UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ – UTFPR
DIRETORIA DE PESQUISA E PÓS-GRADUAÇÃO
ESPECIALIZAÇÃO EM ENGENHARIA DE SOFTWARE

MARCELA TURIM KOSCHEVIC

GERENCIAMENTO DE PROCESSOS COM METODOLOGIAS ÁGEIS

MONOGRAFIA DE ESPECIALIZAÇÃO

MEDIANEIRA
2011

MARCELA TURIM KOSCHEVIC

GERENCIAMENTO DE PROCESSOS COM METODOLOGIAS ÁGEIS

Monografia apresentada como requisito parcial à obtenção do título de Especialista na Pós Graduação em Engenharia de *Software*, da Universidade Tecnológica Federal do Paraná – UTFPR – Campus Medianeira.

Orientador: Prof Msc. Everton Coimbra de Araújo.

MEDIANEIRA

2011

Dedico este estudo aos meus queridos colegas
que sempre me deram força para continuar.

AGRADECIMENTOS

À Deus pelo dom da vida, pela fé e perseverança para vencer os obstáculos.

Aos meus pais, pela orientação, dedicação e incentivo nessa fase do curso de pós-graduação e durante toda minha vida.

Ao meu orientador, professor Everton, que me orientou, pela sua disponibilidade, interesse e receptividade com que me recebeu e pela prestabilidade com que me ajudou.

Agradeço aos colegas de trabalho que auxiliaram na execução das atividades propostas para a realização do estudo de caso.

Enfim, sou grata a todos que contribuíram de forma direta ou indireta para realização desta monografia.

"Porque eu sou do tamanho do que vejo
E não do tamanho da minha altura...
E o que vejo são meus sonhos."

(Alberto Caeiro, heterônimo de Fernando Pessoa)

RESUMO

KOSCHEVIC, Marcela Turim. GERENCIAMENTO DE PROCESSOS COM METODOLOGIAS ÁGEIS. 2011. 33 folhas. Monografia (Especialização em Engenharia de *Software*). Universidade Tecnológica Federal do Paraná, Medianeira, 2011.

Este trabalho tem por objetivo relatar um estudo de caso envolvendo gerenciamento de processos com aplicação de metodologias ágeis.

Palavras-chave: Métodos ágeis para desenvolvimento e gerenciamento de *Software*, Processos ágeis.

ABSTRACT

KOSCHEVIC, Marcela Turim. Process Management with agile methodologies. 2011. 33. Monograph (Specialization in Software Engineering).Universidade Tecnológica Federal do Paraná, Medianeira, 2011.

This paper aims at reporting a case study involving process management with application of agile methodologies.

Keywords: Agile methods for development and software management, Agile processes.

LISTA DE FIGURAS

Lista de Tabelas

SUMÁRIO

1 INTRODUÇÃO

O setor de Educação a Distância da UTFPR, campus Medianeira, é responsável pelo suporte e desenvolvimento de *softwares* para a modalidade de ensino a distância. Estes softwares são utilizados em todos os câmpus da UTFPR que atuam nesse modelo de ensino.

Atualmente este processo de desenvolvimento é pouco gerenciado, fato este que reflete na qualidade dos produtos desenvolvidos. Um dos principais problemas é o cumprimento dos prazos estipulados pela coordenação. Outra área problemática é a área de testes e suporte, a qual por muitas vezes está sobrecarregada devido ao grande volume de atividades.

A proposta deste trabalho é estudar e implantar uma metodologia de trabalho para que a equipe permaneça organizada e consiga entregar os trabalhos dentro do prazo estabelecido e com um nível de qualidade aceitável.

1.1 OBJETIVO GERAL

Implantar uma metodologia de desenvolvimento capaz de tornar a equipe de suporte técnico/desenvolvimento do EaD UTFPR, câmpus Medianeira, auto gerenciável e comprometida com as atividades que estão sendo realizadas.

1.2 OBJETIVOS ESPECÍFICOS

Estudar as abordagens ágeis mais utilizadas para gerenciamento de projetos e de equipes;

Criar uma metodologia específica para a equipe em questão;

Apresentar a metodologia para a equipe;

Aplicar esta metodologia ao caso.

Analisar e documentar os resultados.

1.3 JUSTIFICATIVA

Segundo Paula Torreão (TORREÃO, 2006), um projeto para ser executado precisa ser gerenciado. Sem uma gerência, um projeto ou uma equipe pode facilmente se perder em meio a sua demanda de atividades.

Segundo Koontz e O'Donnel (KOONTZ; O'DONNEL, 1980) citado por Paula Torreão (TORREÃO, 2006), gerenciar consiste em executar atividades e tarefas que têm como propósito planejar e controlar atividades de outras pessoas para atingir objetivos que não podem ser alcançados caso as pessoas atuem por conta própria, sem o esforço sincronizado dos subordinados. Este planejamento tem o objetivo de organizar as atividades da equipe e serve também como uma forma de apresentar aos interessados aquilo que está sendo construído.

Ainda segundo Paula Torreão (TORREÃO, 2006), a gestão de projetos envolve criar um equilíbrio entre as demandas de escopo, tempo, custo, qualidade e bom relacionamento com o cliente. O papel do gerente de projetos é justamente cuidar deste equilíbrio e garantir de alguma forma que as demandas sejam cumpridas.

O sucesso na gestão de um projeto está relacionado ao alcance dos seguintes objetivos: entrega dentro do prazo previsto, dentro do custo orçado, com nível de desempenho adequado, aceitação pelo cliente, atendimento de forma controlada às mudanças de escopo e respeito à cultura da organização (PMI 2000). Nestes pontos, é importante perceber que a equipe que desenvolve o produto é também a parte crucial para a finalização e aceitação do produto. Uma equipe motivada, treinada e bem gerenciada conseguirá atingir seus objetivos e por consequência, os objetivos da organização a qual pertence.

Métodos, práticas e técnicas para o desenvolvimento ágil de projetos prometem aumentar a satisfação do cliente (BOEHM, 2003) para produzir alta qualidade de *software* e acelerar os prazos de desenvolvimento de projetos (ANDERSON, 2003). No entanto, o foco deve estar sempre na equipe e em como ela está desenvolvendo o produto, o que está bom e o que pode ser melhorado.

1.4 ESTRUTURA DO TRABALHO

O capítulo 1 deste trabalho apresenta a introdução, os objetivos e a justificativa do trabalho.

O capítulo 2 apresenta as metodologias de desenvolvimento de *software*, os métodos ágeis e o Scrum.

O capítulo 3 apresenta o gerenciamento de projetos e de pessoas.

O capítulo 4 apresenta o estudo de caso do trabalho.

O capítulo 5 apresenta as considerações finais e trabalhos futuros.

