tem crescido, e melhorar os processos de desenvolvimento é um meio de se manter competitivo neste mercado (Fonseca, 2008). Ainda sobre a carência de abordagens que tratem deste escopo de empresas, Thiry *et al*. (2006) observa a falta de pesquisas na área.

No anuário do Sebrae (2013) sobre micro e pequenas empresas, é descrito como “microempresas” no setor de comercio e serviço aquelas que possuem menos de nove pessoas ocupadas. Métodos de gerência de projetos usualmente trabalham com equipes com uma quantidade semelhante ao total de membros de uma equipe deste porte. Neste contexto, para se utilizar um método de gerência de projetos, todo o corpo da empresa estaria envolvido em um único projeto, diminuindo a competitividade da mesma. No caso da empresa de desenvolvimento situada na UENP-CLM composta pelos alunos do Centro de Ciências Tecnológicas (CCT), ela oscila entre micro e pequena empresa (SEBRAE, 2013), e enfrenta dificuldades no momento de gestão de projetos nos seguintes pontos: nos métodos tradicionais existem documentações extensas e nos métodos ágeis uma grande quantidade de papeis dentro da equipe e um alto número de reuniões para assegurar transparência. Isto ocorre pela dificuldade de encontrar um método adequado a este tipo de organização.

Diante das dificuldades supracitadas este trabalho foca em uma abordagem para pequenas equipes, com número de documentação, papeis e reuniões reduzidos, porém assegurando com que as mesmas usufruam das vantagens do manifesto ágil.

1.3 ORGANIZAÇÃO DO TRABALHO

Este trabalho está disposto conforme a seguinte formatação. No Capítulo 2 são abordados os tópicos sobre métodos ágeis, o método ágil Scrum e trabalhos com finalidades correlatas. No Capítulo 3 é apresentado o tipo de pesquisa adotado neste trabalho e os processos metodológicos. No Capítulo 4 são descritas as características do ambiente em que foi aplicada a pesquisa, o desenvolvimento da pesquisa propriamente dito e o levantamento dos problemas encontrados. No Capítulo 5 são apresentadas as propostas de melhorias para uma abordagem do método *Scrum* em pequenas equipes. Por fim, no Capítulo 6 é apresentada uma síntese do problema abordado, do desenvolvimento do trabalho e dos resultados obtidos.

2. FUNDAMENTAÇÃO TEÓRICA

Este capítulo está dividido em 5 seções, a fim de contribuir ao tema central do trabalho. A Seção 2.1 é abordado o manifesto ágil, com a devida descrição dos 12 princípios do manifesto ágil. Na Seção 2.2 são apontados os principais aspectos e conceitos a respeito do método ágil Scrum. Finalizando este capitulo a Seção 2.3 contém es estratégias e objetivos de trabalhos que atuam em problemáticas similares a problemáticas deste trabalho.

2.1 MÉTODOS ÁGEIS

Nesta seção discutem-se algumas características fundamentais dos métodos ágeis, com enfoque no *Scrum*, que serviu de fundamentação para o desenvolvimento deste trabalho. O termo método ágil foi esclarecido no ano de 2001, quando alguns especialistas em processo de desenvolvimento de software se reuniram e analisaram algumas dos métodos em ascensão na época. Com base nesta análise conclui-se que estes novos métodos seguiam uma norma, a partir de então foram criados os princípios do manifesto para desenvolvimento ágil (BRUNHERA; ZANATTA, 2010).

Em síntese, o foco do movimento ágil são os indivíduos e suas interações, produto funcional, colaboração com o cliente e resposta rápida a mudanças (SABBAGH, 2014).

O movimento ágil segue doze princípios, conforme descrição a seguir (SABBAGH, 2014):

1. **Nossa maior prioridade é satisfazer o cliente por meio da entrega cedo e frequente de software com valor**: o foco no desenvolvimento é satisfazer o cliente desde cedo, gerando retorno do investimento feito por ele por meio de entregas de partes que atendam suas necessidades;
2. **Mudanças de requisitos são bem-vindas, mesmo em fases tardias do desenvolvimento. Os processos ágeis utilizam a mudança em favor da vantagem competitiva para o cliente**: aceitar as mudanças para melhor atender as necessidades do cliente e ciclos curtos permitem a evolução do produto à medida que melhora o entendimento da necessidade;

3. **Entregar software funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos**: este princípio vai na contramão de entregas únicas ou mínimas. Entregas constantes permitem a obtenção de *feedback,* possibilitando adaptação e diminuição de riscos;
4. **As pessoas do negócio e os desenvolvedores devem trabalhar em conjunto diariamente ao longo do projeto**: sustenta a cooperação das pessoas com o mesmo objetivo, gerar algo de valor aos clientes;
5. **Construa projetos em torno de indivíduos motivados. Dê-lhes o ambiente e o suporte que precisam e confie neles para realizarem o trabalho**: o ambiente com suporte e confiança é fundamental para que pessoas estejam motivadas para construir um produto;
6. **O método mais eficiente e efetivo de se transmitir informação para e entre uma equipe de desenvolvimento é a conversa face a face**: a linguagem corporal, olhar e entonação de voz enriquecem a comunicação. Quando não é possível comunicação presencial, deve se fazer uso das tecnologias disponíveis para se aproximar a este dialogo. Este princípio se opõe a documentos como forma de comunicação;
7. **Software em funcionamento é a principal medida de progresso**: o melhor método de medição de progresso é entrega de partes de maior valor ao cliente, ou seja, entrega de software funcionando;
8. **Os processos ágeis promovem o desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter indefinidamente um ritmo constante**: garantir constância no trabalho dos desenvolvedores do produto pelos envolvidos no projeto, sem sobrecarregá-los, para não gerar insatisfação e consequentemente diminuição da qualidade;
9. **A atenção contínua à excelência técnica e a um bom projeto aumentam a agilidade**: excelência na técnica e no projeto do produto facilitam no momento de modificações sem perda de qualidade e agilidade, tornando o projeto flexível e passível de alterações;
10. **Simplicidade - a arte de se maximizar a quantidade de trabalho**

não feito – é essencial: não realizar trabalho sem necessidade, evitar o desperdício desenvolvendo funcionalidades que o cliente não precisa;

11. **As melhores arquiteturas, requisitos e projetos emergem de equipes que se auto organizam**: equipes que se auto organizam tem objetivos e metas em comum, podendo decidir o melhor modo de executar as tarefas, se tornando mais eficientes;
12. **Em intervalos de tempo regulares, a equipe reflete sobre como se tornar mais efetiva e então refina e ajusta seu comportamento de acordo**: promove inspeção continua para serem cada vez mais eficientes, promove adaptação no processo de trabalho.

2.2 Scrum

Nesta seção é realizada uma descrição geral do método *Scrum*, seus papeis, eventos e artefatos, constituindo a base necessária para o desenvolvimento deste trabalho. Este método preza por uma equipe reduzida, gastando um pequeno tempo e construindo uma pequena parte do sistema, porém integrando-as com regularidade para ao final do processo compor um todo (KINBERG; SKARIN, 2009).

Schwaber e Sutherland (2013) descrevem-na como um framework em que pessoas tratam e resolvem situações de alta complexidade por meio de entregas constantes de alto valor. O método é intitulado como leve, simples de entender e tem extrema dificuldade de ser dominada, ou seja, dificuldade para executar com total conformidade as regras do *Scrum*.

O método emprega o desenvolvimento de uma forma incremental e iterativa, como pode ser visto na Figura 1. Outra característica refere-se à equipe com papeis bem definidos e com regras para seguir, utilizando-se de artefatos e eventos para solucionar os problemas (SCHWABER; SUTHERLAND, 2013).

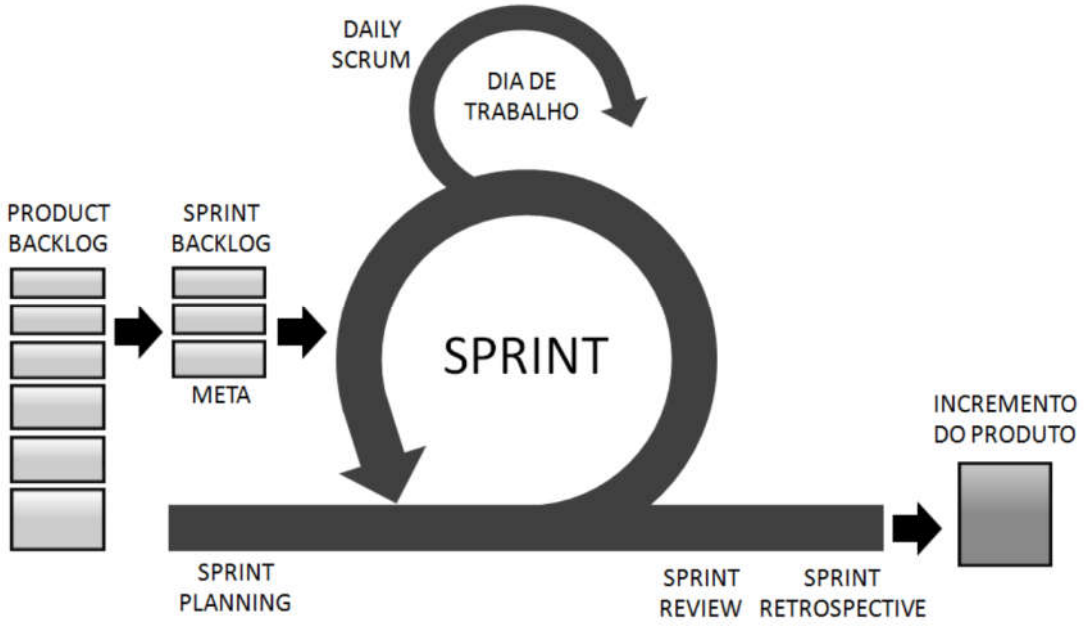

Figura 1 - Ciclo de vida *Scrum* (Adaptado de SABBAGH, 2014)

O *Scrum* segue uma estrutura para gerir os projetos com a finalidade de proporcionar três pilares: transparência, adaptabilidade e inspeção, e são descritos por Schwaber e Sutherland (2013) em sequência:

- A transparência garante que os desenvolvedores tenham a mesma visão comum sobre o projeto. Também remete aos responsáveis ter uma visão sobre o andamento e cronograma do projeto;
- A inspeção é simplesmente observar periodicamente os artefatos e requisitos, com a intenção de monitorar o progresso e observar se está mais próximo do objetivo;
- Adaptação é a possibilidade de o projeto ser reestruturado e modificado conforme a visão do cliente sobre as suas necessidades reais.

