MARASSUEIDE DA SILVA CARVALHO

Desenvolvimento Ágil de Software: O Caso de uma *Softwarehouse* Baiana

VITÓRIA DA CONQUISTA – BA

2011

UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA – UESB

DEPARTAMENTO DE CIÊNCIAS EXATAS – DCE

MARASSUEIDE DA SILVA CARVALHO

Desenvolvimento Ágil de Software: O Caso de Uma *Softwarehouse* Baiana

Trabalho de conclusão de curso para obtenção do título de Bacharel em Ciência da Computação, pela Universidade Estadual do Sudoeste da Bahia – UESB.

Orientador: Prof. Dr. Fábio Moura Pereira

VITÓRIA DA CONQUISTA – BA

2011

MARASSUEIDE DA SILVA CARVALHO

Desenvolvimento Ágil de Software: O Caso de Uma *Softwarehouse* Baiana

Trabalho de conclusão de curso aprovado como requisito parcial para obtenção do título de Bacharel em Ciência da Computação, pela Universidade Estadual do Sudoeste da Bahia – UESB.

Banca Examinadora:

Orientador: Prof. Dr. Fábio Moura Pereira

UESB – Universidade Estadual do Sudoeste da Bahia

Membro: Prof. Esp. Gidevaldo Novais

FTC – Faculdade de Tecnologia e Ciências

Membro: Prof. Esp. Fabricio Sousa Pinto

FTC – Faculdade de Tecnologia e Ciências

Vitória da Conquista, 18 de janeiro de 2011

AGRADECIMENTOS

Gostaria de agradecer a todos que contribuíram direta e indiretamente para que eu pudesse finalizar com êxito mais essa etapa de muitas que virão.

RESUMO

Muitos dos projetos de *software* iniciados não são concluídos dentro dos prazos e custos definidos ou até mesmo são cancelados durante o processo. Atribui-se como uma das causas principais desse problema o pouco ou nenhum gerenciamento. Acredita-se que, por meio de um gerenciamento de projetos efetivo, seja possível direcionar recursos, controlar riscos, manter as mudanças no escopo, dentre outros fatores que podem impactar diretamente no desenvolvimento de um projeto de *software*. Neste trabalho apresentamos um relato de experiência na implantação de um método ágil para gerenciamento de projetos em uma *softwarehouse* de Vitória da Conquista, destacando as dificuldades encontradas, benefícios e experiências adquiridas. Tendo como maior dificuldade encontrada durante o processo a resistência à mudanças por parte dos funcionários envolvidos.

PALAVRAS-CHAVE: Gerência de Projetos. Métodos Ágeis.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 – Introdução

O processo de desenvolvimento de *software* baseia-se em criar um sistema computacional com determinado fim, sendo este planejado e acompanhado durante o tempo necessário até que os resultados esperados sejam obtidos.

Segundo PRESSMAN (2006), um sistema informatizado é composto de dois tipos de componentes: os executáveis em máquina e os não executáveis em máquina. Os componentes de software devem mapear as exigências do cliente em código executável.

Muitos problemas podem surgir no período em que um *software* está sendo desenvolvido. Um erro constantemente observado é um levantamento de requisitos deficiente, bem como a escolha de uma metodologia não aplicável à situação, o que pode acarretar posteriormente em erros graves e até mesmo desencadear o cancelamento do projeto. Um escopo definido de acordo com o desejo do cliente evita que no futuro ocorram atritos e mudanças bruscas no contexto do sistema.

Com o passar do tempo e o desenvolvimento tecnológico, os projetos de *software* se tornaram mais complexos, tanto do ponto de vista operacional, quanto do organizacional, havendo assim, a necessidade de monitorar e planejar as atividades que envolvem todo o processo de desenvolvimento de um *software*.

De acordo com KERZNER (2004), gerenciar projetos em uma época de tantas mudanças é um dos grandes desafios da atualidade. Superar esse desafio é preparar o profissional para gerenciar projetos de forma planejada e competente.

A presença de um gerente de projetos não garante o sucesso, mas reduz os riscos do projeto não estar de acordo com o que foi solicitado pelo cliente. A falha, na maioria das vezes, nos quesitos de mudança de escopo, prazo e custos se dá pela falta de um gerenciamento que centralize as informações e direcione os recursos, além de administrar os riscos e as ações que devem ser tomadas.

Este trabalho apresenta o processo de adoção de um método ágil em uma *softwarehouse* de Vitória da Conquista, tendo como base o *Scrum* e seus princípios. O objetivo principal foi investigar os resultados alcançados por esta *softwarehouse* através do processo de adoção de um método ágil de desenvolvimento de software. Além disso, busca ainda:

- Analisar a importância da gestão de projetos para a melhoria dos produtos desenvolvidos na organização, desde o nível operacional ao gerencial, buscando garantir a satisfação de seus clientes.
- Acompanhar o processo de implantação do processo de gerenciamento de *software* até o nível 2 (dois) do CMMI, bem como do nível G do MPS.BR.
- Analisar se a adoção de um método ágil é adequada a uma pequena empresa.
- Analisar o uso de técnicas de engenharia de software estudadas durante a realização do curso de Ciência da Computação em um caso real.

A falta de gerenciamento ou um gerenciamento insuficiente muitas vezes causa a decadência de projetos de *software*. Inúmeros problemas detectados na indústria de *software* influenciam para que o atraso no processo seja significativo. Além de reduzir o atraso por meio de um controle mais eficiente, a presença de um gerente de projetos também minimiza os custos, podendo chegar a uma redução significativa no valor final do projeto. Este trabalho foi motivado pela inexistência na empresa abordada de nenhum ou pouco gerenciamento no processo de desenvolvimento de *software*. Ao se iniciar um projeto, nenhum cronograma é traçado e os riscos e custos não são planejados. Não há um controle de quando um projeto se inicia e de quando este terminará, bem como o custo final do software desenvolvido.

Acreditava-se que a implantação de um método ágil seria de grande valia para a empresa em questão, não só pelo dinamismo do processo de desenvolvimento, mas também pelo maior controle dos riscos inerentes a cada projeto e também um controle maior das ações dos membros da equipe.

O presente trabalho visou comprovar as afirmações acima.

A metodologia empregada teve como base a aplicação de questionários antes e depois da implantação do processo, tendo como fundamento a

avaliação da situação da empresa pelos funcionários e gestores por meio destes questionários.

Esta monografia está organizada da seguinte maneira: no capítulo 2 é abordado o processo de desenvolvimento de software, destacando o ciclo de vida de software e os principais métodos de desenvolvimento. No capítulo 3 é abordado o processo de gerenciamento de software e levantadas as características dos principais modelos de gerenciamento. O capítulo 4 caracteriza a empresa e situação original. O capítulo 5 refere-se aos resultados alcançados. Finalmente, a conclusão é apresentada no capítulo 6.

2 - O Processo de Desenvolvimento de Software

Neste capítulo, será visto como é realizado o processo de desenvolvimento de *software*, bem como as características do ciclo de vida do *software* e os métodos utilizados para gerir o processo de desenvolvimento.

Nos últimos anos, muitas empresas que desenvolvem *software* vêm se preocupando com a qualidade com a qual seus produtos são desenvolvidos e constantemente investem na melhoria de processos de desenvolvimento. Isso se deve pelo fato das diversas tecnologias adotadas e da ampla concorrência no mercado atual.

Há algum tempo atrás, o *software* era desenvolvido sem controle nem documentação, de forma desestruturada, sem organização e gerenciamento, sendo, na maioria das vezes, entregue fora do prazo.

Por esses motivos, muitas organizações aplicam técnicas, modelos e padrões no decorrer do desenvolvimento de um projeto de software, a fim de que a qualidade atenda às expectativas do cliente.