2 METODOLOGIAS ÁGEIS DE DESENVOLVIMENTO DE *SOFTWARE*

De acordo com Boehm (BOEHM, 2003), estamos vivendo uma tendência para o desenvolvimento ágil de aplicações devido a um ritmo acelerado de mudanças e inovações na área de tecnologia da informação, em organizações e no ambiente de negócios.

Segundo Ana Sofia (MARÇAL et. al, 2011), esse ritmo acelerado de mudanças tem causado frustrações crescentes com planos, especificações e documentações pesados muitas vezes impostos por critérios de conformidade dos modelos de maturidade. Ana Sofia (MARÇAL et. al, 2011) ainda ressalta que Boehm afirma que no final dos anos 90 acompanhou-se o surgimento de vários métodos ágeis, entre eles: *Adaptive Software Development*, *Crystal*, *Dynamic Systems Development*, *eXtreme Programming* (XP), *Feature Driven Development* e *Scrum*.

Todos esses métodos empregam princípios ágeis, tais como ciclos iterativos, entrega rápida de *software* funcionando e simplicidade, como definido no Manifesto para Desenvolvimento Ágil publicado em 2001.

A essência do movimento para desenvolvimento ágil é a definição de novo enfoque de desenvolvimento de *software*, calcado na agilidade, na flexibilidade, nas habilidades de comunicação e na capacidade de oferecer novos produtos e serviços de valor ao mercado, em curtos períodos de tempo.

Juntamente com o manifesto ágil, surgiu também o APM (*Agile Project Management*) que representa um conjunto de valores, princípios e práticas, que auxiliam a equipe de projeto a entregar produtos ou serviços de valor em um ambiente desafiador. De acordo com Ana Sofia (MARÇAL *et. al,* 2011), como agilidade deve-se entender "*a habilidade de criar e responder a mudanças, buscando a obtenção de lucro em um ambiente de negócio turbulento*"; ou ainda, a capacidade de balancear a flexibilidade e a estabilidade. Métodos, práticas e técnicas ágeis para desenvolvimento de *software* prometem incrementar a satisfação do cliente produzindo *software* com maior qualidade e acelerando o tempo de desenvolvimento.

É perceptível que as empresas e organizações querem *software* de uma maneira rápida, geralmente o mais breve possível. Segundo Sommerville (SOMMERVILLE, 2011), muitas empresas estão dispostas a trocar a qualidade e o

compromisso com requisitos de *software* por uma implementação mais rápida, com uma entrega em tempo menor.

Processos de desenvolvimento de *software* que planejam especificar completamente os requisitos para depois implementar e testar o sistema não são considerados aptos para o desenvolvimento ágil de *software* segundo Sommerville (SOMMERVILLE, 2011). Se, por exemplo, acontecerem mudanças nos requisitos, ou ocorrer a descoberta de algum problemas nos mesmos, o projeto e na pior das hipóteses, a implementação, precisará ser refeito ou retestado.

Outra questão a ser considerada no desenvolvimento de *software* acontece quando a equipe de desenvolvimento não utiliza metodologia alguma. A questão é por onde começar? O que fazer para que essa equipe produza *software* de qualidade e no tempo concedido para ela? São questões difíceis para um membro da equipe desenvolver, mas quando há participação de todos, certamente o processo ficará mais tangível.

Segundo Sommerville (SOMMERVILLE, 2011), os processos de desenvolvimento rápido de *software* são concebidos para produzir, rapidamente *softwares* úteis. Existem muitas abordagens para o desenvolvimento ágil, no entanto elas compartilham algumas características:

Os processos de especificação, projeto e implementação são intercalados. Não há especificação detalhada do sistema e geralmente a documentação é minimizada e gerada automaticamente pelo ambiente de programação utilizado pela equipe de desenvolvimento. O documento de requisitos do usuário apenas define as características mais importantes do sistema. (SOMMERVILLE, 2011).

O sistema é desenvolvido em uma série de versões. Os usuários finais e *stakeholders* do sistema são envolvidos na especificação e na avaliação de cada versão. Eles tem autonomia para propor alterações ao *software* ou então, para especificar novos requisitos que deverão ser implementados em uma versão posterior. (SOMMERVILLE, 2011).

Geralmente, as interfaces gráficas dos sistemas são desenvolvidas utilizando ferramentas interativas, que permitem a criação rápida do projeto da interface gráfica. (SOMMERVILLE, 2011).

Os métodos ágeis são métodos incrementais, onde geralmente os itens incrementados são pequenos e normalmente as novas versões são disponibilizadas em duas ou três semanas. Essas metodologias envolvem os clientes no processo

como uma forma de obter retorno rápido sobre a evolução dos requisitos. Dessa forma, minimiza-se a documentação, pois existe uma comunicação informal entre os interessados no sistema.

2.1 MÉTODOS ÁGEIS

Segundo Sommerville (SOMMERVILLE, 2011), nas décadas de 1980 e 1990, existia uma visão construída por engenheiros de *software* de que os melhores *softwares* eram aqueles desenvolvidos com base em um planejamento cuidadoso do projeto, com qualidade de segurança formalizada, com uso de análise e projeto apoiado por ferramentas CASE (*Computer-aided software engineering*), além de apresentarem um processo rigoroso e controlado de desenvolvimento. Essa percepção vem mais especificamente da comunidade responsável pelo desenvolvimento de grandes sistemas de *software*, como por exemplo, sistemas aeroespaciais e governamentais.

No entanto, ainda segundo Sommerville, quando essa abordagem robusta e pesada de desenvolvimento é aplicada a sistemas de médio e pequeno porte, existe uma sobrecarga muito grande e que acaba dominando o processo de desenvolvimento de *software*. O ponto chave, é que acaba-se gastando muito tempo com a análise do sistema e pouco tempo com programação e testes.

De acordo com Sommerville (SOMMERVILLE, 2011) a insatisfação com essas abordagens pesadas levou um grande número de desenvolvedores a proporem novos métodos ágeis. Estes métodos permitiam que a equipe de desenvolvimento focasse no *software* em si, e não em sua concepção e documentação.

Estes métodos são bastante adequados onde o desenvolvimento de aplicativos agrega muitas mudanças de requisitos ao longo do processo. Essas metodologias destinam-se àqueles que precisam entregar um *software* rapidamente aos clientes, em funcionamento e com a possibilidade de alterações e criação de novos requisitos.

O objetivo dessas metodologias ágeis é , segundo Sommerville (SOMMERVILLE, 2011), reduzir a burocracia do processo, evitando qualquer trabalho de valor incerto de longo prazo e também, evitar qualquer documentação que provavelmente nunca será usada.

O manifesto ágil é responsável pela filosofia que existe por de trás dos métodos ágeis. Neste manifesto ficou acordado entre os desenvolvedores de *software* que é necessário valorizar mais:

1. Indivíduos e interações do que processos e ferramentas;
2. *Software* em funcionamento do que documentação abrangente;
3. Colaboração do cliente do que negociação de contrato;
4. Resposta a mudanças do que seguir um plano.