Outro fator importante a ser ressaltado é que, assim como os outros métodos ágeis, o *Scrum* também é flexível, podendo ser adaptado a vários tamanhos de equipes e períodos de iterações mutáveis, de acordo com a necessidade do projeto ou característica da equipe. Todas estas alterações são de responsabilidade do *Scrum* Master; este deve ter a sensibilidade de perceber o que caberá melhor no estado em que se encontra. O *Scrum* possui algumas características importantes que são descritas nos itens a seguir.

1. Papéis e Responsabilidades

O time do *Scrum* é dividido em três papeis, *Product Owner, Scrum Master*, e Equipe de Desenvolvimento (DA SILVA; DE MELLO JUNIOR, 2014). Neste trabalho quando necessário mencionar todos os participantes do projeto será utilizado o termo "time *Scrum*".

Product Owner: O dono do produto é responsável por entender o desejo do cliente, ouvir as diversas pessoas a serem afetadas pelo projeto, sintetizar os requisitos e montar de forma clara e objetiva (não detalhada) o *Product Backlog*. Tal artefato é atualizado constantemente pelo *Product Owner*, permitindo que qualquer membro do time tenha uma única visão do produto (SABBAGH, 2014). O *Product Owner* também tem como atribuição aceitar e rejeitar as produções de cada *Sprint* (DA SILVA; PIRES; CARVALHO NETO, 2015).

Antes do início de uma nova *Sprint*, deve estudar as histórias que estão mais ao topo do *Product Backlog*, histórias candidatas a um lugar no *Sprint Backlog*, para que durante a reunião de planejamento da *Sprint* possa juntamente com a equipe de desenvolvimento discuti-las e então detalhar estas historias para o *Sprint Backlog*, ou até mesmo quebrar em tarefas menores e entrar em acordo com os desenvolvedores do time (SHUTHERLAND; COHN, 2007).

Ele é o único que tem permissão de fazer qualquer tipo de alteração no *Product Backlog*, porem todos podem contribuir, tanto como redefinir prioridade de histórias, inserir ou retirar histórias (SCHWABER; SUTHERLAND, 2013).

Equipe de desenvolvimento: Sua principal função é desenvolver o que foi estabelecido durante a reunião de planejamento (DA SILVA; PIRES; CARVALHO NETO, 2015).

O time é quem decide, juntamente com o *Product Owner*, o quanto de trabalho pode ser posto em desenvolvimento na reunião de planejamento da *Sprint*. Uma das características da equipe de desenvolvimento é ser multidisciplinar, podendo ter especialistas na equipe, porém todos devem ser capazes de executar as tarefas associadas. Outra característica é ser auto-organizada (BRUNHERA; ZANATTA, 2010), isso implica que uma vez aceito o objetivo da *Sprint* e determinado as histórias do produto a serem entregues, é obrigação da equipe de desenvolvimento se organizar, dividindo as tarefas como julgarem melhor. Quanto ao número de membros do time de desenvolvimento, ele deve ser pequeno suficiente para ser auto organizado e capaz de produzir incrementos de valor ao final da *Sprint*

(SHWABER; SUTHERLAND, 2013). A qualidade do produto é competência dos desenvolvedores assim como reportar os empecilhos ao *Scrum* Master para que o mesmo resolva o obstáculo (SABBAGH, 2014).

Scrum Master: Geralmente o *Scrum Master* tem habilidades de unir o grupo e também de liderança. Com já mencionado anteriormente, uma das obrigações deste membro do time *Scrum* é remover contratempos que impeçam a equipe de desenvolvimento a entregar algo de valor ao final da *Sprint*. Também serve como escudo para os desenvolvedores, barrando quaisquer intervenções externas que possam atrapalhar o rendimento. Também atua como mediador entre *Product Owner* e Equipe de Desenvolvimento, sendo um facilitador de negociações durante as reuniões de planejamento. É incumbido a ele o papel de ensinar o uso correto do *Scrum* ao time, assim como verificar constantemente se as regras do método estão sendo cumpridas (SABBAGH, 2014).

2. Eventos

Embora existam rumores sobre a desorganização e falta de planejamento em métodos ágeis, no *Scrum* existem quatro reuniões estruturadas em seu método, sendo uma exclusiva para planejamento, Reunião de Planejamento (*Planning Meeting*), outra para planejamento e organização de tarefas, Reunião Diária (*Daily Scrum*), e uma para revisão, Reunião de Retrospectiva (*Retrospective*), em que se tem uma visão de como melhorar e assim poder aperfeiçoar o planejamento e organização continuamente. Além destas cerimônias existe uma reunião para apresentação de resultados ao cliente, *Review Meeting*, e um evento que dura de duas a quatro semanas, o período de desenvolvimento, chamado de *Sprint*. Estes eventos são descritos abaixo.

Sprint: O objetivo deste evento é ao final entregar uma parte de produto utilizável que fora decidido na reunião de planejamento. Schwaber e Sutherland (2013) aconselha que tenha duração de no mínimo duas semanas e no máximo quatro.

Para se definir o "*time-box*", na tradução literal significa "caixa de tempo", trazendo para o *Scrum* isto reflete um limite de tempo para serem cumpridas as tarefas, isto é, duração que as todas as *Sprints* devem obedecer, deve ser considerada as características da equipe. Equipes que precisam ser constantemente motivadas devem seguir pelo caminho de *Sprints* menores, fazendo com que entreguem mais constantemente produtos de valor, motivando a equipe a cada

entrega.

Pelo *Scrum* ter a característica de ser iterativo, vale ressaltar que durante um projeto ocorrerão diversas *Sprints* (SABBAGH, 2014), a cada final será integrado mais um pedaço de produto funcional ao todo (BRUNHERA; ZANATTA ,2010), o tornando também incremental.

Todas as *Sprints* devem começar logo após o encerramento da reunião de planejamento, ter reuniões diárias para acompanhamento, uma reunião de apresentação de trabalho realizado e para finalizar uma reunião para tomar nota dos pontos positivos e negativos da *Sprint* finalizada (DA SILVA; PIRES; CARVALHO NETO, 2015).

Cada *Sprint* recebe uma meta/objetivo, caso por alguma razão extraordinária esta meta/objetivo não possa ser completa por inteiro, a mesma é abortada, antecipando as reuniões de fim de *Sprint* e agilizando para iniciar uma nova (SABBAGH, 2014).

Planning Meeting: Primeiro passo de um *Sprint*, nesta reunião será definido o *Sprint Backlog*, lista de tarefas a serem cumpridas durante a *Sprint* pela Equipe de Desenvolvedores. Todos os membros do time *Scrum* devem participar, o *Product Owner* deve se preparar para a reunião, trazendo detalhadamente os itens de maior prioridade do *Product Backlog*, para que todo o time *Scrum* construa o *Sprint Backlog* (DA SILVA; PIRES; CARVALHO NETO, 2015). Aconselha-se que esta reunião dure no máximo oito horas para *Sprints* de 4 semanas, e menos tempo para *Sprints* menores (SCHWABER; SUTHERLAND, 2013). O *Scrum* Master deve garantir que este evento ocorra e que não ultrapasse este espaço de tempo.

Daily Scrum: Esta é uma reunião rápida em que o time aborda questões do trabalho, expondo o que foi feito, e o que pretende fazer até a próxima reunião. O *Scrum* Master tem a tarefa de cuidar para que a reunião não passe de quinze minutos (BRUNHERA; ZANATTA, 2010).

A reunião é embasada em três perguntas: o que foi feito ontem; o que será feito hoje; e quais são os impedimentos para a realização do trabalho. O *Scrum* Master deve garantir que apenas a equipe de desenvolvimento participe deste evento, pois a terceira pergunta é feita para que a equipe de desenvolvimento se auto gerencie e inspecione o progresso do projeto em direção ao objetivo da *Sprint* (SHWABER; SUTHERLAND, 2013). Esta é uma reunião curta, então não tem tempo para apontar erros e falhas, este é um tópico para a reunião de retrospectiva (DA

SILVA; PIRES; CARVALHO NETO, 2015).

Sprint Review: No último dia da *Sprint* a equipe de desenvolvimento apresenta o que foi feito durante todo o período de desenvolvimento para as partes interessadas, os *Stakeholders*. Nesta apresentação são identificados os itens que estão “prontos” segundo o a definição de pronto (DoD) da equipe. A definição de pronto, é um consenso da equipe sobre em quais testes o incremento deve ser aprovado para ser considerado como pronto. Ao final da reunião, o *Product Backlog* deve ser revisado com a presença dos *Stakeholders* para fazer quaisquer alterações caso que julgarem necessário (SHWABER; SUTHERLAND, 2013).

Sprint Retrospective: Nas últimas horas da *Sprint*, é feito toda uma análise do que foi feito, os processos, ferramentas utilizadas, relacionamentos, quaisquer pontos que deram certo e pontos que podem ser feitas melhorias (SHWABER; SUTHERLAND, 2013). O objetivo desta reunião é a melhoria continua da forma de trabalho do time *Scrum*. Para isto são discutidas quais as ações feitas na *Sprint* foram positivas ao projeto, e quais não. Com base nestas discussões, propostas são levantadas para serem implantadas já na da próxima *Sprint* (SABBAGH, 2014).

3. Artefatos

Schwaber e Sutherland (2013) definem como Artefatos do *Scrum* apenas quatro, *Product Backlog*, *Sprint Backlog* e Incremento do Produto, porém em várias outras literaturas como Silva e Júnior (2014), Kniberg (2013) e Sabbagh (2014), o *Burn Down Chart* também é inserido com um artefato. Uma vez que neste trabalho os projetos desenvolvidos com o método *Scrum* se utilizam deste mecanismo para medir o progresso, o mesmo será definido nesta seção.

Product Backlog: Lista de funcionalidades que devem estar ordenadas prioridade, determinada pelo *Product Owner* segundo a visão do cliente. O *Product Backlog* deve apresentar uma visão geral do produto para todo o time; ele não é um documento fixo, estando em constante mudança. O *Product Owner* é o responsável por inserir, retirar e repriorizar os itens; porém, todos os membros da equipe podem contribuir com este documento (DA SILVA; PIRES; CARVALHO NETO, 2015).

Sprint Backlog: Seleção dos itens do *Product Backlog* para ser cumprida durante a *Sprint* (DA SILVA; PIRES; CARVALHO NETO, 2015). Toda *Sprint* tem um objetivo, caso uma nova tarefa deva ser executada para atingir o este objetivo, a mesma deve entrar para esta lista de tarefas. O *Sprint Backlog* é definido na reunião de planejamento. Os itens desta lista devem estar bem detalhados e claros, sendo

possível que o time de desenvolvedores compreenda as tarefas e possam realizá-las a partir desta descrição (SABBAGH, 2014).