O método escolhido deve servir de guia para o desenvolvedor, facilitando seu trabalho. O método empregado deve ser escolhido de acordo com o perfil da equipe de desenvolvimento e do resultado que se espera do *software*, com um produto final de qualidade.

2.1- O Ciclo de Vida do *Software*

O ciclo de vida do *software* toma seu segmento a depender do método adotado, podendo ser gerenciado das mais diversas formas. Autores como VARGAS (2005) e KERZNER (2004) destacam a importância de um ciclo de vida bem planejado. Porém, é necessário que haja um método para guiá-lo.

Não há uma regra padrão de um modelo de ciclo de vida que deve ser seguido, porém, cada modelo deve se adequar à realidade na qual o projeto se encontra, ao perfil do cliente, à equipe que irá desenvolver o *software*, aos recursos disponíveis para aquele projeto, dentre outros fatores que podem influenciar direta ou indiretamente no resultado final.

Atualmente, existem normas que certificam o processo de produção de software bem como a avaliação do *software* pronto. Tais normas exigem cada vez mais qualidade no gerenciamento do projeto e tais exigências são transformadas em benefícios para os usuários e desenvolvedores KERZNER (2004). O desenvolvimento de um *software* é caracterizado por fases que quando colocados em seqüência obtêm-se um ciclo de vida do sistema e é este ciclo de vida que deve possuir qualidade.

A norma ISO/IEC 12207, conhecida como *Information Technology - Software Life Cycle,* define padrões para os ciclos de vida:

> O objetivo da ISO/IEC 12207 é estabelecer uma estrutura comum para os processos de ciclo de vida de *software*, com uma terminologia bem definida, que pode ser referenciada pela indústria de *software*. A estrutura contém processos, atividades e tarefas que servem para ser aplicadas durante a aquisição de um sistema que contém *software*, de um produto de *software* independente ou de um serviço de *software*, e durante o fornecimento, desenvolvimento, operação e manutenção de produtos de software. ABNT, 1996.

Segundo VARGAS (2005) “O ciclo de vida pode ser dividido em um conjunto de fases, normalmente fixas para todos os tipos de projeto, contendo uma série de passos principais do processo. Essas fases, por sua vez, são subdivididas em estágios, ou etapas específicas, de cada natureza de projeto, que são, então, subdivididos em atividades”

É interessante que a equipe conheça o ciclo de vida do projeto, pois, a partir dele, pode ser feita a análise para saber se as fases planejadas estão sendo cumpridas, como está o progresso de desenvolvimento e o que deve ser feito para que as fases seguintes não fujam ao escopo esquematizado.

POSSI (2006) descreve o ciclo de vida do *software* como um conjunto de fases sequenciais e que geralmente incluem passos principais englobados pela conceituação, planejamento, projeto, desenvolvimento, implementação e operação dos subprodutos relacionados ao desempenho técnico do projeto.

De acordo com REZENDE (2005), não há *software* pronto e acabado, pois, ao longo de sua permanência exigirá manutenção legal, correções, melhorias e implementações.

VARGAS (2005) subdivide, didaticamente, o ciclo de vida de um projeto em cinco fases: fase de iniciação, fase de planejamento, fase de execução, fase de monitoramento e controle e fase de encerramento. SOMMERVILLE

(2006) subdivide um projeto de desenvolvimento de *software* em três etapas distintas: definição, desenvolvimento e manutenção. De qualquer modo, as fases do ciclo de vida de um projeto dependem da natureza do mesmo, podendo variar de acordo com a magnitude e complexidade com que foi planejado no seu momento inicial.

2.2 - Metodologias para gestão de projetos

Segundo POSSI (2006) “Uma metodologia de planejamento de projetos é uma abordagem estruturada usada para guiar a equipe do projeto durante a elaboração do plano. Ela pode ser simples ou complexa”. SOMMERVILLE (2006) define metodologia como um conjunto de práticas recomendadas para o desenvolvimento de *software*. Essas práticas, geralmente, passam por fases ou passos, que são subdivisões do processo para ordená-lo e melhor gerenciá-lo. De acordo com o PMBOK (2004), a metodologia para gestão de projetos auxilia uma equipe de gerenciamento de projetos no desenvolvimento e controle das mudanças ao longo do processo.

Conforme discutido por KERZNER (2004), boas metodologias de projetos permitem uma boa administração dos clientes e de suas expectativas:

> Não se admite minimizar a importância de uma boa metodologia. Além de melhorar o desempenho durante a execução do projeto, ela criará, igualmente, as condições para aumentar a confiança dos clientes e, assim, aperfeiçoar o relacionamento com eles.

Seguir uma metodologia de gerenciamento de projetos não garante que o mesmo se concluirá com êxito. Porém, se empregada corretamente e aliada às ferramentas de apoio, este direcionamento torna-se crucial para que o projeto tome os rumos de um projeto bem sucedido.

As metodologias têm que se adequar à necessidade do usuário final e também se moldar à medida que o escopo do projeto mude, seja essa mudança por pedido do cliente ou por consequência de fatores externos.

A metodologia, quando escolhida da maneira correta e adequada ao projeto em questão, auxilia o gestor a controlar e direcionar os recursos ao longo do processo por meio de regras, técnicas e ferramentas, integrando-as ao contexto no qual se deseja aplicá-las.

A metodologia se diferencia do método pelo fato de o método ser o objeto de estudo da metodologia. Por meio da metodologia é que se chega ao método. O método é o meio (técnica) que deve ser seguido para que o processo seja realizado de acordo com o que se planeja.

A seguir, apresentaremos os principais métodos para gerenciamento de projetos e suas principais peculiaridades.

2.3 – Métodos Clássicos

Alterações no projeto custam caro. Sendo assim, muitos projetos que necessitam de mudança são abortados, por conta de ser inviável financeiramente tocá-los adiante. Neste contexto são utilizados os métodos clássicos ou tradicionais. Os métodos clássicos possuem como característica marcante o quesito da divisão de etapas distintas, englobando atividades de análise, modelagem, desenvolvimento e testes. Para a finalização de uma fase, há um marco que a caracterize, seja por um diagrama UML, seja por um protótipo de software.

Processos que são guiados dessa forma acabam sendo um limitador para o desenvolvedor. Os desenvolvedores possuem dificuldade em ter que gerar um novo documento a cada fase do ciclo de vida de um software desenvolvido por ele, fazendo com que, muitas vezes, atrase o processo de desenvolvimento pelo fato de ter que gerar os artefatos e a documentação do projeto, características inerentes aos métodos clássicos. Sistemas desenvolvidos dessa forma, geralmente estão no contexto de sistemas estáveis, cujas mudanças necessárias no escopo do projeto são previsíveis e planejadas.

Muitas empresas pequenas que não possuem controle do que produzem, possuem equipes pequenas e não possuem gerenciamento suficiente para gerar os elementos a cada iteração, optam por não utilizar os métodos tradicionais.

A seguir será apresentado o método clássico mais conhecido.

2.3.1 RUP – Rational Unified Process

O maior representante dos métodos clássicos é o RUP (*Rational Unified Process*). Dividido em quatro fases, podendo as mesmas se subdividir em outras menores. FILHO(2000) *apud* JACOBSON (1999) detalha as fases do RUP na tabela abaixo:

Tabela 1: Fases do RUP

Fase	Descrição
Concepção	Fase na qual se justifica a execução de um projeto de desenvolvimento de software, do ponto de vista do negócio do cliente.
Elaboração	Fase na qual o produto é detalhado o suficiente para permitir um planejamento acurado da fase de construção.
Construção	Fase na qual é produzida um versão completamente operacional do produto.
Transição	Fase na qual o produto é colocado à disposição de uma comunidade de usuários.

.

Ao final de cada fase, para que possa haver a análise da situação do projeto, é gerado um marco (*milestone*). Posteriormente, o *milestone* é avaliado e verifica-se a passagem para uma nova fase, o cancelamento do projeto ou se deve haver um replanejamento do mesmo.