Por mais que os itens a direita sejam importantes, é necessário valorizar mais o que está a esquerda.

Existem diversas metodologias ágeis, sendo que, segundo Sommerville (SOMMERVILLE, 2011), a *Extreme Programming* é uma das mais conhecidas. Outras abordagens incluem Scrum e Crystal, dentre outros.

De acordo com Sommerville (SOMMERVILLE, 2011), embora esses métodos ágeis sejam todos baseados em desenvolvimento e entregas incrementais, eles propõem diferentes processos para alcançar tal objetivo. Como compartilham um conjunto de princípios advindos do manifesto ágil, têm bastante coisa em comum.

2.2 GERENCIAMENTO ÁGIL DE PROJETOS

Segundo Sommerville (SOMMERVILLE, 2011), a principal responsabilidade dos gerentes de projetos de *software* é gerenciar o projeto de tal maneira que este seja entregue no prazo e dentro do orçamento previsto. Os gerentes devem supervisionar o trabalho dos desenvolvedores de *softwares* e acompanham o quão bem o desenvolvimento de *software* está progredindo.

Ainda de acordo com Sommerville (SOMMERVILLE, 2011), o desenvolvimento ágil tem de ser gerenciado de maneira a usar de melhor forma o tempo e os recursos disponíveis para a equipe. Desse modo, é necessária uma abordagem diferente, adaptada para a forma de desenvolvimento incremental e com suporte aos pontos fortes dos métodos ágeis.

A abordagem Scrum é um método ágil geral e seu foco está no gerenciamento de *software* iterativo e não é uma técnica específica da engenharia de *software*.

2.3 SCRUM

O Scrum, de acordo com Ana Sofia (MARÇAL *et. al*, 2011) é um método que aceita que o desenvolvimento de *software* é imprevisível e formaliza a abstração, sendo aplicável a ambientes voláteis. Ele se destaca dos demais métodos ágeis pela maior ênfase dada ao gerenciamento do projeto. Reúne atividades de monitoramento e *feedback*, em geral, reuniões rápidas e diárias com toda a equipe, visando à identificação e correção de quaisquer deficiências e/ou impedimentos no processo de desenvolvimento.

O Scrum assume a premissa de que o desenvolvimento de *software* é bastante complexo e imprevisível para ser planejado totalmente no início. É sugerido então, que ao invés disso, seja usado controle do processo empírico para garantir a visibilidade, inspeção e adaptação.

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

2.3.1 Papéis e Responsabilidades

O Scrum, segundo Ana Sofia [MARÇAL et. al, 2011], implementa um esqueleto iterativo e incremental através de três papéis principais:

1. ***Product Owner***: representa os interesses de todos no projeto; define os fundamentos do projeto criando requisitos iniciais e gerais (*Product Backlog*), retorno do investimento (ROI), objetivos e planos de entregas; prioriza o *Product Backlog* a cada *Sprint*, garantindo que as funcionalidades de maior valor sejam construídas prioritariamente.
2. ***Scrum Master***: Gerencia o processo do *Scrum*, ensinando o *Scrum* a todos os envolvidos no projeto e implementando o *Scrum* de modo que esteja adequado a cultura da organização; deve garantir que todos sigam as regras e práticas do Scrum; é responsável por remover os impedimentos do projeto.

3. **Time**: desenvolve as funcionalidades do produto; define como transformar o *Product Backlog* em incremento de funcionalidades numa iteração gerenciando seu próprio trabalho sendo responsáveis coletivamente pelo sucesso da iteração e consequentemente pelo projeto como um todo.

2.3.2 Fases do Scrum

O Scrum, de acordo com Ana Sofia (MARÇAL et. al, 2011), possui um ciclo de vida composto por quatro fases:

1. **Planejamento:** estabelecer a visão do projeto e expectativas garantindo recursos para a sua execução. Nesta fase são criadas as versões iniciais do *Product Backlog* e o plano de release, arquitetura de negócio e técnica em alto nível.
2. ***Stagging*:** avaliar as várias dimensões do projeto criando itens adicionais ao *Product Backlog* relacionados com o tipo do sistema, time, ambiente de desenvolvimento, tipo de aplicação. Nesta fase os **Times** são formados e são construídos os mecanismos de comunicação e coordenação entre eles.
3. **Desenvolvimento:** consiste de múltiplas *Sprints* para o desenvolvimento dos incrementos de funcionalidade do produto.
4. ***Releasing*:** realizar a entrega do produto ao cliente.

Sommerville (SOMMERVILLE, 2011) considera que o Scrum possui três fases apenas: planejamento geral, ciclos de *Sprints* e o encerramento do projeto.

De acordo com Sommerville (SOMMERVILLE, 2011), a característica inovadora do Scrum é justamente a fase central, onde é são construídas as *Sprints*. Uma *Sprint* é uma unidade de planejamento na qual o trabalho a ser feito é avaliado, os recursos para o desenvolvimento são selecionados e o *software* é implementado. Ao final de uma *Sprint*, a funcionalidade completa é entregue aos *stakeholders*.

Segundo Sommerville (SOMMERVILLE, 2011), as principais características desse processo são:

1. As *Sprints* devem ter tamanho fixo, normalmente de duas a quatro semanas.

2. O ponto de partida (também chamado *kickoff*) para o planejamento é *backlog* do produto. Este *backlog* consiste em uma lista do trabalho a ser feito no projeto. Durante a fase de avaliação da *Sprint*, este documento é revisto e as prioridades e riscos são identificados. Os clientes devem estar envolvidos nessa fase do processo. No início de cada *Sprint*, podem ser introduzidos novos requisitos ou tarefas.
3. Todos os itens do *backlog* devem ser vistos pelos envolvidos no processo e a equipe de desenvolvimento selecionará os itens a serem implementados nessa *Sprint*. Uma vez que todos estejam de acordo, a equipe se organiza para desenvolver o *software*.
4. Devem ser realizadas reuniões diárias rápidas, envolvendo todos os membros da equipe. Dessa forma, é possível, se necessário, repriorizar o trabalho, segundo Sommerville (SOMMERVILLE, 2011).
5. Ao final da *Sprint*, o trabalho é revisto e apresentado aos *stakeholders*. Em seguida, deverá ser iniciada a próxima *Sprint*.

A **Erro! Fonte de referência não encontrada.** abaixo ilustra o processo do Scrum:

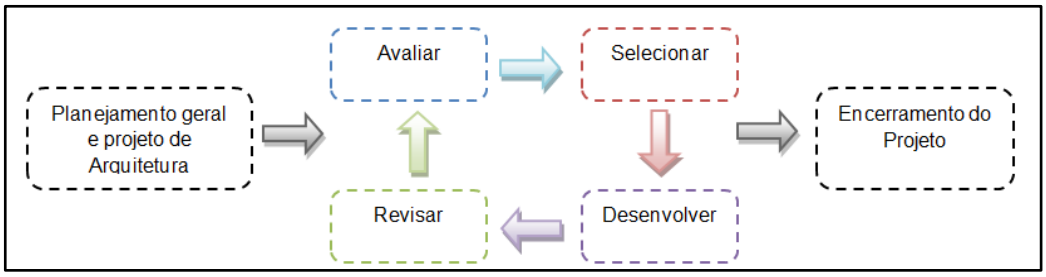

Figura 1 – O processo Scrum
Fonte: Sommerville (2011, p. 50).