Incremento do Produto: O incremento é o resultado da produção durante a *Sprint* dos itens selecionados para o *Sprint Backlog*. Todos os itens que foram dados como “pronto” são apresentados e incrementados ao que já fora feito anteriormente. (SABBAGH, 2014). Exemplos de Incrementos do Produto são: documentos, pedaços de software, layout, dentre outros.

De acordo com da Silva e de Mello Junior (2014) as vantagens da utilização do *Scrum* como método de gerência de projeto são observadas durante todo o processo de desenvolvimento, destacando o acompanhamento minucioso, a participação do cliente durante todo o projeto, a comunicação da equipe pelas reuniões e a responsabilidade da equipe em cumprir os prazos. Reunindo estes diversos pontos positivos trazem a segurança para que o projeto atenda às necessidades do cliente.

2.3 TRABALHOS RELACIONADOS

Esta seção tem o objetivo de expor trabalhos com características semelhantes a este, relatando seus objetivos, quais estratégias abordaram para chegar ao objetivo e resultados alcançados.

2.3.1 *Scrum* Solo

Este trabalho, realizado por Pagotto *et al*. (2016), foi desenvolvido visando auxiliar no desenvolvimento de software por equipes de apenas uma pessoa, para que possa gerenciar o desenvolvimento de modo formal unindo as boas práticas descritas pelo *Personal Software Process* (PSP) e *Scrum*.

O Scrum Solo tem no geral uma estrutura similar ao *Scrum* tradicional proposto por Schwaber e Sutherland. Inicialmente faz se o levantamento dos requisitos, definição de escopo e estruturação do *Product Backlog* por um orientador, este é um conhecedor dos processos de software, juntamente com o cliente.

Definido a primeira versão do *Product Backlog* é iniciado o *Management* que tem como objetivo planejar e controlar o desenvolvimento, gerando a estrutura analítica do projeto, cronograma, planilha de custo e planilha de controle. Após esta

fase é executado a *Sprint*, com o tempo definido de uma semana, se diferenciando do *Scrum* tradicional que aconselha entre duas a quatro semanas de duração, gerando parte do produto. Ao entregar o produto é feita uma reunião de orientação fechando o ciclo. Este processo ocorre quantas vezes forem necessárias, de modo iterativo e incremental, até que seja entregue o produto final completo ao cliente.

Este processo fora utilizado por alunos do curso de Engenharia da Computação e Análise de Desenvolvimento de Sistemas da Universidade Tecnológica Federal do Paraná nos anos de 2012 a 2014 com sucesso no desenvolvimento. Ex-alunos desenvolvedores de software relataram que o processo contempla as necessidades de gestão de projetos.

Este trabalho tem a contribuir com desenvolvedores individuais, uma abordagem restrita a equipes com apenas um desenvolvedor. A abordagem utilizada desconsidera equipes com mais de uma pessoa, ponto abordado pelo presente trabalho, com o objetivo de propor uma abordagem para equipes pequenas.

2.3.2 Um Método Ágil Híbrido

Prass e Puntel (2010) identifica que nos últimos tempos surgiram um grande número de métodos para gerencia de software, porém nem sempre estas são adequadas aos problemas enfrentados pelas micro e pequenas empresas. Para resolver este problema o trabalho propõe uma análise das boas práticas dos mais conhecidos métodos ágeis e posteriormente fazer uma mescla destes buscando se adequar ao cenário de micro e pequenas empresas, formando assim um método híbrido.

Para que este trabalho tenha algum valor seria necessário aplicar o método proposto em algum ambiente de desenvolvimento, então inicialmente foi exposto o paradigma ágil aos diretores de uma empresa para que apoiassem a implantação do mesmo, tendo o aceite passou para a próxima fase, foi definir entre as boas práticas do FDD, XP, *Scrum* e Microsoft Solutions Framework (MSF) que melhor serviriam para solucionar problemas identificados no processo de desenvolvimento.

Feito isto, deu início a fase de implementação das boas práticas selecionadas, de uma forma gradual. Foram distribuídos papéis e modificado a forma de estimar o tempo de desenvolvimento. Foi explicado aos clientes a nova forma de trabalho para

desenvolvimento dos produtos e modificado a forma de interação entre empresa e cliente. No momento de modelagem prezou pela simplicidade e objetividade, se precavendo, caso houvessem mudanças nos requisitos. Durante a codificação permaneceu o foco na simplicidade, adicionando padronização ao código e comentários também padronizados. O desenvolvimento foi feito de forma iterativa, e ao fim de cada iteração uma etapa chamada de “avaliar e reagir” foi incluída justamente para examinar o decorrer no desenvolvimento e propor adaptações, visando a melhoria continua encerrando assim o ciclo.

Os pontos positivos observados foram: ganho de tempo com a padronização tanto na manutenção como no desenvolvimento, a documentação estabelecida refletiu em ganho de produtividade, a melhoria na definição de requisitos e com o decorrer dos projetos houve uma alta aceitação das boas práticas pelas equipes.

Os resultados negativos observados foi o difícil convencimento e entendimento do paradigma ágil aos diretores das empresas, onde o método seria aplicado. A dificuldade em educar a equipe de desenvolvimento as boas práticas, também sempre haverá erros na escolha das boas práticas e várias modificações, a ausência do cliente dificultou a procedência de algumas boas práticas e pôr fim a carência de gerência acabou com que a equipe tivesse de desviar o foco de suas atividades.

Este trabalho constatou que o envolvimento deve ser total desde a diretoria aos desenvolvedores, incluindo o cliente, e com isto a empresa se tornará ágil.

Os pontos positivos, gerados pelas adequações dos métodos ágeis, citados pelo trabalho exposto nesta seção, confirmam que existe a necessidade de adaptação dos métodos ágeis. Justificando a necessidade do presente trabalho.

2.3.3 Relato de Experiência na Implantação de um Método Ágil em uma Equipe de Desenvolvimento de Software

Coelho (2011) identificou a escassez de relatos sobre os resultados obtidos pela implantação dos métodos ágeis, sendo que o modo de validação das mesmas é baseado em nestes relatos. Sendo assim o objetivo do trabalho é responder as seguintes questões: “Quais são os benefícios do uso dos métodos ágeis? O que efeito no desenvolvimento ágil para se alcançar as melhorias? E quais são as melhorias

proporcionadas pelas práticas *Scrum*? ”, estas são respondidas por meio de um estudo de caso único.

A equipe participante do estudo de caso utilizou do *Product Backlog*, *Sprint*, reuniões diárias e reunião de retrospectiva, que são boas práticas do *Scrum.*

O projeto foi dividido em quatro fase, na primeira delas foi definido o escopo do projeto e levantado requisitos pelo *Product Owner*, porém não se utilizou nenhuma boa prática ágil, e esta fase foi chamada de concepção. A partir da segunda fase foram utilizadas as práticas, esta fase foi a fase de elaboração, onde foi levantado as prioridades juntamente com o cliente, gerando um *Product Backlog*, a reunião de planejamento da *Sprint* também entrou nesta fase. A terceira fase, chamada de construção, inicia com o *Sprint* de 2 a 4 semanas produzindo parte do produto, esta utilizou de algumas das práticas do movimento ágil, como a programação em par, reuniões diárias, porém estas não foram executadas corretamente, por incompatibilidade de horário dos desenvolvedores, *Sprint Review*, *Sprint Retrospective*. A quarta e última fase nomeada de transição é feito o *commit* dos códigos em um servidor de desenvolvimento, facilitando na refatoração do código.

Ao termino do projeto foi feito uma análise e constatou problemas na comunicação, planejamento das *Sprints*, processo de desenvolvimento, erros nas especificações e atrasos nas entregas, e recomenda-se o uso de outros métodos em conjunto para resolvê-los. Contudo também se destacam pontos positivos, participação ativa e *feedback* do cliente permite um melhor entendimento do sistema, produzindo um produto correto conforme as especificações e objetivos do cliente.

3. MATERIAIS E MÉTODOS

Método de pesquisa consiste em reunir métodos, técnicas e procedimentos que tenham o objetivo de tornar possível a execução da pesquisa, resultando em um novo conhecimento, produto ou processo (JUNG, 2009).

Pela ótica da natureza do problema da pesquisa, este trabalho se encaixa como uma pesquisa tecnológica ou aplicada. Desenvolve uma abordagem qualitativa com pressuposto exploratório, abordando um procedimento de estudo de caso único, como pode ser observado na Figura 2.

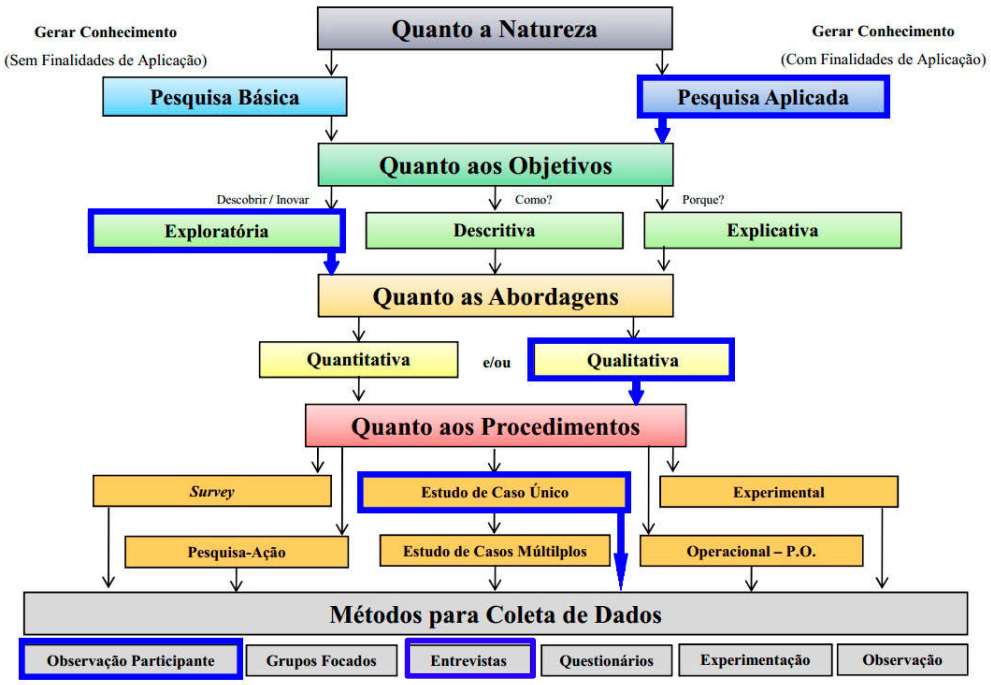

Figura 2 - Classificação da pesquisa adaptado de (JUNG, 2002)

A pesquisa aplicada tem como objetivo construir conhecimento para aplicações práticas com capacidade de resolver problemas pontuais (SILVA; MENEZES, 2005).