A Figura 1 exibe as quatro fases de Concepção (Iniciação), Elaboração, Construção e Transição, bem como as subdivisões em iterações das mesmas, tratadas como Inicial para a fase de Iniciação, E1 e E2 para a fase de Elaboração, C1, C2 e CN (variável) para a fase de construção e T1 e T2

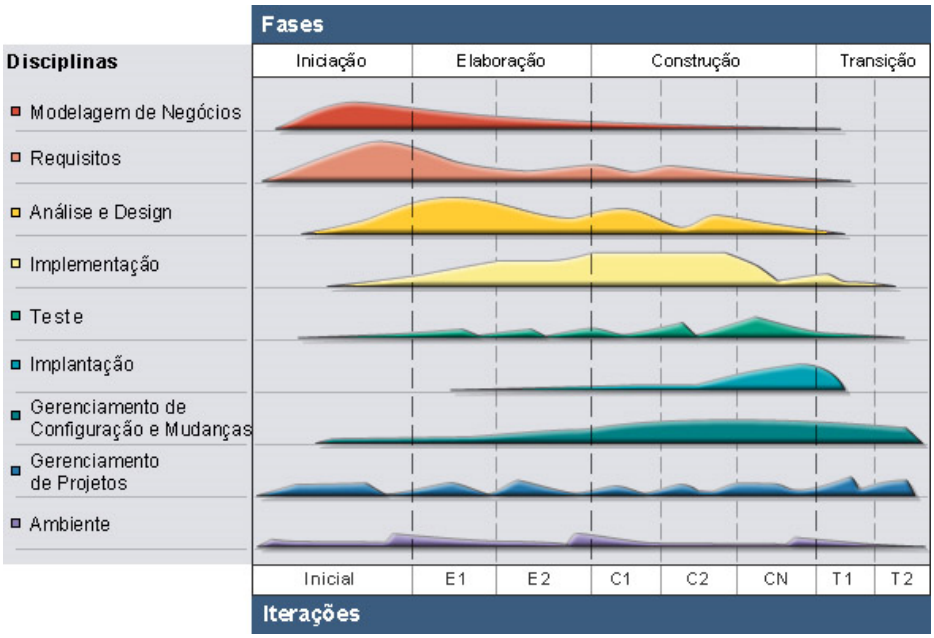

Figura 1: Fases do RUP. Fonte: http://dearchitectura.files.wordpress.com

para a fase de Transição.

A imagem também aborda disciplinas como Modelagem de negócios, Requisitos, Análise e Design, Implementação, Teste, Implantação, Gerenciamento de Configuração e Mudanças, Gerenciamento de Projetos e Ambiente e a situação das mesmas em cada fase.

Outro ponto marcante na arquitetura do RUP é o conceito de *worker*, cujo significado é dado pelo papel desempenhado pelo indivíduo no processo, definindo atividades e responsabilidades deste na equipe.

> O RUP é uma metodologia para gerenciar projetos de desenvolvimento de *software* que usa a UML como ferramenta para a especificação de sistemas. O RUP é composto por um conjunto de disciplinas que fornecem diretrizes para definição das tarefas e para atribuição das responsabilidades. MARTINS (2007).

2.4 – Métodos Ágeis

Os métodos ágeis têm sido vistos como alternativas à utilização dos métodos clássicos. O termo de métodos ágeis surgiu quando uma série de profissionais se reuniu para avaliar o desempenho dos processos de criação de *software*. Tal encontro deu início à Aliança Ágil e ao posterior Manifesto Ágil, cujo pilar se constitui em produzir *software* de qualidade acima de tudo. PRESSMAN (2006).

Nos métodos ágeis prevalece o *software* em funcionamento ao invés de uma documentação extensa. O relacionamento com o cliente é importante, bem como a adaptação do *software* às constantes mudanças. O Manifesto Ágil não deixa de lado quesitos importantes como processos, ferramentas e documentação, mas coloca-os em segundo plano, fazendo com que seja mais importante o software executável, que está pronto, a interação entre os indivíduos envolvidos no seu desenvolvimento e a resposta rápida às mudanças ocorridas durante o processo.

PRIKLADNICKI (2009) define método ágil como um planejamento constante, focando principalmente a codificação, tendo como alternativa para esta finalidade a adaptabilidade, agregação de valores para cada item do processo, orientação aos recursos humanos, comunicação e aprendizado.

Autores como PRESSMAN (2006) E MARTINS (2007) aconselham a utilização de métodos ágeis nas situações nas quais o projeto sofre inúmeras mudanças, os requisitos são alterados constantemente, as equipes são pequenas, as datas para entrega do *software* são curtas, fazendo com que o desenvolvimento rápido seja fundamental.

> A engenharia de *software* ágil combina uma filosofia e um conjunto de diretrizes de desenvolvimento. A filosofia encoraja a satisfação do cliente e a entrega incremental do *software* logo de início; equipes de projeto pequenas, altamente motivadas; métodos informais; produtos de trabalho de engenharia de software mínimos e simplicidade global do desenvolvimento. As diretrizes de desenvolvimento enfatizam a entrega em contraposição à análise e ao projeto (apesar dessas atividades não serem desencorajadas) e a comunicação ativa entre desenvolvedores e clientes. PRESSMAN (2006).

AMBLER (2002) define modelagem Ágil como uma atitude, não como um processo e compreende um conjunto de valores aos quais os modeladores ágeis aderem, resultando em um *software* de melhor qualidade e de desenvolvimento mais rápido, ao mesmo tempo em que evita simplificações excessivas e expectativas não realistas.

Como alternativa à utilização dos métodos clássicos, os métodos ágeis guiam projetos e equipes dinâmicas, tendo entregas para cada fase determinada, atividades que foram planejadas e cumpridas dentro do prazo, equipes em interação, conhecimento das mudanças no andamento do projeto, dentre outros.

> A modelagem Ágil é uma maneira de trabalhar em conjunto de modo eficaz para alcançar os objetivos dos clientes do projeto. Os desenvolvedores ágeis trabalham como uma equipe com os clientes de seu projeto, os quais exercem um papel direto e ativo no desenvolvimento do sistema. É preciso lembrar-se de que, em se tratando de trabalho em equipe, não há “Eu” na modelagem ágil. AMBLER (2002).

PRESSMAN (2006) cita doze princípios da Aliança Ágil para as equipes que desejam aplicá-los no desenvolvimento de seus projetos, dentre os quais destacamos:

- Satisfação do cliente desde o início por meio de entrega contínua de *software* funcional.

- Modificações de requisitos são bem vindas, mesmo que tardias no desenvolvimento.
- Trabalho em conjunto entre os membros da equipe
- *Software* funcionando é a principal medida de progresso.
- Simplicidade: a arte de maximizar a quantidade de trabalho não efetuado é essencial.
- Equipes auto-organizadas.

Existem diversos métodos compatíveis com a Aliança Ágil, dentre os mais populares destacamos o *Extreme Programming* e o *Scrum*.

2.4.1 XP – *Extreme Programming*

De acordo com PRESSMAN (2006) *O Extreme Programming (*XP*)* possui um conjunto bem definido de práticas e valores: a comunicação com o cliente, a simplicidade para a resolução dos problemas, *feedback* aos usuários, coragem para seguir o XP como ele é de fato e realizar um trabalho com a qualidade desejada.

BECK (1999) define XP como uma metodologia ágil para equipes pequenas e médias que desenvolve *softwares* baseados em requisitos vagos e que se modificam rapidamente.

Algumas das características do XP são a prototipagem, a abordagem incremental e a interação dos indivíduos em todas as fases. O XP enfatiza o desenvolvimento rápido do projeto e visa garantir a satisfação do cliente, além de favorecer o cumprimento das estimativas. As regras, práticas e valores do XP proporcionam um agradável ambiente de desenvolvimento de software para os seus seguidores, que são conduzidos por quatro valores: comunicação, simplicidade, *feedback* e coragem.BECK(1999).