O Scrum permite que toda a equipe possua poderes para tomar decisões, de modo que o termo “gerente de projeto” geralmente seja evitado. Aqui entra a figura do *Scrum Master*, que é um facilitador. Ele organiza reuniões diárias, controla o *backlog* e se comunica com os clientes e a gerência externa à equipe, de acordo com Sommerville (SOMMERVILLE, 2011).

Ainda segundo Sommerville (SOMMERVILLE, 2011), toda a equipe deve participar das reuniões diárias. Estas reuniões geralmente acontecem com os participantes em pé (*stand-up*) e são reuniões muito rápidas para que a equipe não

se disperse. Durante a reunião, todos os membros da equipe devem relatar seu progresso desde a última reunião, os problemas que estão surgindo e o que estão planejando para o dia seguinte. Dessa forma, garante-se que todos os participantes conheçam os trabalhos e problemas alheios, e se por ventura algum dos membros conhecer a solução para um problema relatado por outro membro, este tem a liberdade de poder ajudar. Assim, também é possível replanejar o trabalho a curto prazo. Não existe hierarquia *top-down* a partir do *Scrum Master*.

O Scrum foi projetado para equipes onde os membros trabalham fisicamente em um mesmo lugar, onde todos poderiam se encontrar todos os dias em reuniões diárias, de acordo com Sommerville (SOMMERVILLE, 2011).

3 GERENCIAMENTO DE PROJETOS

O gerenciamento de projetos de *software* é uma parte fundamental da engenharia de *software*. Todo projeto deve ser gerenciado pois todo projeto está condicionado a orçamentos organizacionais e restrições de cronograma.

O trabalho do gerente de projetos, segundo Sommerville (SOMMERVILLE, 2011), é garantir que o projeto de *software* atenda e supere estas restrições, além de oferecer um *software* de alta qualidade.

O sucesso de um projeto não é garantido por um bom gerenciamento, no entanto, a má gerência, costuma resultar em falha do projeto, de acordo com Sommerville (SOMMERVILLE, 2011).

Geralmente, para um gerente de projetos, as metas mais importantes são:

1. Fornecer o *software* ao cliente no prazo estabelecido.
2. Manter os custos do *software* dentro do estimado no orçamento.
3. Entregar um *software* que atenda às expectativas do cliente.
4. Manter a equipe de desenvolvimento trabalhando bem e feliz.

Esses objetivos não são exclusivos da engenharia de *software*, são objetivos gerais de todos os tipos de projetos. No entanto algumas diferenças são perceptíveis.

O produto da engenharia de *software*, por exemplo, não é tangível. Seu progresso não pode ser visto apenas olhando o artefato que está sendo construído. O *software* depende de outros artefatos e pessoas, que revisem o seu progresso constantemente, buscando verificar se o produto está sendo construído de forma adequada.

Não é possível, de acordo com Sommerville (SOMMERVILLE, 2011), construir uma descrição padrão para um gerente de projetos de *software*. No entanto, a maioria dos gerentes assumem certas responsabilidades ao longo de sua jornada. São elas:

1. Fazer o planejamento do projeto;
2. Emitir relatórios sobre o andamento dos projetos;
3. Fazer o gerenciamento de riscos;
4. Gerenciar as pessoas de sua equipe;
5. Elaborar propostas descrevendo os objetivos do projeto e como ele será executado.

3.1 GERENCIAMENTO DE RISCOS

Segundo Sommerville (SOMMERVILLE, 2011), o gerenciamento de riscos é um dos trabalhos mais importantes para um gerente de projetos. Essa atividade envolve antecipar os riscos que podem afetar o cronograma do projeto ou a qualidade do *software* que se deve entregar. Caso forem encontrados riscos, estes devem ser mitigados.

Os riscos podem ameaçar o projeto, o *software* ou a organização. Conforme Sommerville adota, existem três categorias de riscos relacionados:

1. Riscos de projeto;
2. Riscos de produto;
3. Riscos de negócio;

Os riscos de projeto são aqueles que afetam o cronograma ou os recursos do projeto.

Os riscos de produto são aqueles que afetam a qualidade ou o desempenho do *software* que está sendo desenvolvido.

Os riscos de negócio são aqueles que afetam a organização que desenvolve ou adquire o *software*.

Segundo Sommerville (SOMMERVILLE, 2011) é importante documentar os resultados do processo de gerenciamento de riscos em um plano de gerenciamento de riscos. Neste plano, deve ser incluída uma discussão acercados riscos encontrados no projeto e uma análise desses riscos. É importante obter informações sobre a maneira de gerenciar o risco caso seja provável que ele se torne um problema.

O processo do gerenciamento de riscos é um processo iterativo que continua ao longo de todo o projeto, segundo Sommerville (SOMMERVILLE, 2011).

3.2 GERENCIAMENTO DE PESSOAS

De acordo com Sommerville (SOMMERVILLE, 2011), é importante que os gerentes de projeto de *software* compreendam as questões técnicas que influenciam o trabalho de desenvolvimento de *software*. Geralmente, pode acontecer que

pessoas da área técnica de desenvolvimento, podem não ter as habilidades mais flexíveis que lhes permitam motivar e liderar uma equipe de desenvolvimento. É necessário que o gerente de projetos esteja ciente dos problemas potenciais de gerenciamento de pessoas.

Sommerville (SOMMERVILLE, 2011) afirma que há quatro fatores críticos no gerenciamento de pessoas:

1. Consistência
2. Respeito
3. Inclusão
4. Honestidade

A consistência significa que todas as pessoas de uma equipe devem ser tratadas da mesma forma. As pessoas não devem sentir que sua contribuição para a organização é subvalorizada.

Todas as pessoas são diferentes e possuem habilidades diferentes. Um gerente de projetos deve respeitar isso. Todos os membros devem ter oportunidade de contribuir.

As pessoas contribuem efetivamente quando sentem que são ouvidas por outras pessoas e que suas propostas são levadas a sério. É importante desenvolver um ambiente onde as pessoas se sintam incluídas verdadeiramente no projeto.

Um gerente de projetos deve sempre ser honesto sobre o que está indo bem e o que não está tão bem em uma equipe. Caso os problemas sejam ignorados, um gerente de projetos pode inclusive perder o respeito do grupo.