Pesquisas exploratórias propiciam maior proximidade com o problema, objetivando torná-lo explicito. Este tipo de pesquisa busca aprimorar ideias, com planejamento flexível, dando margem a consideração de vários aspectos ao estudo abordado. Em sua maioria, elas consistem em uma ou mais destas características:

levantamento bibliográfico, entrevistas com pessoas envolvidas com o problema e/ou análise de exemplos (Gil, 2002).

Sobre a abordagem qualitativa Silva e Mendez (2005) constatam que o ambiente natural é origem para tomada de dados, não requer uso de técnicas estatísticas, analisando-os de forma indutiva com o foco voltado aos significados. E para Gil (2002) esta abordagem depende de muitos fatores, este tem um processo sequencial de atividades, implica na redução dos dados, categorização, interpretação e formulação dos resultados.

Sobre os procedimentos técnicos a pesquisa é identificada como estudo de caso. Estudo exaustivo e em profundidade de um ou mais objetivos permitindo gerar um conhecimento amplo e detalhado (SILVA; MENDEZ, 2005).

Sobre os métodos de coletas de dados Yin (2002) relata que cada fonte de evidências é mais adequada para tipos diferentes de pesquisa, e a escolha de várias fontes de evidências em combinação, reflete em uma pesquisa acurada. Neste trabalho as fontes de evidências utilizadas são: Entrevista e Observação participante.

3.1 PROCEDIMENTOS METODOLÓGICOS

Este trabalho está dividido em quatro fases:

1.Entendimento do problema e revisão bibliográfica sobre métodos ágeis de desenvolvimento

Dentro do contexto deste trabalho identificou-se a carência de estudos na área de engenharia de software, mais especificamente na gestão de projetos no cenário das pequenas e microempresas de desenvolvimento de software. Posteriormente identificado buscou-se compreensão das características do manifesto ágil focando no método *Scrum*.

2.Aplicação do método *Scrum* em uma pequena equipe

Neste período foi aplicado o método *Scrum* em uma equipe pequena em um projeto real, com o intuito de verificar se a forma tradicional do *Scrum* pode ser utilizada em equipes com tamanho reduzido, ou necessita de adequações.

3. Coleta e análise dos dados

Durante a utilização do método *Scrum* tradicional na equipe foco do estudo é feita a primeira parte da coleta de dados, por meio da observação assistida. E após

finalizar o projeto é realizada a segunda etapa da coleta de dados, por meio de entrevistas.

Após realização da coleta é feito a análise dos dados, que foram obtidos por meio da observação participada e entrevistas. Estas informações são cruzadas a fim de indicar os pontos negativos e positivos da aplicação do *Scrum* no desenvolvimento do projeto.

4.Elaboração da proposta de abordagem *Scrum*

Propor uma abordagem que tente mitigar os pontos negativos, apontados durante a fase de coleta e análise de dados, da utilização do método *Scrum* tradicional na equipe foco do estudo.

4. DESENVOLVIMENTO

Neste capitulo é apresentado uma visão geral do desenvolvimento do trabalho, incluindo a descrição do ambiente na qual o estudo foi aplicado, o andamento do projeto, levantamento dos problemas e considerações sobre a abordagem proposta.

4.1 EMPRESA E PROJETO

Para o desenvolvimento deste trabalho foi utilizada uma equipe da empresa júnior de desenvolvimento web residida na Universidade Estadual do Norte do Paraná. Dentre suas características, essa empresa possui alta rotatividade de membros. Estes membros são graduandos do Centro de Ciências Tecnológicas e assinam um termo de voluntariado, cumprindo uma carga horaria de 12 horas semanais presenciais.

Os projetos são, em sua maioria, realizados para a comunidade de Bandeirantes no estado do Paraná e região e para própria universidade. Estes projetos tem um prazo maior e custo menor comparados a empresas convencionais de desenvolvimento.

O projeto tido como base para este estudo tinha como objetivo o desenvolvimento de um site para exposição de imóveis destinados à locação e venda. Logo o site é composto por: (1) *home page* ou página principal que expõe os imóveis em destaque; (2) página destinada à busca de imóveis a partir de filtros; (3) página com descrição dos serviços prestados pela imobiliária; (4) página contendo histórico; e (5) outra com um formulário para contato. Há também uma página de administração com acesso restrito ao usuário gerenciador do site, com opções de cadastro de novos imóveis, edição de imóveis já cadastrados e exclusão de imóveis.

O prazo máximo estabelecido para a entrega foi de 120 dias. Uma equipe de três pessoas foi definida para o desenvolvimento do projeto, sendo uma para o cargo de *Scrum Master* e *Product Owner* e outras duas compondo o Time de Desenvolvimento.

4.2 ESTRUTURA DA EQUIPE DO PROJETO

O Time *Scrum* definido para este projeto contou com três integrantes, distribuídos seguindo os papeis do *Scrum* da seguinte maneira. No Time de

desenvolvimento foram alocados dois membros da empresa, ambos com conhecimento básico no desenvolvimento *back-end* e *front-end*, porém um deles especialista em *front-end* e o outro no desenvolvimento *back-end.* O pesquisador deste trabalho, e também membro da empresa júnior, ocupou o cargo de *Scrum Master* e *Product Owner*.

Em atividades que foram necessárias a criação de imagens foi terceirizada um designer. A empresa possui designers para estes fins, porém estes estavam alocados em outros projetos, não podendo compor a equipe deste projeto. Outro fator para não ser associado um designer a esta equipe foi pelo fato do projeto não ter atividades suficientes para tal inclusão. Contudo, quando necessário o Time de Desenvolvimento recorria ao designer da própria empresa para realizar estas tarefas.

O integrante responsável pelo papel de *Scrum Master* absorveu as responsabilidades do *Product Owner.* Ele ficou incumbido de ser o contato direto com o cliente e único responsável pela elaboração do *Product Backlog*. Pela falta de integrantes disponíveis na empresa para ocupar tal cargo, foi feita então a junção destes dois papéis em um único.

4.2.1 Decorrer do projeto

Após a negociação de valor e prazo de entrega, baseado no levantamento de requisitos feito entre *Product Owner* e o contratante, foi elaborado o *Product Backlog*.

Na sequência a diretoria da empresa contratada delegou os membros para compor o Time *Scrum*, que ficou estruturado da seguinte forma: um membro nomeado como *Scrum Master* e *Product Owner* e dois membros compondo o Time de Desenvolvimento, com habilidades em *Front-end* e *Back-end*.

Apesar do projeto ter o prazo de 120 dias, as *Sprints* foram iniciadas após os 40 dias, pelo fato dos desenvolvedores estarem ocupados em outros projetos. O projeto foi composto por cinco S*prints* com duração de 2 semanas cada.

Para iniciar a *Sprint* foi feito então uma reunião de planejamento com o objetivo de definir o *Sprint* Backlog a partir das tarefas do *Product Backlog*, traçando o que deveria ser desenvolvido durante a *Sprint*.

Diariamente houve tentativas de se reunir para a Daily *Scrum*, porém a incompatibilidade de horários entre membros da equipe impossibilitou estes encontros. Como medida paliativa, os membros utilizavam-se de encontros causais

fora do expediente e meios alternativos para a troca de informações, tais como redes sociais.

No término da *Sprint*, foi realizada a reunião com o cliente e Time *Scrum* a fim de expor o que foi produzido no decorrer das duas últimas semanas. O cliente era conduzido a falar sobre o que estava sendo entregue, se estava de acordo com o esperado ou distanciando do que havia pedido, e também as expectativas para os próximos itens a serem desenvolvidos. Esta rotina auxiliou a refinar constantemente os itens do *Product Backlog* e redefinir as prioridades do cliente.

Ao termino das reuniões com o cliente, um novo evento se inicia, a Retrospectiva da *Sprint.* Nestas reuniões além do *Scrum Master* e do Time de Desenvolvimento discutirem as formas de trabalho empregadas durante a *Sprint,* também era utilizada para realizar mudanças no *Product Backlog* e um pré planejamento da *Sprint* seguinte. Isto foi necessário para que as informações cedidas pelo cliente na reunião anterior não fossem esquecidas, fazendo com que a reunião de planejamento da *Sprint,* que seria realizada no início da próxima semana, cumprisse com o objetivo de desenvolver algo de maior valor para o cliente.

4.3 LEVANTAMENTO DOS PROBLEMAS NO *SCRUM* EM PEQUENAS EQUIPES

Para a coleta de evidências, neste estudo de caso foram utilizadas duas fontes de evidências: entrevista e observação participante. Segundo Yin (2002), a entrevista é uma das mais importantes fontes de evidências, podendo investigar sobre pontos específicos e eventos ou pedir para que o entrevistado relate suas próprias interpretações sobre os acontecimentos. A observação participante é uma modalidade em que o pesquisador pode assumir funções dentro do estudo de caso, sendo um participante ativo dentro do estudo de caso. Foi utilizado ambas as fontes de evidências, para evitar o enviesamento de dados.

O pesquisador trabalhando como *Scrum Master* e *Product Owner* da equipe foco do estudo, pode analisar a aplicação do método *Scrum* em uma pequena equipe. E por meio desta observação participante, três pontos destoaram ao que é proposto pelo *Scrum* tradicional. O primeiro deles está presente na distribuição dos papéis, pelo fato dos papéis *Product Owner* e *Scrum Master* ocuparem dois integrantes do Time *Scrum*, restando apenas um membro para compor a Equipe de Desenvolvimento. Os demais pontos encontrados foram durante os eventos, pela falta de necessidade das

Reuniões Diárias ocasionada pela alta proximidade da Equipe de Desenvolvimento e nas Reuniões de Retrospectiva, com o foco diferente do que é pregado pelo método tradicional do *Scrum*, dando margem para discussões sobre as prioridades do *Product Backlog*.

A outra fonte de evidência deste trabalho, entrevista, com perguntas semiestruturadas, conduzida de forma espontânea e feitas individualmente com os demais participantes da equipe foco de estudo. Em suma relataram que a utilização de um método para organizar o desenvolvimento teve um respaldo positivo, pelo fato de trazer organização e visão do processo. Ao serem questionados sobre o *Scrum* aplicado no projeto e se tiveram algumas diferenças (SCHWABER; SUTHERLAND, 2013).