Conforme discutido por PRESSMAN (2006), o XP inclui um conjunto de regras e práticas que ocorrem no contexto de quatro atividades de arcabouço: Planejamento, Projeto, Codificação e Teste:

- A fase de planejamento descreve as funcionalidades do *software* que vai ser construído, descritas pelo cliente através das Histórias do Usuário, semelhante aos casos de uso da UML. Uma prioridade, chamada de

Valor, é atribuída à História. Os membros da equipe avaliam a história e atribuem um custo, o qual origina o tempo de desenvolvimento da história. Se houver necessidade de protelar o tempo determinado no custo, a história é dividida em histórias menores. Após a conclusão da primeira fase do projeto, o mesmo é avaliado e esse resultado determina a velocidade do projeto.

- A fase de Projeto no XP segue a linha da simplicidade. É sempre preferível projetos simples a projetos complexos. O projeto é desenvolvido com base no que foi planejado na fase anterior.
- Na fase de codificação, após a definição das Histórias, o XP recomenda que antes da codificação em si, sejam iniciados testes unitários, analisando as Histórias e o que vai ser gerado de código nesta fase. Uma recomendação do XP é que o desenvolvimento seja feito aos pares, pois o código desenvolvido por duas pessoas pode ser analisado e corrigido de forma melhor do que quando somente uma trabalha na codificação de uma História. À medida que as duplas vão terminando seu trabalho, o mesmo é integrado ao das outras duplas.
- A fase de testes fornece dados a respeito dos resultados. Se aquilo que foi planejado de início está sendo executado, o projeto permanece em seu fluxo normal e em caso contrário, o projeto é direcionado para que os erros possam ser corrigidos.

2.4.2 *Scrum*

Segundo MARTINS (2007) o *Scrum* é um método ágil que segue as filosofias iterativa e incremental. Ele se concentra no que é realmente importante: gerenciar o projeto e criar um produto que acrescente valor para o negócio.

O *Scrum* adota uma abordagem empírica, aceitando que o problema pode não ser totalmente entendido ou definido na análise e que provavelmente os requisitos mudarão com o passar do tempo, focando na maximização da habilidade da equipe de responder de forma ágil aos desafios emergentes PRIKLADNICKI (2009).

As equipes do *Scrum* trabalham de forma a dinamizar o processo de comunicação, adaptando as modificações, com incrementos do *software* que são entregues ao final de cada etapa e testes constantes para que os erros sejam detectados inicialmente, impedindo que, posteriormente, seja mais difícil corrigi-los.

De acordo com o Guia do *Scrum* (2009), o *Scrum* é fundamentado na teoria de processos empíricos e é sustentado por três pilares: A transparência, a Inspeção e a Adaptação. A transparência garante que os aspectos dos processos que afetam o resultado final devem ser visíveis aos gerentes. A inspeção possui o objetivo de avaliar a situação do projeto e a adaptação, ajusta o processo à medida que há necessidade.

O *framework* do *Scrum* é composto por times onde cada integrante possui um papel associado, eventos, artefatos e regras.

A Figura 2 ilustra o ciclo de atividades do Scrum, que são divididas em intervalos de tempo, denominados *Sprints*. Todos os requisitos que foram planejados para aquele projeto recebem a denominação de *Product Backlog*, que pode ser alterado ao longo do processo de desenvolvimento. As tarefas do *Product Backlog* que devem ser realizadas no Sprint são apontadas como *Sprint Backlog*.

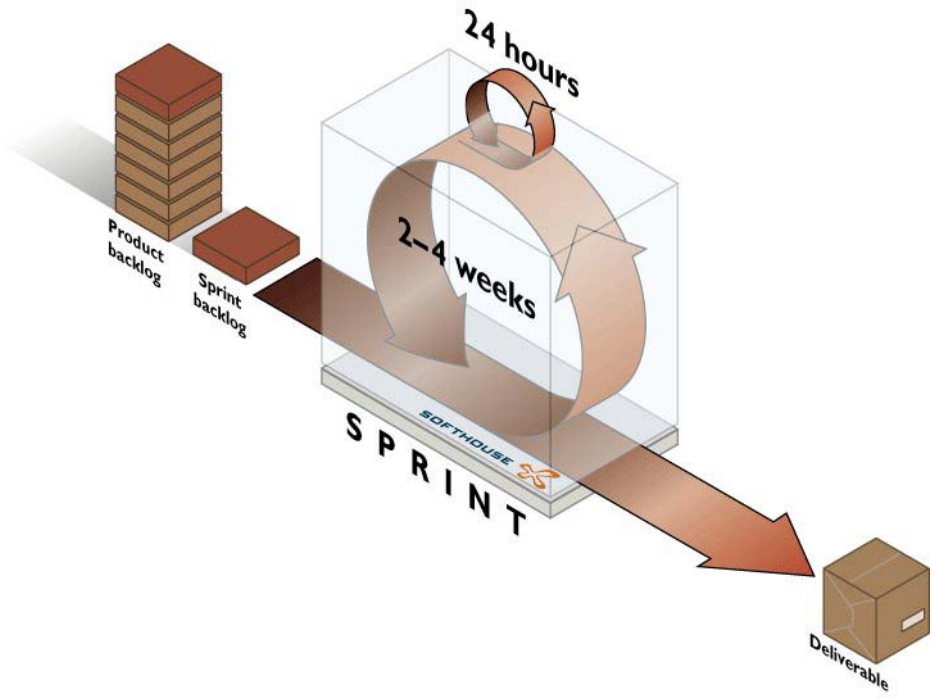

Figura 2: Ciclo do Scrum. Fonte: www.visaoagil.wordpress.com

Os times do *Scrum* são auto-organizáveis, interdisciplinares e trabalham em iterações. Cada time possui três papéis:

- O *Scrum Master*, que é responsável por garantir que o processo seja compreendido e seguido. O *Scrum Master* vai coordenar o time no decorrer dos *Sprints*, fazendo com que o mesmo trabalhe da forma correta para desenvolver seu trabalho.
- O *Product Owner*, que representa o dono do produto (cliente) e é responsável por maximizar o valor do trabalho que o time faz. É o *Product Owner* quem define o *Product Backlog* e as prioridades que dele devem ser executadas, gerando o *Sprint Backlog.*
- O Time, que executa o trabalho propriamente dito. O Time consiste em desenvolvedores com todas as habilidades necessárias para transformar os requisitos do *Product Owner* em uma parte potencialmente "entregável" do produto ao final do *Sprint*.

Dentre os eventos que constituem a harmonia do *Scrum* estão: a reunião de planejamento inicial, a reunião de planejamento do *Sprint*, o *Sprint*, a reunião diária, a revisão do *Sprin*t, a retrospectiva do *Sprint*.

A reunião realizada inicialmente, envolvendo o *Scrum Master* e o *Product Owner,* são elicitados os requisitos de maior relevância e colocados em ordem de prioridade. A partir desta reunião é gerado o *Product Backlog*, constituído pelo conjunto de condições necessárias para que se possa chegar ao produto final conforme planejado.

No início de cada *Sprint* é realizada a reunião de planejamento do mesmo, conhecida como *Sprint Planning Metting*, na qual são retirados do *Product Backlog* o que será realizado neste intervalo de tempo (*Sprint Backlog*) e as tarefas que serão realizadas para que os itens selecionados possam ser efetivados.

As tarefas não podem levar mais de 16 horas para ser executadas ou terem mais de um dia de duração. O próprio time escolhe quais tarefas vai realizar, ou seja, tarefas nunca são atribuídas.

Diariamente, é realizada uma reunião de revisão do Scrum, conduzida pelo *Scrum Master*, geralmente com duração máxima de quinze minutos. A reunião é realizada com todos os membros de pé, para que não só o tempo de

reunião seja mínimo, mas também para que os integrantes interajam de maneira igualitária.