Sommerville (SOMMERVILLE, 2011) considera ainda que, o gerenciamento de pessoas deve ser baseado na experiência, pois é com experiência que é possível saber como agir diante de determinadas situações.

3.2.1 Motivação das pessoas

Um gerente de projetos, segundo Sommerville (SOMMERVILLE, 2011), deve motivar as pessoas que trabalham com ele para que elas contribuam com o melhor de suas habilidades.

Motivação significa organizar o trabalho e o ambiente para incentivar as pessoas a trabalharem do modo mais eficaz possível. Se as pessoas estão motivadas elas se interessam pelo trabalho que está sendo desenvolvido. Senão,

elas trabalharam de forma lenta, desinteressadas e mais propensas a cometer erros, de acordo com Sommerville (SOMMERVILLE, 2011).

Para fornecer incentivos às pessoas, um gerente de projetos deve entender um pouco sobre o que motiva as pessoas. Sommerville (SOMMERVILLE, 2011), citando Maslow (MASLOW, 1954), sugere que as pessoas são motivadas por satisfazer suas necessidades. Um gerente de projetos deve certificar-se que as necessidades sociais, de autoestima e de autorrealização das pessoas estão sendo satisfeitas no ambiente de trabalho.

Sommerville (SOMMERVILLE, 2011) afirma que para satisfazer as necessidades sociais, um gerente de projetos precisa dar tempo às pessoas para se encontrarem com os colegas de trabalho e lhes oferecerem oportunidade para isso.

Para satisfazer as necessidades de autoestima, um gerente de projetos deve mostrar as pessoas que elas são valorizadas pela organização, e isso é possível através do reconhecimento público.

A autorrealização significa que um gerente de projetos deve dar as pessoas responsabilidade por seu trabalho, atribuir-lhes tarefas exigentes, mas não impossíveis e fornecer treinamentos para que elas possam desenvolver suas habilidades. De acordo com Sommerville (SOMMERVILLE, 2011), o treinamento é uma importante influência motivadora para as pessoas, assim como a aquisição de novos conhecimentos e novas habilidades.

Tornar-se membro de um grupo coeso é algo altamente motivador para a maioria das pessoas. Segundo Sommerville (SOMMERVILLE, 2011), as pessoas com empregos satisfatórios, muitas vezes gostam de ir trabalhar porque são motivadas pelas pessoas com quem trabalham e pelo trabalho que realizam. Um gerente de projetos, além de pensar na motivação individual, deve pensar na motivação do grupo, como um todo, pois isso pode garantir que os objetivos da organização sejam atingidos.

Os tipos de personalidade, de acordo com Sommerville (SOMMERVILLE, 2011), também influenciam na motivação. De acordo com Bass e Dunteman (BASS & DUNTEMAN, 1963), citados por Sommerville (SOMMERVILLE, 2011), os profissionais são classificados em três tipos:

1. Pessoas orientadas a tarefas
2. Pessoas automotivadas
3. Pessoas orientadas a interação

As pessoas orientadas a tarefas são motivadas pelo trabalho que fazem, geralmente são pessoas motivadas pelo desafio intelectual.

Pessoas automotivadas são aquelas motivadas pelo reconhecimento pessoal. Geralmente essas pessoas têm objetivos em longo prazo, como progressão na carreira e desejam ser bem-sucedidas em seus trabalhos para concretizar tais objetivos.

Pessoas orientadas a interações são motivadas pela presença e ações dos colegas de trabalho. Geralmente essas pessoas gostam de trabalhar como parte de um grupo, enquanto que pessoas automotivadas e pessoas orientadas a tarefas geralmente preferem trabalhar sozinhas.

A motivação de cada indivíduo é constituída por elementos de cada classe, mas normalmente um tipo de motivação é dominante em um determinado momento, segundo Sommerville (SOMMERVILLE, 2011). Lembrando que os indivíduos podem mudar ao longo do tempo, o que pede que o gerente de projetos sempre esteja atendo a sua equipe para que saiba identificar o momento de seus colaboradores.

3.3 GERENCIAMENTO DE EQUIPE

De acordo com Sommerville (SOMMERVILLE, 2011), em grandes organizações, com grandes projetos, às vezes é bastante complicado garantir uma boa comunicação entre os membros da equipe.

Em uma equipe pequena de desenvolvimento, geralmente os problemas de comunicação são reduzidos.

Segundo Sommerville (SOMMERVILLE, 2011), montar uma equipe que tenha o equilíbrio entre as habilidades técnicas, a experiência e as personalidades é uma tarefa de gerenciamento crítico. Os grupos bem sucedidos consistem de pessoas em equilíbrio, coesas e que tem espírito de equipe. É como se a equipe fosse um time, todos tem suas responsabilidades, mas o resultado é obtido pelo grupo, pela formação. As pessoas envolvidas são motivadas pelo sucesso do grupo e também pelos seus objetivos pessoais.

Os membros de um grupo coeso trabalham como se o grupo fosse mais importante do que seus indivíduos. De acordo com Sommerville (SOMMERVILLE, 2011), estes membros se identificam e com os outros membros do grupo. Eles tentam proteger o grupo de influências externas como se o grupo fosse uma

entidade. Para Sommerville (SOMMERVILLE, 2011), isso torna o grupo forte e capaz de lidar com problemas e situações inesperadas.

Os benefícios de criar um grupo coeso são:

1. O grupo pode estabelecer seus próprios padrões de qualidade.
2. As pessoas se apoiam e aprendem umas com as outras.
3. O conhecimento é compartilhado.
4. Melhorias contínuas são incentivadas pelos próprios participantes dos grupos.

Estabelecer seus próprios padrões de qualidade, é importante pois quem está interno ao grupo consegue observar melhor os padrões do que membros externos.

Geralmente, em um grupo, as pessoas aprendem umas com as outras. A aprendizagem mutua acaba sendo encorajada pelos próprios participantes.

Caso um membro da equipe deixar o grupo, outros membros devem ter a capacidade de assumir tarefas críticas e devem garantir que o projeto continue.

Os bons gerentes, de acordo com Sommerville (SOMMERVILLE, 2011), devem sempre incentivar a coesão do grupo. Uma das maneiras mais eficazes de promover a coesão é ser um gerente inclusivo. Isso significa que todos os membros do grupo devem ser tratados como responsáveis e confiáveis. As informações devem ser disponibilizadas livremente a todos os participantes dos grupos.

Segundo Sommerville (SOMMERVILLE, 2011), existem três fatores genéricos que afetam o trabalho da equipe:

1. As pessoas no grupo.
2. A organização do grupo.
3. Comunicações técnicas e gerenciais.

Todo projeto acaba precisando de várias pessoas, com personalidades diferentes e com necessidades diferentes. Um grupo deve ser organizado de maneira que os indivíduos possam contribuir com o melhor de suas habilidades e as tarefas podem ser concluídas no prazo esperado.