As complicações relatadas durante as entrevistas estão relacionadas à composição do Time *Scrum* e à reunião diária. Sobre a divisão dos papeis no Time *Scrum,* houve uma sobrecarga de atividades pelo fato de um membro ficar encarregado dos papéis de *Product Owner* e *Scrum Master*. Estas diferenças prejudicaram o andamento do projeto, em momentos que o Time de desenvolvimento precisou entrar em contato direto com o cliente. Pois os desenvolvedores não devem ter contato direto com o cliente durante o período de desenvolvimento, as dúvidas do Time de Desenvolvimento devem ser sanadas pelo *Product Owner*, uma vez que este deve saber as especificações do projeto.

Outro apontamento durante as entrevistas foi sobre a falta de necessidade das reuniões diárias por conta da proximidade do time de desenvolvimento e uma comunicação informal bem estabelecida entre os membros. E outro motivo pela qual não verifica necessidade destas reuniões é pelo fato de que as tarefas geralmente demoram mais de um dia para serem finalizadas, causando uma falta de conteúdo a ser discutido pela proximidade temporal destas.

Ambos fatores apontados pelas entrevistas coincidem com as duas diferenças encontrados pela observação assistida que são: na estrutura do time *Scrum*, na Reunião Diária. Estes problemas identificados serão tratados a seguir neste trabalho.

4.4 CONSIDERAÇÕES PARA NOVA ABORDAGEM

Este capítulo comporta as comparações entre as diretrizes do *Scrum* tradicional ao *Scrum* aplicado no projeto foco do estudo e as considerações apontadas relevantes, pelas fontes de evidências.

4.4.1 Papéis

Product Owner

Scrum tradicional: Responsável pela gerência do *Product Backlog*, incluindo a alteração, exclusão ou inserção de itens no *Product Backlog*. É uma atividade exclusiva do *Product Owner.* Ele deve expressar claramente cada requisito do produto e determinar a ordem de desenvolvimento dos itens segundo a prioridade de cada item definida por ele. Este membro se relaciona diretamente com o cliente e deve ter o entendimento total do sistema.

Scrum do projeto: Devido ao número de membros reduzido na equipe do projeto houve falta de recursos humanos disponíveis para o compor este papel, e suas responsabilidades foram transferidas para o *Scrum Master*.

Considerações: Schwaber e Sutherland (2013) concedem a liberdade do *Product Owner* delegar suas tarefas para os desenvolvedores, mas ele deve aprovar ao final, e continua sendo o responsável por fazer alterações. No entanto, pela falta de recursos humanos em uma pequena equipe não é viável a utilização deste papel, e suas atividades devem ser distribuídas pelo Time *Scrum*, não apenas ao Time de desenvolvimento.

Scrum Master

Scrum tradicional: Responsável por educar o time conforme as regras do *Scrum*, controlar as reuniões e tempo limite dos eventos. Também deve deixar o Time de Desenvolvimento livre de empecilhos, e proporcionar a eles maior tempo hábil voltado ao desenvolvimento.

Scrum do projeto: Permaneceu com as responsabilidades do *Scrum* tradicional, porém houve o acréscimo das responsabilidades do *Product Owner*. Tornou um único membro responsável por dois papéis.

Considerações: O resultado desta união de papéis foi a sobrecarga de atividades, e dificuldade de exercer com excelência ambas atividades, tornando um fator problemático no *Scrum* em equipes pequenas.

Time de Desenvolvimento

Scrum tradicional: Responsável por entregar software de valor ao cliente de forma incremental. Composto por mais de três e menos de nove integrantes. O Time de Desenvolvimento deve ser multifuncional e capaz de entregar incremento do produto ao final de uma *Sprint*. O time deve ser auto-organizado, sendo responsável pela divisão de tarefas entre si.

Scrum do projeto: Foi responsável pelas entregas de software de valor ao cliente de forma incremental. Porém, composto por apenas dois integrantes e recebeu auxilio do *Scrum* Master na divisão das tarefas no início do projeto. O Time de Desenvolvimento necessitou de um canal de comunicação direta com o cliente, pois o *Product Owner* em alguns momentos estava ausente devido à sobrecarga de trabalho.

Considerações: Para o alivio da sobrecarga de atividades do *Scrum Master* o Time de desenvolvimento criou um contato direto com o cliente. Além disso outras atividades do *Product Owner* foram delegadas ao Time de desenvolvimento como: repriorização do *Product Backlog* para garantir que a visão de todos sobre o produto seja a mesma.

4.4.2 Eventos

Planejamento da *Sprint*

Scrum tradicional: Esta reunião tem como finalidade elaborar o *Sprint Backlog*, ou seja, definir as tarefas a serem realizadas durante a *Sprint* e o incremento do produto a ser entregue ao final deste período de desenvolvimento.

Scrum do projeto: Durante o projeto este evento permaneceu conforme as diretrizes do *Scrum* tradicional.

Considerações: Este evento ocorreu sem apresentar pontos negativos durante o projeto foco do estudo.

Sprint

Scrum tradicional: Período de desenvolvimento com duração máxima de quatro semanas e mínima de duas. Ao fim deste evento é entregue algo de valor ao cliente.

Scrum do projeto: Durante o projeto este evento permaneceu conforme as diretrizes do *Scrum* tradicional.

Considerações: Este evento ocorreu sem apresentar pontos negativos durante o projeto foco do estudo.

Reunião Diária

Scrum tradicional: É um evento com duração máxima de quinze minutos com o intuito de propiciar aos desenvolvedores um tempo para que sincronizem suas atividades e planejem suas próximas vinte e quatro horas. Também é utilizado para inspecionar o progresso das atividades e medir se será possível entregar o incremento proposto na Reunião de planejamento.

Scrum do projeto: Não houve uso deste evento.

Considerações: Neste projeto a troca de informações dos desenvolvedores ocorreu de um modo informal, por conversas durante os expedientes em comum ou fora do horário de expediente, por meio de redes sociais. Esta troca continua de informações tornou desnecessária a realização de reuniões diárias.

Entretanto, deve ser estabelecido um tempo para que haja troca de informações, isto porque com os desenvolvedores tendo contato com o cliente visões diferenciadas sobre o projeto podem aparecer dentro do Time *Scrum*. Para que isto não ocorra é necessário estabelecer um evento formal para nivelamento da visão sobre o escopo geral do produto.

Revisão da *Sprint*

Scrum tradicional: Executado no final da *Sprint* com a participação do cliente e com a finalidade de inspecionar o incremento produzido durante a *Sprint,* adaptar o *Product Backlog* e discutir juntamente com o cliente o que deve ser feito a seguir. O time de Desenvolvimento deve apresentar ao cliente o que foi desenvolvido na *Sprint*. E este evento também é utilizado para tirar dúvidas com o cliente, caso houver.

Scrum do projeto: Este evento precisou de ser adiantado em alguns momentos para que o cliente pudesse estar presente. Porém, a finalidade do evento e as demais características permanecem as mesmas do *Scrum* tradicional.

Considerações: Apesar de adiantar esta reunião em algumas *Sprints* estas mudanças não foram problemáticas, pois eram agendadas com antecedência.

Retrospectiva da *Sprint*

Scrum tradicional: Este evento é o momento de trazer melhorias para o Time *Scrum*. Neste momento deve ser apontado as falhas da última *Sprint* e propor ações para otimizar o trabalho nas próximas.

Scrum do projeto: Estas reuniões foram pouco utilizadas para trazer melhorias internas, e poucas propostas partiram do Time *Scrum*. Geralmente tinham duração reduzida.

Considerações: Pelo fato do método *Scrum* ser uma novidade para a equipe poucas sugestões de mudanças na forma de preceder durante a *Sprint* foram levantadas, tornando uma reunião curta.

4.4.3 Artefatos

Product Backlog

Scrum tradicional: Um documento ordenado de itens a serem cumpridos durante todo o projeto, e deve ser capaz de traduzir por si só o produto a ser desenvolvido. Este documento é alterado conforme são identificadas possíveis mudanças para o produto ser mais útil e apropriado. O único com permissão para alterar este documento é o *Product Owner*. Os itens do *Product Backlog* são compostos por descrições, estimativas e valor.

Scrum do projeto: O *Scrum Master* ficou responsável pela elaboração e alteração deste documento.

Considerações: Esta alteração, individualmente, não trouxe problema ao andamento do projeto, porem poderia ser distribuída para o resto do Time reduzindo a sobrecarga deste membro.

Sprint Backlog

Scrum tradicional: Este artefato é uma seleção dos itens do *Product Backlog* para serem realizados durante a *Sprint*. Ele contém a descrição técnica e detalhada das funcionalidades para serem entregues ao final da *Sprint*, a junção destes itens forma o incremento.

Scrum do projeto: Não foi necessária alteração neste artefato.

Considerações: Este artefato atendeu as necessidades da equipe sem a necessidade de alteração.

Incremento

Scrum tradicional: Incremento é a junção dos itens do *Sprint* Backlog que foram completos e está utilizável, ao final de cada S*print* um novo incremento é somado aos incrementos anteriores.

Scrum do projeto: Não foi necessária alteração neste artefato.

Considerações: Este artefato atendeu as necessidades da equipe sem a necessidade de alteração.

4.4.4 Time-Box

Scrum tradicional: Tempo limite para os eventos. Os time-box dos eventos: planejamento de *Sprint*, Revisão de *Sprint* e Retrospectiva da *Sprint*, eram baseados no Time-Box da *Sprint*. Isto é, quanto maior for definido o *Time-Box* da *Sprint*, consequentemente maior será o Time-box dos outros eventos. Exceto a Reunião diária, pois esta permanece fixa, quinze minutos.

No *Scrum* tradicional esta é uma grande preocupação do *Scrum Master*, ele deve controlar os eventos para que compram este prazo, pois exceder o tempo limite prejudica o progresso de desenvolvimento do Time.

Scrum do projeto: Não foi um ponto preocupante para o desenvolvimento do projeto, uma vez que as reuniões entre dois ou três integrantes não tornavam discussões prolongadas a ponto de prejudicar o andamento do projeto.

Considerações: Para a método proposto este período de tempo será aberto para os eventos, com exceção da reunião semanal. E proposto que em trabalhos futuros seja feito uma análise sobre necessidade do limite de tempo fixo dos eventos.

5. RESULTADOS

Este capítulo tem a finalidade de expor os resultados encontrados neste trabalho por meio do estudo de caso.