Ao final do Sprint, o time se reúne para avaliar os resultados obtidos, onde são levantados pontos de melhoria para o próximo *Sprint*.

Todo ciclo é reiniciado até que o produto final seja obtido. Para monitoramento do andamento das atividades do projeto, dois gráficos são gerados: o *Product Burndown* e o *Sprint Burndown*, que representam as funcionalidades entregues do produto e do *Sprint* na linha do tempo.

O *Product Burndown* (Figura 3) mostra o trabalho a ser realizado no tempo restante. O trabalho restante é representado pelas coordenadas do eixo Y e o tempo necessário para a realização deste trabalho é representado pelas coordenadas X. A linha representa o esforço demandado para a realização das tarefas. Espera-se que a execução das atividades leve a linha de início em Y ao encontro de X, concluindo a realização das tarefas.

O *Sprint Burndown* (Figura 4) mostra o acompanhamento diário da realização das tarefas existentes no *Sprint.* O *Product Burndown* representa a execução das tarefas e, todos os *Sprints,* que agrega o projeto por inteiro.

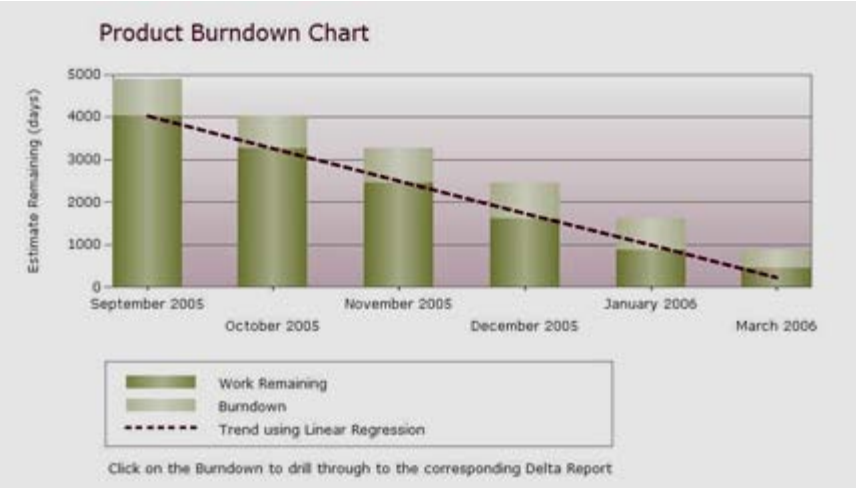

Figura 3: *Product Burndown Chart*

Fonte: ttp://consultingblogs.emc.com

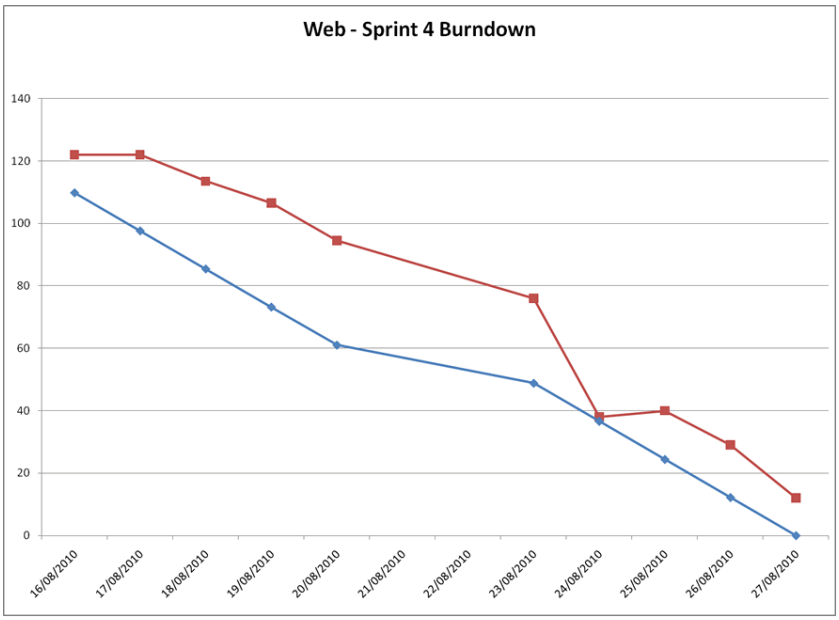

Figura 4 : *Sprint Burndown Chart*

Fonte: http://www.carlbruiners.co.uk/

A seguir será abordado o processo de gerenciamento de *software*, englobando a qualidade do *software*, os principais modelos de processo de gerenciamento de *software* e suas características.

3 - O Processo de Gerenciamento de *Software*

Neste capítulo, será discutido o processo de Gerência de *software* e serão apresentados os modelos de processos de gerenciamento de *software*, suas características, vantagens e desvantagens.

Com o surgimento de inúmeros erros no desenvolvimento dos projetos de *software*, tais como atraso na entrega ou custo excedente, houve a necessidade de se produzir um plano de gerenciamento com a finalidade de solucionar estes e outros problemas relacionados à falta de um sistema de gestão.

Um estudo feito pelo Departamento de Defesa dos Estados Unidos (1994) mostrou que 75% de todos os grandes sistemas de *software* falham e que a causa principal é o pobre gerenciamento por parte do desenvolvedor ou adquirente. O mesmo estudo mostra que o tempo dos projetos de *software* excede em 50% do tempo planejado no cronograma de todo o projeto.

A seguir, será discutida brevemente a importância da qualidade do *software* no processo de desenvolvimento do mesmo.

3.1 – Qualidade do Software

O processo de planejamento da busca pela qualidade do projeto ocorre paralelamente às suas atividades, tendo em vista que a qualidade no ponto de vista do cliente é indispensável. MARTINS (2007) define qualidade como a avaliação entre o que foi executado e quais foram os objetivos do projeto, planejados primeiramente, além das informações necessárias para que o cliente possa utilizar o produto, entrega pontual e dentro do custo avaliado, bem como a documentação de todos os processos envolvidos.

> A qualidade de um produto se aplica tanto aos elementos técnicos e funcionais quanto à documentação. A solução deve agradar ao cliente, ser de fácil administração, robusta, confiável e estável. A documentação deve ser completa, clara, de fácil consulta e seguir os padrões impostos pelo cliente. No âmbito do projeto, a qualidade está associada ao cumprimento das instruções de trabalho que regem o projeto: comunicação, cumprimento das metas, prazos e

> orçamento de cada pacote de trabalho, gerenciamento dos riscos e tudo mais, conforme fora planejado. (MARTINS, 2007).

Os testes são muito importantes para a garantia de que o produto está conforme acordado com o cliente. Por meio destes, é possível avaliar o nível de confiabilidade do *software* e buscar corrigir os problemas que são descobertos nessa fase.

Há inúmeras formas de avaliar a qualidade de um *software*, tanto por fatores externos, quanto por fatores internos. A avaliação feita por fatores externos se dá pelo julgamento dos usuários, como velocidade, facilidade de uso e funcionalidade. Os fatores internos são avaliados pela equipe de desenvolvimento, como *bugs*, robustez e integridade dos dados.

A norma internacional ISO/IEC, publicada em 1991, define qualidade de software como a “totalidade de características de um produto de *software* que lhe confere a capacidade de satisfazer as necessidades explícitas e implícitas”. A mesma norma fornece um modelo geral, o qual define seis categorias utilizadas para analisar a qualidade de um produto de software: funcionalidade, confiabilidade, usabilidade, eficiência, manutenibilidade e portabilidade.

Para garantir a qualidade do produto final e obter êxito com relação à avaliação do cliente, o processo de desenvolvimento do software precisa ser bem gerenciado, tendo cada fase acompanhada e monitorada por um gestor responsável pelo projeto.