Sommerville (SOMMERVILLE, 2011) escreve que assim como em qualquer projeto, ter a equipe certa, não garante o sucesso do projeto. No entanto ela pode ser um elemento chave e se bem trabalhada, aumenta a probabilidade de sucesso no projeto.

4 ESTUDO DE CASO

Em setembro de 2011, foi proposta a implantação de alguma metodologia de trabalho que pudesse organizar e avaliar o trabalho da equipe de suporte técnico e desenvolvimento do EaD-UTFPR, câmpus Medianeira. Esta implantação originou o estudo de caso descrito neste trabalho.

4.1 IMPLANTAÇÃO DO SCRUM NO AMBIENTE DE DESENVOLVIMENTO E SUPORTE DO EAD-UTFPR, CÂMPUS MEDIANEIRA

Inicialmente não havia sido especificada uma metodologia de desenvolvimento de software para a realização dos trabalhos da equipe de suporte/desenvolvimento do EaD-UTFPR, câmpus Medianeira , no entanto, foi sugerido que a metodologia deveria ser prática, fácil de aprender e que pudesse ser incorporada sem prejudicar o andamento dos projetos.

Em estudos realizados, foi identificado que a metodologia Scrum poderia ser adaptada para a realidade da equipe EaD-UTFPR, câmpus Medianeira. A ideia do Scrum é ser uma metodologia iterativa e incremental para o desenvolvimento de *software* e também para o gerenciamento de equipes.

O desenvolvimento incremental é uma estratégia de planejamento onde várias partes do sistema são desenvolvidas em paralelo, e integradas quando completas. Já o desenvolvimento iterativo é uma estratégia de planejamento de trabalho em que o tempo de revisão e melhorias de partes do sistema é pré-definido ainda no planejamento.

Além do estudo da metodologia Scrum, foi necessário um estudo abrangente sobre gerenciamento de projetos e gerenciamento de pessoas. A união desses três pontos, metodologia Scrum, gerenciamento de projetos e gerenciamento de pessoas, foi o alicerce para organização da equipe nos projetos.

4.2 FASE INICIAL

Durante a fase inicial da implantação, foram realizadas algumas reuniões com a equipe. Nestas reuniões foram explicados os objetivos da implantação de uma metodologia de desenvolvimento e também foram coletadas informações

vindas da própria equipe, sugestões sobre os processos e sobre a própria organização da equipe.

De maneira geral, houve uma boa aceitação por parte da equipe para a implantação da metodologia Scrum.

Em uma reunião inicial, a equipe definiu o tamanho das *Sprints* e a pontuação dos itens de *backlog*. Para auxílio na implantação da metodologia, foi utilizado um *software* online chamado ClockingIt. O sistema ClockingIt, de acordo com Miguel Galves (GALVES, 2008), permite o cadastro de diversos projetos e colaboradores. Em cada projeto, é possível criar *milestones*[1] e tarefas. Estas tarefas possuem diversos graus de prioridade, nível de dificuldade, tempo estimado de execução e data de entrega. Para cada tarefa, é necessário designar um colaborador para sua execução. Além disso, o sistema permite a criação de subtarefas e permite registrar o tempo levado para executar cada tarefa, gerando um log de trabalho. Além de tarefas e subtarefas, é possível cadastrar novas características, defeitos e melhorias. Com todos estes dados, é possível gerar relatórios diversos, como *timesheets*[2] que são extremamente úteis para consultoria. O sistema ainda oferece várias perspectivas de visualização de todos estes dados: *timeline*, *schedule*, lista de tarefas.

A própria equipe de suporte/desenvolvimento reuniu-se com o intuito de definir prioridade e importância das atividades cadastradas no ambiente ClockingIt[3].

A Figura 2 ilustra a tela principal do ambiente ClockingIt.

[1] Também chamada de marco, é uma expressão inglesa utilizada como designação de um ponto de controle em um cronograma, através da definição de pontos de checagem ou marcos de desenvolvimento.

[2] É um método para verificar a quantidade de tempo que um trabalhador gasta em cada trabalho.

[3] A ferramenta ClockingIt é um sistema online gratuito, disponível no endereço http://www.clockingit.com/.

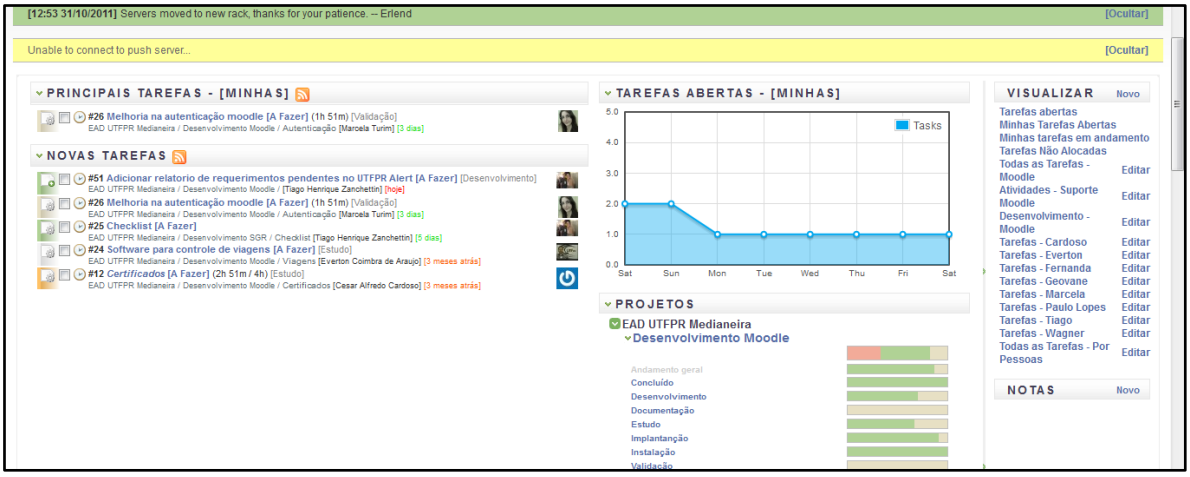

Figura 2 – Tela principal do ClockingIt

No ambiente ClockingIt, cada membro da equipe possui um perfil e este perfil tem acesso aos projetos que estão sendo desenvolvidos pela equipe. Todos têm permissão de visualizar os projetos, fazer comentários ou criar novas tarefas. Esse método foi adotado para que todos os membros da equipe possam participar de maneira semelhante no desenvolvimento dos projetos. Qualquer membro pode sugerir uma nova funcionalidade, ou então cadastrar um *bug*.