5.1 ABORDAGEM PROPOSTA

Partindo das considerações citadas no capítulo anterior, é proposto uma abordagem do método *Scrum* com a finalidade de mitigar os problemas encontrados no método tradicional quando utilizado por equipes pequenas.

A Figura 3 demonstra o ponto inicial da S*print* com o evento “planejamento da *Sprint*” e terminando com “Retrospectiva da *Sprint*”. Cada evento produz um artefato de saída, este artefato é então utilizado pelo próximo evento como artefato de entrada.

Sprint devem ser feitas enquanto existirem itens no *Product Backlog* para serem realizados, quando todos os itens forem cumpridos então o Software está pronto.

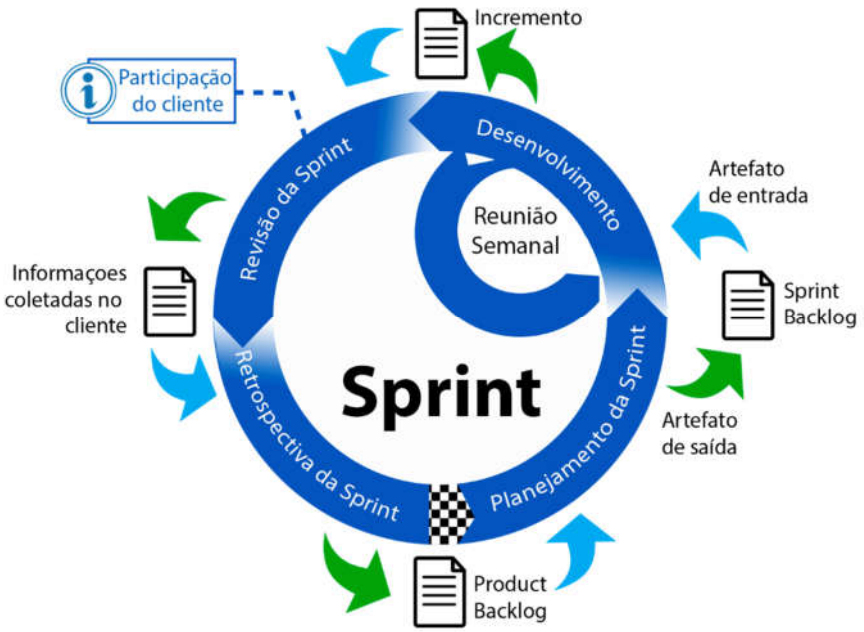

Figura 3 - Abordagem proposta

Nas seções seguintes são apresentadas as propostas de adequações feitas no *Scrum* tradicional.

5.1.1 Eventos

As modificações presentes nos eventos são encontradas principalmente na reunião de planejamento, Reunião Semanal e Reunião de Retrospectiva. E são detalhadas no decorrer dessa seção.

Estas mudanças foram ocasionadas principalmente pela distribuição das responsabilidades do *Product Owner*, as decisões tomadas anteriormente por este único membro, agora são discutidas pelo Time. Ciando a necessidade de um tempo para tais discussões, alterando diretamente os eventos.

1. Reunião de planejamento. Nesta reunião o *Product Backlog* é o artefato de entrada, o Time de Desenvolvimento deve avaliar os itens com a prioridade mais alta e decidirem quais destes itens serão entregues ao final da *Sprint*.

Por conta da diluição das responsabilidades do *Product Owner* pelo Time, os desenvolvedores devem se preparar para a reunião, trazendo detalhadamente os itens de maior prioridade do *Product Backlog*.

O artefato de saída desta reunião é o *Sprint Backlog*, documento descrevendo o incremento a ser gerado por esta *Sprint*.

2. *Sprint*. Este evento deve durar no máximo quatro semanas e no mínimo duas. O artefato de entrada deste evento é o *Sprint Backlog* e o artefato de saída é o incremento do produto.

A obrigação do *Scrum* Master durante este evento é estar apto a proporcionar um ambiente de desenvolvimento favorável ao Time de desenvolvimento retirando os impedimentos que possam dificultar a entrega do incremento ao final da *Sprint*. Por conta da remoção das reuniões diárias, onde seriam apontados estes impedimentos, o *Scrum Master* deve dobrar a atenção a estes fatores.

O papel do desenvolvedor é converter os itens do *Sprint Backlog* em incremento do produto. E apesar de o desenvolvedor ter contato direto com o cliente durante este evento, não deve ser frequente, para que não atrapalhe no desenvolvimento do incremento. As dúvidas devem ser sanadas na reunião de revisão, a não ser que esta dúvida seja um impedimento para a entrega do incremento.

A *Sprint* é composta por Reunião de planejamento, reunião semanal, reunião de revisão e Reunião de retrospectiva.

3. Reunião Scrum. A duração deste evento não pode exceder quinze minutos. Esta reunião deve ser feita no segundo dia da semana da *Sprint* e as

próximas com intervalo de um dia, sendo assim, equipes que trabalham cinco dias por semana terão duas Reuniões Scrum e equipes que trabalham seis dias terão três.

Equipes que trabalham seis dias por semana, não fazem Reunião Scrum na última semana da *Sprint*, uma vez que no final desta semana será feito a Reunião de Retrospectiva e tratará destes assuntos.

A Figura 4 exemplifica os dias de Reunião Scrum durante a semana. No lado esquerdo uma equipe que trabalha cinco dias da semana com *Sprint* de duas semanas de duração e no lado direito equipes que trabalham seis dias da semana e *Sprint* com duração de três semanas.

Figura 4 - Reunião Scrum

A Reunião Semanal tem o objeto de manter o Time *Scrum* engajado e que obrigatoriamente o Time de Desenvolvimento mantenha uma comunicação para sincronizar as atividades.

É importante que o todos os membros do Time *Scrum* estejam presentes nesta reunião. Para que seja discutido a visão de cada um sobre o produto e unificar esta visão, uma vez que no decorrer da semana o contato com o cliente pode alterar a visão individual dos desenvolvedores.

4. Revisão da *Sprint*. Este evento tem como artefato de entrada o incremento do produto. Ele ocorre no último dia da *Sprint* com a presença do cliente e todo o time *Scrum*.

Como consequência da divisão das responsabilidades do *Product Owner* entre *Scrum* Master e time de Desenvolvimento, cabe ao *Scrum Master* abordar o assunto sobre: esclarecer os itens do *Product Backlog* que foram completos, o estágio do projeto em relação ao *Product Backlog* e possíveis datas de conclusão do projeto.

Os desenvolvedores apresentam o incremento ao cliente e também utilizam a presença do cliente para questioná-lo sobre as dúvidas que surgiram durante a *Sprint*.

O cliente deve avaliar o incremento e ser um participante ativo nesta reunião, fornecendo informações valiosas para a adaptação do *Product Backlog*. As informações cedidas por ele serão utilizadas na próxima reunião, reunião de retrospectiva. O cliente também deve ser capaz de responder as dúvidas do Time sobre o projeto.

5. Retrospectiva da *Sprint*. Evento que encerra a *Sprint* com a participação de todo o Time *Scrum*. Os objetivos deste evento é melhorar a forma de trabalho visando o maior rendimento do Time, unificar a visão do Time sobre o produto e repriorizar o *Product Backlog*.

O *Scrum* Master deve fomentar o Time de Desenvolvimento a apontar falhas no Time e trazer propostas que proporcionem melhorias a ele. Em decorrência de todos possuírem contato com o cliente, diferentes visões sobre o produto podem surgir dentro do time, portanto durante esta reunião deve ser reservado um momento para que se discuta o entendimento de cada um sobre o projeto.

Um momento importante desta reunião é a discussão sobre alterações no *Product Backlog*. O *Scrum* Master deve conduzir este trecho da reunião a fim de trazer maior organização ao artefato.

O *Product Backlog* deve ser atualizado constantemente, e por esta abordagem não conter um único responsável pela alteração deste artefato, é necessário um momento para que, em conjunto, sejam feitas as alterações. Estas alterações devem ser feitas durante este evento, pois ele ocorre logo após uma reunião na qual o cliente estava presente, e antecede a reunião que será planejado o *Sprint* Backlog com base no *Product Backlog*.

5.1.2 Papeis

A principal modificação nos papéis é a distribuição das responsabilidades do *Product Owner* entre *Scrum* Master e Time de desenvolvimento, ocasionado pelo número de papéis ser incompatível com o número de membros na equipe.

1. *Scrum Master*. Este membro deve ter a característica de liderança. Geralmente é a pessoa com mais conhecimento sobre as regras da abordagem do Scrum e suas decisões devem ser respeitadas.

As funções deste membro são:

- Deixar o Time de Desenvolvimento desimpedidos para codificar. Quaisquer impedimentos que apareçam durante o período de

desenvolvimento devem ser levados ao *Scrum Master*, para que este tome as medidas cabíveis;

- Por ser o membro com mais experiência do Time deve educar o Time segundo as diretrizes desta abordagem do *Scrum*;
- Fomentar o auto-gerenciamento do Time de Desenvolvimento;
- Cobrar que os eventos sejam cumpridos conforme a abordagem propõe;
- Ser o principal contato do entre cliente e Time *Scrum*. Apesar de todo o Time ter contato com o cliente, é aconselhável que o *Scrum* Master regule a frequência de comunicação entre o Time de desenvolvimento e cliente, a fim de não atrapalhar o progresso do Time; e
- Liderar o Time nas decisões sobre o *Product Backlog*, embora esta decisão seja tomada pelo todo, isto é feito para manter a organização e assegurar a clareza do artefato.

2. Time de Desenvolvimento. Composto por desenvolvedores, estes realizam o trabalho de converter itens do *Product Backlog* em incrementos utilizáveis.

O Time de Desenvolvimento deve:

- Ser capaz de realizar a grande maioria dos itens do *Backlog* por ser multifuncional, ou seja, ter a capacidade de criar um incremento do produto. Caso alguma tarefa exija um especialista, este deve ser terceirizado apenas para realizar esta tarefa, e não compor o time durante todo o projeto;
- Deve ser auto-organizado, isto é, ser capaz de dividir as tarefas entre si de modo a entregar o incremento ao final de uma *Sprint*;
- Pode conter especialistas, porém o time é visto como um todo, o importante é ser capaz de entregar o incremento;
- Decidir o que pode entrar no *Sprint Backlog* durante a reunião de planejamento, sendo responsável por cumprir o que prometeu entregar;
- Ser responsável, junto com o *Scrum Master*, por alterações no *Product Backlog*, como: repriorização, inclusão e alteração de itens;
- Ser responsável por a analisar o *Product Backlog*, principalmente os itens mais ao topo, antes das reuniões de Planejamento; e

- Ter o contato com o cliente, porém com salvaguarda, não deve ser frequente a ponto de atrapalhar o progresso do desenvolvimento.