3.2 – Gerência de Projetos

KERZNER (2004) define projeto como uma série de atividades e tarefas que tenham um objetivo específico a serem completados dentro de certos requisitos, prazos e escopos bem definidos.

O *Project Management Body Of Knowledge* (PMBOK) foi desenvolvido pelo *Project Management Institute* (PMI), uma entidade internacional que congrega os profissionais de áreas relacionadas à Gerência de Projetos (*Project Managment*). Essa entidade possui o objetivo padronizar as atividades referentes à gestão das atividades.

O PMBOK (2004) define Gerência de Projetos:

> O gerenciamento de Projetos é a aplicação de habilidades e técnicas às atividades do projeto a fim de atender aos seus requisitos. O gerenciamento de projetos é realizado através da aplicação e da integração dos seguintes processos de gerenciamento: iniciação, planejamento, monitoramento, controle e encerramento. PMBOK (2004).

PRESSMAN (2006) define a Gestão de Projetos como um conjunto de atividades sincronizadas. Tais atividades são distribuídas a partir do levantamento das estimativas (tempo, custo, esforço), objetivos do projeto, escopo do *software*, levantamento dos recursos, etc.

De acordo com SOMMERVILLE (2006), muitos projetos de *software* fracassaram nas décadas de 60 e 70 por conta do pobre gerenciamento. Não há uma definição específica para as atividades as quais o gerente de *software* deve se concentrar, porém a maioria deles segue as fases de definição e elaboração das propostas, planejamento, custo, seleção de pessoal e elaboração dos relatórios inerentes ao monitoramento do processo de desenvolvimento.

Conforme foi discutido KERZNER (2006), a definição de gestão de projetos se concentra no planejamento, na programação e no controle de uma série de tarefas integradas com o objetivo de atingir suas metas com êxito, em benefício dos participantes do projeto. Para uma gestão de projetos bem sucedida são necessários planejamento e coordenação, havendo o monitoramento de cada fase do processo de gerenciamento.

Um estudo levantado no ano de 2002 pelo *Standish Group International* revela, através de dados alarmantes, estatísticas que explanam o quanto é necessária uma boa gerência de projetos por meio de profissionais qualificados: 31% de todos os projetos são cancelados antes do término; 88% dos projetos ultrapassam seu prazo, custo ou ambos; Os projetos ultrapassam, em média, 189% dos custos estimados e 222% do prazo estimado.

A Tabela 2 reúne os dados obtidos em uma pesquisa realizada pelo PMI-Rio no ano de 2003, que resume os motivos pelos quais muitos projetos fracassam constantemente.

Tabela 2: Motivos pelos quais os projetos fracassam. Fonte: PMI - Rio (2003)

Motivos pelos quais os projetos fracassam	
Falta de autoridade dos Gerentes de Projetos	**19%**
Falta de apoio da alta gerência	**22%**
Falta de detalhamento na Estrutura Analítica do Trabalho (EAT)	**51%**
Não definição dos produtos finais do projeto	**50%**
Falta de entendimento do objetivo do projeto	**52%**
Falta de formalização do escopo do projeto	**55%**
Expectativas do cliente (interno/externo) não alinhadas à expectativa do projeto	**86%**

Abaixo, serão mostrados alguns dos modelos de processo de gerenciamento de projetos e seus pontos relevantes.

3.3 – Modelos de Processo de Gerenciamento de Software

Existem diversos modelos para gerenciamento de projetos com o objetivo de traçar de maneira simples e direta de como se deve chegar a um nível de gerenciamento desejável.

São ressaltados modelos de processo de software amplamente utilizados, cujo objetivo é moldar fases dos níveis de maturidade, a fim de que o gerenciamento possa ser gradativo, seguindo os níveis planejados.

Dentre os modelos, destacamos o PMBOK, o CMM e o MPS.BR.

3.3.1 – PMBOK

O PMBOK reúne as boas práticas de gerenciamento de projetos, com o objetivo de fundamentar o processo de desenvolvimento de *software*, tendo por base princípios de controle e aplicação da gerência de projetos nas organizações.

> O principal objetivo do *Guia PMBOK* é identificar o subconjunto do Conjunto de conhecimentos em gerenciamento de projetos que é amplamente reconhecido como boa prática. “Identificar” significa fornecer uma visão geral, e não uma descrição completa. “Amplamente reconhecido” significa que o conhecimento e as práticas descritas são aplicáveis à maioria dos projetos na maior parte do tempo, e que existe um consenso geral em relação ao seu valor e sua utilidade. “Boa prática” significa que existe acordo geral de que a aplicação correta dessas habilidades, ferramentas e técnicas podem aumentar as chances de sucesso em uma ampla série de projetos diferentes. PMBOK (2004).

De acordo com o PMBOK (2004), como é mostrado na Figura 5, a gestão de projetos é padronizada pela divisão das seguintes sub-áreas:

- Gerenciamento de Integração do Projeto: Inclui os processos e atividades necessários para identificar, definir, combinar, unificar e coordenar os diversos processos e atividades de gerenciamento de projetos dentro dos grupos de processo. A integração inclui características de unificação, consolidação, articulação e ações integradoras que são essenciais para o término do projeto.
- Gerenciamento de Escopo: Engloba os processos necessários para garantir que o projeto inclua todo o trabalho necessário para que o projeto seja finalizado com sucesso. O gerenciamento de escopo do projeto trata principalmente da definição e controle do que está e do que não está incluído no projeto.
- Gerenciamento de Tempo: Inclui os processos necessários para que o projeto seja finalizado dentro do prazo.
- Gerenciamento de Custos: Abrange os processos envolvidos com estimativas, planejamentos, a fim de que seja possível terminar o projeto dentro do orçamento aprovado.
- Gerenciamento de Qualidade: Engloba todas as atividades, responsabilidades, objetivos e políticas de qualidade, de modo que o projeto atenda às necessidades que motivaram sua realização.
- Gerenciamento de Recursos Humanos: Inclui todos os processos que gerenciam a equipe do projeto. As pessoas envolvidas no projeto possuem papéis e responsabilidades atribuídas ao longo da execução do projeto.

- Gerenciamento de Comunicação: Emprega os processos necessários para garantir a geração, coleta, distribuição, armazenamento, recuperação e destinação final das informações sobre o projeto de forma eficiente e adequada.
- Gerenciamento de Riscos: Agrega os processos envolvidos com a identificação, análise, resposta, monitoramento e controle do gerenciamento dos riscos em um projeto. O objetivo do mesmo é aumentar a probabilidade de eventos favoráveis ao projeto e reduzir o impacto dos eventos negativos ao projeto.
- Gerenciamento de Aquisições: Inclui os processos de compra ou aquisição de produtos, serviços ou resultados necessários para a realização do trabalho.