A equipe definiu as prioridades e importâncias de acordo com a Tabela 1:

Tabela 1 – Prioridades e Importâncias

Prioridade	Importância
A mais baixa	Trivial
Baixa	Menor
Normal	Normal
Alta	Principal
Urgente	Crítico
Crítica	Muito grave

As prioridades são maneiras de priorizar o trabalho, ou seja, os itens críticos devem ser resolvidos antes dos itens normais ou urgentes, por exemplo. A importância define um grau de importância do item para o projeto. Um item muito grave é mais importante que um item normal. A junção de prioridade e importância é o que define a ordem de priorização dos itens a serem implementados ou feitos. A Figura 3 ilustra uma tela de criação de tarefa:

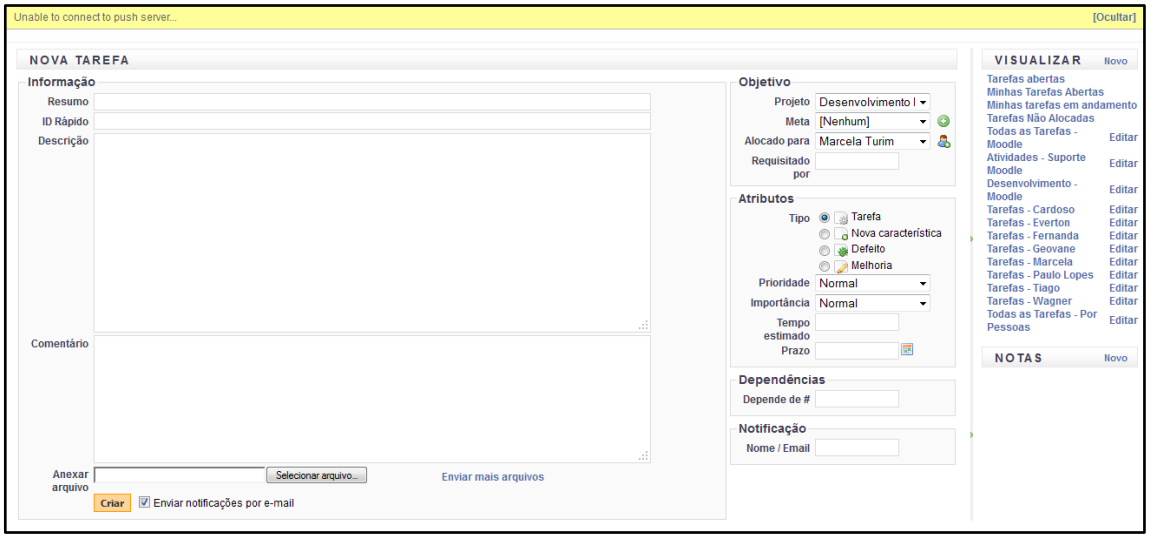

Figura 3 – Nova tarefa no ClockingIt

Para cadastrar uma atividade, o membro da equipe deve informar um título (resumo) para ela e uma descrição. É possível inserir um ID rápido e um comentário também.

O membro da equipe deverá selecionar o projeto para qual a atividade está sendo cadastrada, a meta (validação, implantação, instalação, estudo, desenvolvimento, documentação ou concluído), a alocação (nome do membro designado para resolver a atividade), o tipo (tarefa, defeito, melhoria ou nova característica), a prioridade, a importância, o tempo estimado, o prazo para conclusão e se houver, devem ser informadas as dependências. É possível ainda anexar um arquivo nesta atividade.

Em se tratando de desenvolvimento de *software*, a ferramenta ClockingIt, pode servir para cadastrar e monitorar os requisitos de um sistema, por exemplo.

Uma forma interessante de monitorar os requisitos é extraindo relatórios do ambiente ClockingIt.

A Figura 4 ilustra um desses relatórios, chamado pelo sistema de relatório pivô.

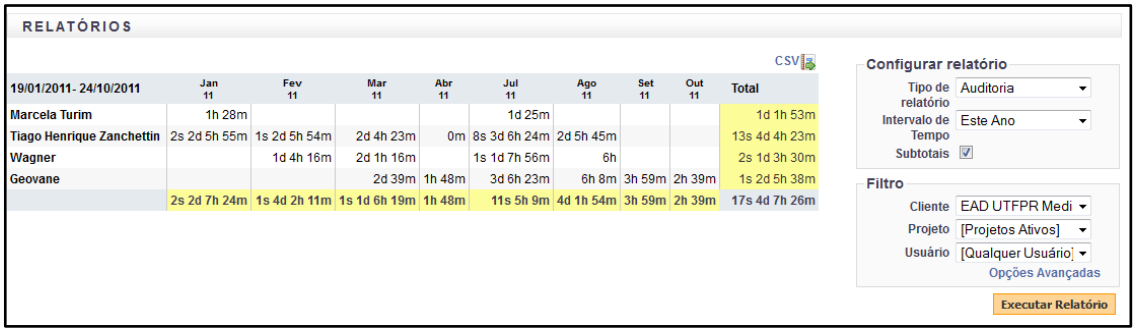

Figura 4 – Relatório Pivô – ClockingIt

Este relatório exibe o total de horas trabalhadas pelos membros da equipe. Com base nesses valores, o gerente de projeto pode argumentar sobre o número de horas trabalhadas nos projetos. No caso de um funcionário ter valores de carga horária incompatíveis com suas atividades, será mais fácil encontra-lo por meio deste relatório do sistema.

O sistema ClockingIt fornece também uma visualização de gráfico de Gantt, onde é possível visualizar de forma gráfica o andamento do projeto. Um exemplo de gráfico de Gantt pode ser *visualizado* na Figura 5.

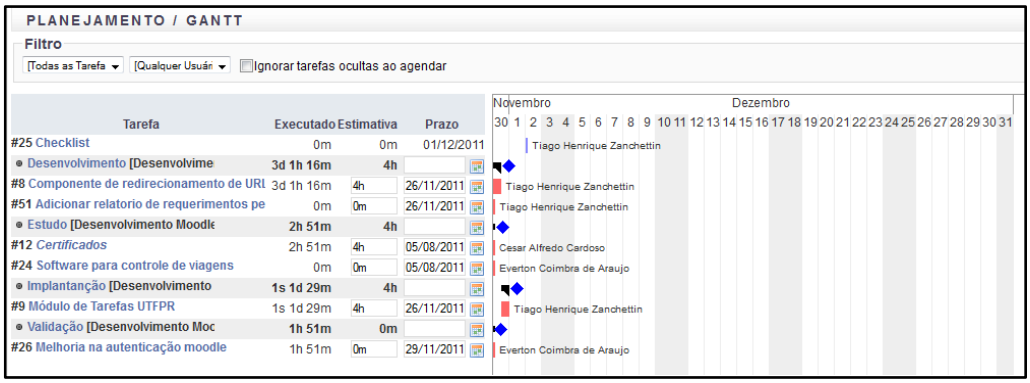

Figura 5 – Gráfico de Gantt – ClockingIt

4.3 FASE DE DESENVOLVIMENTO

Dar início a implantação de um sistema de gerenciamento baseado em Scrum é relativamente simples se for considerado que para iniciar basta o conhecimento da metodologia e a vontade de por em prática esses conhecimentos. Muito mais difícil é manter essa rotina em funcionamento. pensa É errado pensar que apenas a explicação para a equipe de como as coisas devem funcionar seja o suficiente para que a própria equipe se auto gerencie. Ao superestimar uma equipe, o gerente de projetos pode fazer com que o projeto fracasse, pois a equipe ainda

não possui experiência suficiente diante de algumas situações. Além do mais, um gerente disperso pode fazer com que a equipe não sinta segurança ao longo do processo.