5.1.3 Artefatos

As diferenças propostas nos artefatos encontram-se nas permissões de alteração.

1. *Product Backlog*. Este artefato é uma lista de itens ordenada por prioridade, quanto maior o nível de prioridade, mais alto ele estará do topo, e mais rapidamente deve ser posto em desenvolvimento para ser um incremento do produto.

Os itens do *Product Backlog* são compostos por:

- ID – Identificador único;
- Nível de prioridade – Quanto maior este indicador mais logo deve ser posto em desenvolvimento;
- Tema – Descrição curta de no máximo cinco palavras que identifique a atividade;
- Estimativa de duração – Uma suposição de tempo que esta tarefa irá demorar para ser concluída, deve ser padronizada, ou em horas ou dias, a equipe pode escolher conforme o que melhor se adaptarem; e
- Observações – Este campo é livre, para ser feito apontamentos sobre a tarefa.

Estes atributos podem ser modificados, adicionado outros campos ou retirados, conformo o Time *Scrum* julgue necessário.

A prioridade do item deve ser discutida pelo Time nas reuniões de retrospectiva, sempre colocando o que é de maior valor para o cliente com a prioridade mais alta.

Os itens mais ao topo devem ser os mais claros e bem detalhados, pois estes estão prestes a entrar em fase de produção. Fica a cargo de todo o Time buscar informações de requisitos sobre os itens. A auto-organização do Time deixa aberto como será feita esta divisão de tarefas.

Apesar do Time *Scrum* ter o poder de alterar este artefato, ele só pode ser feito na presença de todo o Time, preferivelmente durante as reuniões de Retrospectiva da *Sprint*. O *Scrum* Master tem peso maior em opinar nas alterações, em caso em empasses tem a palavra final.

2. *Sprint Backlog*. Este artefato é uma seleção dos itens de maior prioridade do *Product Backlog*. Esta seleção é feita pelo Time de desenvolvimento na reunião de planejamento da *Sprint.*

Este documento deve representar o incremento a ser entregue no final da *Sprint*. Os itens são mais detalhados e técnicos em comparação aos do *Product Backlog*, estes atributos são:

- ID – Identificador único, deve ser o mesmo do *Product Backlog*;
- Tema – Descrição curta de no máximo cinco palavras que identifique a atividade, deve ser o mesmo do *Product Backlog*;
- Descrição – Esta descrição deve ser suficiente para que qualquer membro do time a qualquer momento da *Sprint* consiga desenvolver este item sem nenhuma informação adicional; e
- Responsável – Nome do desenvolvedor responsável por completar está atividade.

Estes atributos podem ser modificados, adicionado outros campos ou retirados, conformo o Time *Scrum* julgue necessário.

Durante a *Sprint*, caso necessite ser feito algum trabalho não identificado anteriormente, este deve ser adicionado ao *Sprint Backlog*.

3. Incremento. É uma parte do produto. Uma *Sprint* deve gerar um incremento, isto é, uma parte utilizável do produto. Esta parte deve ser capaz de integrar aos outros incrementos produzidos anteriormente.

5.2 DIFERENÇAS ENTRE SCRUM TRADICIONAL E ABORDAGEM PROPOSTA

O Quadro 1 expõe as diferenciações dos eventos, papéis e artefatos entre Scrum tradicional e da abordagem proposta por este trabalho.

	Scrum tradicional	**Abordagem proposta**
Papéis		
Product Owner	Responsável pelo Product Backlog, contato com o cliente, satisfaz as dúvidas do time sobre o projeto, traz os itens de maior prioridade do Product Backlog detalhados para a reunião de planejamento da *Sprint.*	Extinção do papel, tarefas foram divididas entre Scrum Master e Time de Desenvolvimento.
Scrum Master	Livra o time de impedimentos, educa o time sobre o Scrum, fomenta o auto-gerenciamento.	Continua com as competências do Scrum tradicional, porém deve ser o principal contato com o cliente e liderar as alterações do Product Backlog.

Time de Desenvolvimento	Auto-organizados, pode conter especialistas, decidem o que deve entrar no *Sprint Backlog.*	Continua com as competências do Scrum tradicional, porém pode ter contato com o cliente, modifica o Product Backlog e traz os itens de maior prioridade do Product Backlog detalhados para a reunião de planejamento da Sprint.
Eventos		
Reunião de planejamento	Product Owner traz os itens de maior prioridade do Backlog detalhados e Time de Desenvolvimento decide o que entra no Sprint Backlog.	Time de Desenvolvimento traz os itens de maior prioridade do Backlog detalhados e Time de Desenvolvimento decide o que entra no Sprint Backlog.
Sprint	Scrum Master deve proporcionar um ambiente favorável para os desenvolvedores e Time de Desenvolvimento deve entregar algo de valor para o cliente.	Scrum Master deve aumentar a atenção sobre as possíveis adversidades dos desenvolvedores e Time de desenvolvimento pode contatar o cliente para sanar duvidas
Reunião diária	Reunião utilizada para tratar sobre os itens que já foram feitos, que devem ser feitos no próximo dia e impedimentos do Time de Desenvolvimento.	Substituída pela "Reunião Scrum" acontece com intervalo de um dia, mesmo propósito da reunião diária, porém adicionado a pauta a discussão sobre a visão do projeto.
Revisão da Sprint	Product Owner deve conduzir a reunião e apresentar: itens cumpridos do Product Backlog, estagio do projeto e previsão de entrega.	As tarefas do Product Owner neste evento foram transpostas para o Scrum Master.
Retrospectiva da Sprint	Após a Revisão da *Sprint* e tem o objetivo de melhorar a forma de trabalho.	É adicionado a pauta a discussão sobre a visão do projeto e atualização do Product Backlog.
Artefatos		
Product Backlog	Product Owner tem o direito de fazer as alterações, porem todos podem opinar.	Alteração deste artefato é feita pelo Time Scrum quando todos estão presentes.

Quadro 1 - Diferenças entre Scrum tradicional e abordagem proposta.

Os artefatos Sprint Backlog e Incremento não estão presentes no Quadro 1 pois não obtém diferenças no Scrum tradicional para a abordagem proposta.

6. CONSIDERAÇÕES FINAIS

Este trabalho justifica-se pela necessidade de suprir a dificuldade em encontrar métodos para gestão de projetos de desenvolvimento de software no âmbito de equipes pequenas.

Este estudo de caso foi feito em uma equipe de desenvolvimento de software da empresa júnior residida na Universidade Estadual do Norte do Paraná, em um projeto de desenvolvimento real. Todo o planejamento e desenvolvimento do projeto foi regido pelas diretrizes do método *Scrum*.

Durante o decorrer do projeto foram levantados os problemas encontrados na aplicação do método *Scrum* em uma pequena equipe. Estes problemas foram identificados por meio da observação participante, ou seja, com o pesquisador envolvido diretamente com o projeto. E também por entrevistas, realizadas após o termino do projeto com os outros participantes da equipe. Esta coleta de dados apontou para complicações em alguns dos eventos e papeis do *Scrum*.

Em relação aos papéis do Scrum, houve sobrecarga de responsabilidades causada pela falta de membros disponíveis para divisão dos papéis. Como alternativa foi proposto o cargo de *Product Owner* fosse dissolvido por todo o Time *Scrum*. Isso desencadeou outras alterações como: desenvolvedores devem ter uma comunicação direta com o cliente, tendo permissão para alterar o *Product Backlog* (quando o Time estiver de acordo); o *Scrum* Master deve ser o principal contato com o cliente a fim de que o contato excessivo do cliente com o Time de Desenvolvimento prejudique o andamento do projeto; houve e alterações na pauta da reunião de retrospectiva da *Sprint*.

As alterações na reunião de retrospectiva referem-se à adição de um tópico na pauta para fazer com que todos do time tenham uma mesma visão sobre o projeto. Uma vez que indivíduos do Time de desenvolvimento se comunica o cliente, pode ser formado diferentes visões sobre o projeto.

Outra modificação é a utilização de uma parte do tempo desta reunião para todos do time entrar em um consenso sobre as alterações no *Product Backlog*. Isto é necessário pelo fato de a diluição das responsabilidades do *Product Owner* ter deixado mais de um responsável por este artefato. Este é um momento ideal para tais modificações, pois ela ocorre logo após uma reunião na qual

o cliente estava presente, e antecede a reunião que será planejado o *Sprint Backlog* com base no *Product Backlog*.

Em relação aos eventos do Scrum, a incompatibilidade detectada foi a falta de necessidade de Reunião Diária, por conta da proximidade dos membros possibilitar uma comunicação informal diária satisfatória, tanto durante o expediente, proporcionada pela distribuição física da equipe no ambiente da empresa, quanto em momentos fora do extra expediente, utilizando de trocas de mensagens instantâneas pela internet.

A proposta desta abordagem do método *Scrum* é a substituição da Reunião Diária para uma Reunião Semanal, com o objetivo de manter a visão do produto unificada pelos desenvolvedores. Isto justifica-se por conta do contato isolado de um membro com o cliente pode alterar a visão deste em relação ao produto.

O trabalho evidenciou que o método *Scrum* pode ser empregado em pequenas equipes, desde que passe por ajustes.

Este trabalho foi conduzido dentro de uma empresa júnior, com características distintas de uma empresa de desenvolvimento tradicional. Isto trouxe especificidades para o trabalho, como carga horária diferenciada e nível de formação e experiência dos membros, limitando a abrangência da pesquisa.

6.1 TRABALHOS FUTUROS

Aplicação da abordagem proposta, pois, há necessidade de observar o comportamento das pequenas equipes diante desta abordagem, e se esta traz benefícios no processo de desenvolvimento de software, desta maneira realizar a validação da mesma.

Aplicação da abordagem *Scrum* em pequenas equipes observando a influência do time-box dos eventos e a influência dele sobre o projeto.

Utilização de *Sprints* de uma semana, verificando se aumentar a dinamicidade e inspeção do projeto.

REFERÊNCIAS

AVERY, Josélia Maria Rabêlo. **A INFLUÊNCIA DA ESTRATÉGIA TECNOLÓGICA NA GESTÃO DO CONHECIMENTO E NA INOVAÇÃO EM MICRO E PEQUENAS EMPRESAS DE SOFTWARE**. 2003. 169 f. Dissertação (Mestrado) - Curso de Administração, Universidade Federal do Paraná, Curitiba, 2003.

BARBOSA, António et al. Metodologia Ágil: Feature Driven Development. **Faculdade de Engenharia da Universidade do Porto**, 2008.