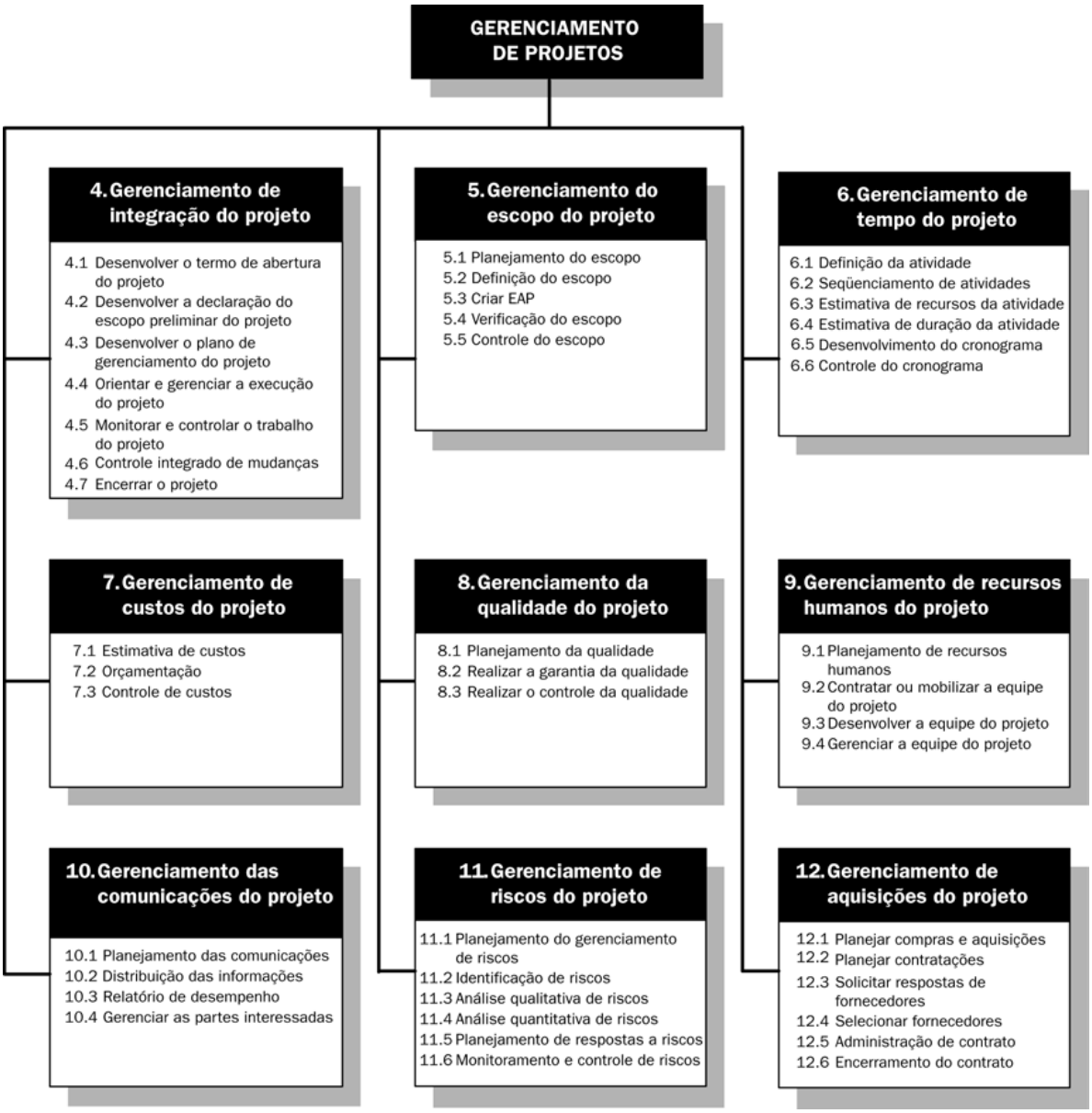

Figura 5: Visão geral das áreas de conhecimento em gerenciamento de projetos e os processos de gerenciamento de projetos. Fonte: PMBOK (2004).

O conhecimento de gerenciamento de projetos, discutido no PMBOK, consiste na definição do ciclo de vida do projeto, nas nove áreas descritas acima e nos cinco grupos de processos de gerenciamento de projetos, apresentados a seguir:

- Grupo de processos de iniciação. Define e autoriza o projeto ou uma fase do projeto.

- Grupo de processos de planejamento. Define e refina os objetivos e planeja a ação necessária para alcançar os objetivos e o escopo para os quais o projeto foi realizado.
- Grupo de processos de execução. Integra pessoas e outros recursos para realizar o plano de gerenciamento do projeto para o projeto.
- Grupo de processos de monitoramento e controle. Mede e monitora regularmente o progresso para identificar variações em relação ao plano de gerenciamento do projeto, de forma que possam ser tomadas ações corretivas quando necessário para atender aos objetivos do projeto.
- Grupo de processos de encerramento. Formaliza a aceitação do produto, serviço ou resultado e conduz o projeto ou uma fase do projeto a um final ordenado.

3.3.2 – Capability Maturity Model (CMM)

O CCM, modelo de maturidade de capacidade, desenvolvido pelo *Software Engineering Institute* (SEI), descreve os elementos mais importantes para a melhoria e avaliação do processo de software, bem como da qualidade do mesmo, possuindo uma sequência de passos para que um projeto atinja o nível de aceitação que converge para um projeto otimizado.

O CMMI (*Capability Maturity Model Integration*), descreve princípios de maturidade do software, buscando auxiliar as organizações a melhorar o seus processos, passando de um nível de desenvolvimento totalmente desorganizado (não gerenciado) ao nível de processos maduros e disciplinados.

Conforme discutido por FERNANDES (2009) o principal propósito do CMMI é fornecer diretrizes baseadas em melhores práticas para melhoria dos processos e habilidades organizacionais, cobrindo o ciclo de vida de produtos e serviços completos, nas fases de concepção, desenvolvimento, aquisição, entrega e manutenção.

Há dois modelos de níveis de maturidade no CMMI. O primeiro deles se inicia com o nível zero ou *ad-hoc* (Incompleto), que representa os projetos sem nenhuma área de processo. O restante se iguala ao modelo descrito a seguir.

São cinco os níveis de maturidade do CMMI:

- Inicial: Neste nível, os processos são informais, caóticos e excedem o orçamento e o cronograma. A realização dos processos se dá pelo esforço individual em executar as tarefas.
- Gerenciado: Nesta fase, os processos, requisitos e serviços são gerenciados, passando da desorganização do nível 1 para o controle e monitoramento neste nível.
- Definido: O conjunto de processos padrão da organização é estabelecido e melhorado ao longo das fases.
- Gerenciado Quantitativamente: Os objetivos quantitativos, baseados nas necessidades dos clientes e usuários finais, são estabelecidos e utilizados como critério para gerenciamento de projetos.
- Otimizado: Há o contínuo melhoramento dos processos por meio dos avanços tecnológicos incrementais e inovadores.

3.3.3 – Melhoria de Processo do Software Brasileiro - MPS.BR

O MPS.BR foi criado em 2003, com o objetivo de melhorar a produção de software no Brasil, adequando-se às empresas dos mais variados portes, possuindo um Guia de Implementação(2006) cuja função é descrever os passos para a implementação dos níveis de maturidade nas organizações.

Os níveis são:

- Nível G – Parcialmente Gerenciado: é o primeiro nível de maturidade do MPS.BR. Ao final da implantação do mesmo, a empresa deve ser capaz de gerenciar seus projetos de desenvolvimento de software. O escopo e o ciclo de vida já devem estar definidos, cronograma e orçamento estabelecidos, o planejamento do projeto é realizado com todos os envolvidos com o mesmo.

- Nível F – Gerenciado: são acrescidos aos processos do nível G os processos de Gerência de Aquisição, configuração, qualidade e medição.
- Nível E – Parcialmente Definido: Agrega os processos dos níveis de maturidade F e G, acrescentando os processos de Adaptação do Processo para Gerência do Projeto, Avaliação e Melhoria do Processo Organizacional, Definição do Processo Organizacional e treinamento.
- Nível D – Largamente Definido: Este nível é composto dos níveis anteriores (G ao E), agregados aos processos de anteriores (G ao E) e acrescido dos processos de Desenvolvimento de Requisitos, Integração do Produto, Solução Técnica, Validação e Verificação.
- Nível C – Definido: O nível de maturidade C é combinado pelos processos dos níveis - de maturidade anteriores (G ao D), acrescidos dos processos de Análise de Decisão e Resolução e Gerência de Riscos.
- Nível B – Gerenciado Quantitativamente: Este nível é composto por todos os níveis de maturidade anteriores, agregados aos processos de Desempenho do Processo Organizacional e Gerência Quantitativa do Projeto.
- Nível A – Em Otimização: Este nível de maturidade é composto pelos processos dos níveis de maturidade anteriores (G ao B), acrescidos dos processos Implantação de Inovações na Organização e Análise e Resolução de Causas.