Uma equipe com pouco conhecimento de rotinas Scrum precisa ser alertada diariamente para seus papéis e tarefas, isso até que a metodologia torne-se algo natural. É extremamente importante que o gerente ou coordenador dessa equipe fique atento durante a fase de implantação e apresente os caminhos que a equipe deve seguir, porque até então, essa equipe pode ser considerada um tanto quanto imatura para controlar sozinha todas as rotinas Scrum.

O importante é que o gerente fique sempre alerta e disposto a auxiliar a equipe nesse processo, que em sua visão pode ser algo trivial, mas talvez para quem não obteve o mesmo embasamento teórico e prático que ele, essas rotinas podem se tornar algo desinteressante e em suas concepções, até mesmo desnecessárias.

Mesmo levando em consideração as ponderações anteriores, o gerente de projetos não pode achar que sua equipe é incapaz. Ele deve saber a medida de delineamento da equipe, até onde ele pode colaborar e a partir de onde ele deve deixar a equipe ser mais autônoma, pelo menos nos primeiros meses de desenvolvimento da metodologia.

A partir de certo ponto, quando a equipe já estiver madura, o gerente de projetos pode deixar a equipe um pouco mais independente, no sentido de dar mais liberdade para que a equipe tome iniciativas de organização da rotina Scrum. A medida que o tempo irá passando, o Scrum será algo natural, cotidiano da equipe.

A ideia é que, na ausência do gerente de projetos, a equipe consiga se auto organizar e entregar os mesmos resultados (ou mais e melhores) que se o gerente de projetos estivesse presente.

Isso certamente não é algo fácil de conseguir e também não existe uma fórmula exata para isso acontecer. Tudo depende de alguns fatores, mas o principal deles é certamente a persistência. É claro que a persistência não pode virar uma insistência se aquela ideia não deu os resultados esperados. Um gerente de projetos deve sempre saber medir suas atitudes em relação à equipe e a metodologia, afinal, não existe produto ou serviço sem as pessoas que os desenvolvem.

4.4 FASE ATUAL

A primeira tentativa de implantação do Scrum poderia ser vista como um fracasso, no entanto, apesar de não ter correspondido com os resultados esperados, algumas lições importantes foram aprendidas.

Uma equipe não pode ser superestimada nem subestimada. O gerente de projetos deve ficar atento a qualidade de sua equipe e saber conduzi-la. Os membros da equipe não têm as mesmas necessidades e não são estimulados da mesma maneira. Um gerente de projetos deve saber identificar essas necessidades e estar apto para saná-las.

Uma nova tentativa de implantação do Scrum está sendo realizada no setor de Educação a Distância da UTFPR, câmpus Medianeira, no entanto, com uma nova visão.

O gerente de projetos deve ser capaz de fazer com que a equipe acredite no que está sendo desenvolvido, deve saber mostrar a equipe o quanto seu trabalho é importante e o quanto a própria equipe tem a crescer e melhorar. E o mais importante, jamais deve perder a fé em seu próprio trabalho.

5 CONSIDERAÇÕES FINAIS

A pesquisa é certamente algo extremamente importante para qualquer coisa que se deseja implantar ou implementar. Contudo, não é a única coisa a ser levada em consideração. Ela apenas é um elemento dentre tantos outros necessários para que um projeto se concretize.

Quando uma pesquisa ou um estudo é colocado em prática certamente obterá um resultado, mas nem sempre será o resultado esperado ou o resultado adequado. Cabe ao condutor da pesquisa ou estudo saber conduzir suas ideias e seu próprio projeto livrando-se dos obstáculos surgidos ao longo do caminho.

Um projeto nem sempre resultará em sucesso conforme o que se estabelece como sucesso adequado. Muitas vezes erros podem ser até mais proveitosos que os acertos. Claro que o sucesso é sempre requerido, mas quando não é possível obtê-lo, ao menos é possível fazer uma avaliação daquilo que se aprendeu. Se isso não for possível, aí sim, pode-se considerar que houve um fracasso.

O projeto de implantação do Scrum no EaD da UTFPR, câmpus Medianeira, não obteve o resultado esperado até o momento. No entanto, alguns pontos estão sendo revistos.

A assiduidade de participação da equipe é um ponto que será revisto. Algumas práticas serão estimuladas, como por exemplo, as reuniões diárias, a atualização constante do sistema ClockingIt, e a definição dos papéis de acordo com o projeto ou serviço.

5.1 TRABALHOS FUTUROS/CONTINUAÇÃO DO TRABALHO

Está prevista uma nova tentativa de implantação da metodologia Scrum no setor de EaD, da UTFPR, câmpus Medianeira. A equipe e a gerencia de projetos já estão mais amadurecidas e é possível que com trabalho e dedicação, a implantação aconteça com sucesso.

REFERÊNCIAS

ANDERSON, D. J., Agile Management for Software Engineering, Applying the Theory of Constraints for Business Results, Prentice Hall, 2003.

BASS, B., & DUNTEMAN, G.. ***Behavior in groups as a function of self, interaction and task orientation*.** J. Abnorm Soc. Psychology. 1963.

BOEHM, B. and Turner, R., Balancing Agility and Discipline: A Guide for the Perplexed, AddisonWesley, 2003.

GALVES, M.. **ClockingIt**. Acesso em 26 de Novembro de 2011, disponível em http://log4dev.com/: http://log4dev.com/2008/04/12/clockingit/ 12 de Abril de 2008.

KOONTZ E O'DONNEL KOONTZ, H. E O'DONNEL,C; **Os Princípios de Administração: Uma Análise das Funções Administrativas**. São Paulo, Pioneira. 1980.

MARÇAL, Ana Sofia Cysneiros, Estendendo o SCRUM segundo as Áreas de Processo de Gerenciamento de Projetos do CMMI. 2011.

MASLOW, A. (1954). ***Motivation and Personality***. New York: Harper and Row.

PROJECT MANAGEMENT INSTITUTE – PMI, ***A guide to the project management body of knowledge*.** Syba: PMI Publishing Division, 2000. Disponível em: http://www.pmi.org.

SOMMERVILLE, I. (2011). **Engenharia de *Software***. Pearson Education.

TORREÃO, Paula G. B. C, **Gerenciamento de Projetos**, 2006. Disponível em <http://www.fecra.edu.br/admin/arquivos/artigo-gerenciamento-de-projetos-paula-coelho.pdf>. Acessado em Setembro de 2011.