BEDÊ, Marco Aurélio; FONSECA, - Paulo Jorge de Paiva; MO, Almiro Breno de. Anuário do Trabalho na Micro e Pequena Empresa 2013. 6. ed. Brasília: Dieese, 2013. 288 p.

BERNI, Jean Carlo Albiero. **GESTÃO PARA O PROCESSO DE DESENVOLVIMENTO DE SOFTWARE CIENTÍFICO, UTILIZANDO UMA ABORDAGEM ÁGIL E ADAPTATIVA NA MICROEMPRESA.** 2010. 77 f. Dissertação (Mestrado) - Curso de Engenharia de Produção, Centro de Tecnologia, Universidade Federal de Santa Maria, Santa Maria, 2010.

BERNI, Jean Carlo Albiero; DORNELLAS, Marcos Cordeiro; FERREIRA, Tiago Keller. **PRODUÇÃO DE SOFTWARE CIENTIFICO ATRAVÉS DE METODOLOGIAS ÁGEIS NA MICROEMPRESA**. 2009.

BONATO, A. Extreme Programming e Qualidade de Software. In: **Congresso Brasileiro Extreme Programming Brasil (http://www.xispe.com. br/evento2002/index2.html, último acesso em 13/10/2003)**. 2002.

BRITTO, Jorge; STALLIVIERI, Fabio. Inovação, cooperação e aprendizado no setor de software no Brasil: análise exploratória baseada no conceito de Arranjos Produtivos Locais (APLs). **Economia e Sociedade,** Campinas, v. 19, n. 2, p.315-358, ago. 2010. Quadrimestral.

BRUNHERA, Diego; ZANATTA, A. *Scrum*: Uma aplicação em uma software house. 2010.

CARVALHO, Bernardo Vasconcelos de; MELLO, Carlos Henrique Pereira. Aplicação do método ágil *scrum* no desenvolvimento de produtos de software em uma pequena empresa de base tecnológica. **Gestão & Produção,** São Carlos, v. 19, n. 3, p.557-573, abr. 2012.

COELHO, Cristiane dos Santos. **RELATO DE EXPERIÊNCIA NA IMPLANTAÇÃO DE UM MÉTODO ÁGIL EM UMA EQUIPE DE DESENVOLVIMENTO DE SOFTWARE.** 2011. 47 f. TCC (Graduação) - Curso de Sistemas de Informação, Departamento de Ciência da Computação, Universidade Federal de Lavras, Lavras, 2011.

DA SILVA, Edna Lúcia; MENEZES, Estera Muszkat. Metodologia da pesquisa e elaboração de dissertação. **Florianópolis, UFSC**, v. 5, n. 6, 2001.

DE CARVALHO, Paulo Roberto Ferreira et al. DESENVOLVIMENTO DE UMA FERRAMENTA WEB PARA O GERENCIAMENTO DE PROJETO DE SOFTWARE UTILIZANDO METODOLOGIAS 100% ÁGEIS. **RE3C-Revista Eletrônica Científica de Ciência da Computação**, v. 9, n. 1, 2014.

DE OLIVEIRA, Fernando Gonçalves; SEABRA, João Manuel Pimentel. METODOLOGIAS DE DESENVOLVIMENTO DE SOFTWARE: UMA ANÁLISE NO DESENVOLVIMENTO DE SISTEMAS NA WEB. **TECNOLOGIAS EM PROJEÇÃO**, v. 6, n. 1, p. 20-34, 2015.

DOS SANTOS SOARES, Michel. Metodologias ágeis extreme programming e *scrum* para o desenvolvimento de software. **Revista Eletrônica de Sistemas de Informação ISSN 1677-3071 doi: 10.5329/RESI**, v. 3, n. 1, 2004.

EDER, Samuel et al. Diferenciando as abordagens tradicional e ágil de gerenciamento de projetos. **Production Journal.** São Carlos, p. 498-509. out. 2015.

FADEL, Aline Cristine; SILVEIRA, Henrique da Mota. **Metodologias ágeis no contexto de desenvolvimento de software: XP, *Scrum* e Lean.** 2010. 107 f. Monografia (Especialização) - Curso de Gestão de Projetos e Qualidade, Unicamp, Campinas, 2010.

FARIAS, Marcela Soares; SANTOS, Roberta Andreza Almeida dos. **Histimate - Ferramenta de Apoio a Estimativas em Projetos de Desenvolvimento Ágeis.** 2013. 69 f. TCC (Graduação) - Curso de Informática Aplicada, Centro de Ciências Exatas e Tecnologia, Universidade Federal do Estado do Rio de Janeiro, Rio de Janeiro, 2013.

FONSECA, Isabella. Por que *SCRUM*? **Engenharia de Software Magazine**, Rio de Janeiro, v. 4, n. 1, p.1-8, abr. 2012.

GIL, Antonio Carlos. **Como Elaborar Projetos de Pesquisa.** 4. ed. São Paulo: Atlas S.a., 2002. 171 p.

JUCÁ JUNIOR, Antonio da Silva; CONFORTO, Edivandro Carlos; AMARAL, Daniel Capaldo. Maturidade em gestão de projetos em pequenas empresas desenvolvedoras de software do Polo de Alta Tecnologia de São Carlos. **Gestão & Produção,** São Carlos, v. 17, n. 1, p.181-194, out. 2010. Trimestralmente.

JUNG, Carlos Fernando. **Metodologia aplicada a projetos de pesquisa**: Sistemas de Informação & Ciência da Computação. Taquara, 2009. 1 CD-ROM.

KINIBERG, Henrik. ***Scrum* e XP Direto das Trincheiras.** Estocolmo: C4media Inc, 2007. 130 p.

LIMA, Anderson dos Santos; LEAL, Gislaine Camila Lapasini. UM DIAGNÓSTICO DO DESENVOLVIMENTO DE SOFTWARE DE UMA MPE. **Revista Tecnológica: edição especial SIMEPRO,** Maringá, p.65-71, jan. 2013. Semestral.

PAGOTTO, Tiago et al. *Scrum* Solo Processo de software para desenvolvimento individual. In: CONFERENCIA IBÉRICA DE SISTEMAS Y TECNOLOGIAS DE INFORMACIÓN, 11, 2016, Isla Gran Canaria. **Conferência.** Isla Gran Canaria: Cisti, 2016. p. 0 - 6.

PRESSMAN, Roger; MAXIM, Bruce. **Engenharia de Software-8ª Edição**. McGraw Hill Brasil, 2016.

PUNTEL, Marcio; PRASS, F. Um Método Ágil Híbrido. **Sistemas de Informação. Universidade Luterana do Brasil, Cachoeira do Sul**, 2010.

SABBAGH, Rafael. ***Scrum*:** Gestão Ágil para Projetos de sucesso. São Paulo: Casa do Código, 2014. 280 p.

SCHWABER, Ken; SUTHERLAND, Jeff. **Guia do *SCRUM***. Cidade: *Scrum*.Org and ScrumInc 2014. 19 p.

SEBRAE-NA/ Dieese. Anuário do trabalho na micro e pequena empresa 2013, p. 17.

SILVA JUNIOR, Salmo Roberto da; PIRES, Daniel Facciolo; CARVALHO NETO, Silvio. *SCRUM*: UM GUIA PRÁTICO NO GERENCIAMENTO DE PROJETOS. **Revista Eletrônica de Sistemas de Informação e de Gestão Tecnológica,** Franca, v. 5, n. 1, p.71-87, 2015.

SILVA, Edna Lucia da; MENEZES, Estera Muszkat. Metodologia da Pesquisa e Elaboração de Dissertação. 2005. 139 f. Dissertação (Mestrado) - Curso de Ciencia da Informação, Universidade Federal de Santa Catarina, Florianópolis, 2005.

SILVA, Mislene Dalila da; MELLO JUNIOR, Fernando Corrêa de. Estudo de caso: aplicação das metodologias ágeis de desenvolvimento: *Scrum* e XP no desenvolvimento do sistema Unidisciplina. **Revista Perquirere,** Patos de Minas, v. 1, n. 11, p.113-129, jul. 2014. Semestral.

SOARES, Liziane Santos. **Obtenção de requisitos para customização de processo de desenvolvimento de software.** 2007. 77 f. Dissertação (Mestrado) - Curso de Ciencia da Computação, Universidade Federal de Viçosa, Viçosa, 2007.

SOARES, Michel dos Santos. Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software. **Infocomp: Journal of Computer Science.** Conselheiro Lafaiete, p. 8-13. Dezembro. 2004.

SOUZA, Diogo Rodrigues de. **IMPLANTAÇÃO DA METODOLOGIA ÁGIL *SCRUM* EM UM AMBIENTE DE DESENVOLVIMENTO.** 2014. 59 f. TCC (Graduação) - Curso de Tecnologias da Informação e Comunicação, Universidade Federal de Santa Catarina, Pararanguá, 2014.

TANIGUCHI, Kenji; CORREA, Fernando Eugenio. Metodologias ágeis e a motivação de pessoas em projetos de desenvolvimento de software: Aplicando práticas de *SCRUM* a XP para promover a motivação de equipes de projetos de desenvolvimento

de software. **Revista de Ciências Exatas e Tecnologia,** São Paulo, v. 4, n. 4, p.163-179, dez. 2010.

TELES, V.M. Extreme Programming. 2006. Disponível em: <http://www.desenvolvimentoagil.com.br/xp>. Acesso em: 22 jul. 2016.

TELES, V.M. *SCRUM*. 2006. Disponível em: <http://www.desenvolvimentoagil.com.br/xp>. Acesso em: 22 jul. 2016.

THIRY, Marcello et al. Uma Abordagem para a Modelagem Colaborativa de Processos de Software em Micro e Pequenas Empresas. In: V SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE, 5, 2006, São José. **Uma Abordagem para a Modelagem Colaborativa de Processos de Software em Micro e Pequenas Empresas.** São José: SBQS, 2006. v. 5, p. 189 - 202.

UTIDA, Kleber Hiroki. **METODOLOGIAS TRADICIONAIS E METODOLOGIAS ÁGEIS:** ANALISE COMPARATIVA ENTRE RATIONAL UNIFIED PROCESS E EXTREME PROGRAMMING. 2012. 47 f. TCC (Graduação) - Curso de Tecnologia em Processamento de Dados, Faculdade de Tecnologia do Estado de São Paulo.

Wildt, D.F. e Lacerda, G.S. (2010) “Conhecendo o eXtreme Programming (XP)”, http://www.slideshare.net/dwildt/conhecendo-o-extreme-programming. Acessado em 13 de outubro de 2013.