Os processos apresentados buscam padronizar o gerenciamento de projetos, subdividindo-o em etapas, a fim de que seja cumprido o que foi planejado inicialmente, garantindo a constância do produto final.

O gerenciamento de projetos mostra-se fundamental na indústria do software. É possível concluir um projeto com êxito, tendo prazos cumpridos, custos equilibrados, estabilidade do escopo, riscos controlados, recursos humanos integrados por meio de um gerenciamento eficiente.

4 – Estudo de caso: Conquista Tecnologia

4.1 – A Empresa

A Conquista Tecnologia (CTEC), localizada à Rua Franklin Ferraz, 890, Bairro Candeias, cidade de Vitória da Conquista, Bahia, é uma empresa especializada em sistemas de saúde pública. A CTEC possui, atualmente, seis funcionários registrados e dois prestadores de serviço, desenvolvendo atividades de suporte e desenvolvimento.

A missão da empresa é “Dotar as administrações de políticas públicas positivas e fazer com que a tecnologia seja um meio eficiente de atendimento e compromisso permanente com a comunidade”.

Com relação ao histórico da CTEC, retirado do diagnóstico realizado em 2009, no que diz respeito a atividades inovadoras, pode-se destacar: (1) A CTEC foi a primeira empresa do país a produzir um software na área de regulação em saúde para a plataforma Web; (2) A CTEC foi a primeira empresa do Nordeste a desenvolver solução para o Ministério da Saúde (sistema GIROS), com a finalidade de atender às necessidades da Secretaria de Saúde do Município de Aracaju/SE e que posteriormente foi implementada em outros municípios dos estados do Paraná e Ceará; (3) A CTEC foi a primeira empresa do interior da Bahia a utilizar uma metodologia ágil de desenvolvimento de software (Scrum); e (4) A empresa está desenvolvendo produtos inovadores nas áreas de Segurança Pública e Educação com o intuito de diversificar os seus produtos.

O principal produto da CTEC é o sistema Central de Saúde que visa contribuir para a reorganização da rede SUS, segundo um grau de complexidade dos serviços estabelecendo métodos, critérios e parâmetros para facilitar o acesso e a melhoria da qualidade da assistência. O sistema permite a regulação da assistência, cujo principal objetivo é promover a igualdade do acesso, permitindo ajustar a oferta assistencial disponível às necessidades imediatas do cidadão, de forma imparcial e qualificada.

A empresa emprega as seguintes tecnologias no seu processo de desenvolvimento de software: para linguagem de desenvolvimento Java e JSP; XML para definição e Intercâmbio de Dados; DHTML e JavaScript para

apresentação e controle de eventos; JSTL para encapsulamento de código; AJAX para transmissão assíncrona de dados na Web; PostgreSQL/FireBird/MySQL como sistema gerenciador de banco de dados; Windows ou Linux, sistema operacional; Apache + Tomcat como servidores Web + Web dinâmico.

4.2 – Diagnóstico

Apresento, a seguir, o diagnóstico apresentado na empresa em outubro de 2009, antes da implantação do processo de gerenciamento de projetos.

Alguns pontos foram destacados da Proposta de Implantação do Processo de Gerenciamento de Software :

- O gerenciamento de projetos de software está relacionado às atividades envolvidas em assegurar que o software será entregue dentro do prazo definido no cronograma e de acordo com os requisitos levantados. Apesar de estar há muito tempo no mercado e possuir um software consolidado, a empresa não possui processo de gerenciamento formal das atividades que envolvem projetos de software, a exemplo de: gerenciamento de tempo, custo, comunicação, riscos e de qualidade.
- A empresa não tem controle, por exemplo, do tempo necessário para realização de uma determinada atividade, o que impossibilita também o controle de custos. Como a empresa comercializa apenas um produto de software, todo o custo é alocado de maneira geral ao desenvolvimento e suporte deste produto, mas não se conhece o custo/tempo alocado a cada módulo individual que compõe o produto.
- Os prazos são mantidos de maneira informal, por meio da experiência dos desenvolvedores e, na maioria das vezes, não são seguidos. Não existe o controle de riscos ou de qualidade, levando, por muitas vezes, à descoberta de problemas após a implantação do sistema/módulo.
- Não existe um método formal de desenvolvimento de software. O desenvolvimento é realizado de forma empírica. Os desenvolvedores baseiam-se na experiência adquirida ao longo da execução de trabalhos anteriores.

- No que diz respeito ao processo de desenvolvimento de software, o processo de implementação é baseado no único documento gerado durante as fases de análise e projeto: o modelo ER do banco de dados e nas anotações do desenvolvedor. O código desenvolvido é comentado de maneira desorganizada.
- Após o estabelecimento de contrato de entrega/instalação do software e de prestação de serviços de manutenção entre a empresa e o cliente, a entrega é marcada pela assinatura do cliente confirmando o recebimento do produto.
- O processo de instalação é acompanhado pelo funcionário da empresa nas instalações do cliente. O treinamento é realizado nas instalações do cliente por funcionário da empresa, geralmente do setor de suporte.
- O suporte ao cliente é realizado através do programa *Microsoft Messenger*, de maneira informal e conversas não são registradas. Os funcionários do setor de suporte não passam por processo periódico de avaliação. O desempenho é medido apenas através do atendimento aos chamados abertos no sistema de gerenciamento de bugs MANTIS.
- Chamadas do MANTIS que relatam erros no sistema são repassadas ao coordenador de suporte que realiza avaliação e caso necessário, encaminha ao responsável pela manutenção do sistema.
- Problemas de comunicação foram detectados por falta de informação aos funcionários do suporte sobre modificações em funcionalidades do sistema. Muitas vezes os funcionários são avisados pelos próprios clientes. Outra dificuldade do setor de suporte é a existência de diferentes versões nos diversos clientes.

A proposta de implantação de um método ágil de gerenciamento de *software* teve como objetivo garantir a melhoria da qualidade dos produtos da empresa, bem como dos seus processos de produção.

O método ágil escolhido para a implantação foi o *Scrum,* por sua simplicidade, objetividade e facilidade de adequação a times pequenos.

A seguir, apresento uma breve descrição das atividades programadas para serem desenvolvidas durante o processo de implantação:

- Redefinição das atividades de manutenção de software: as atividades de desenvolvimento de novos produtos e manutenção dos produtos existentes devem ser atribuídas a funcionários distintos. Não é possível a manutenção de um cronograma de desenvolvimento consistente caso esta atividade seja acumulada por uma mesma pessoa. A proposta é de planejamento, treinamento e implantação das novas atividades, além de realização de uma avaliação das interfaces mais utilizadas do software atual (parte do processo de evolução do software).
- Redefinição das atividades do setor de suporte: tarefas de planejamento, treinamento e implantação de métodos de trabalho do setor de suporte.
- Gerenciamento de configuração: tarefas de planejamento, treinamento e implantação da gerência de configuração no setor de suporte. A gerência de configuração é a disciplina responsável por controlar a evolução de sistemas de software, não se propõe a definir quando e como devem ser executadas as modificações nos produtos, papel reservado ao próprio processo de desenvolvimento, a sua atuação ocorre como processo auxiliar de controle e acompanhamento.
- Política de segurança: notou-se durante a realização do diagnóstico que a empresa não possui preocupação especial com a segurança, principalmente no que diz respeito ao acesso aos dados, tanto em nível interno, como em nível externo. Nesta etapa serão planejadas e implantadas políticas de segurança tanto em nível interno, como em nível externo.
- Gerenciamento de projeto/ciclo de vida: atividades de planejamento, treinamento e implantação de método de trabalho para gerenciamento de projetos e para acompanhamento do ciclo de vida do software.
- Método de desenvolvimento de software: atividades de planejamento, treinamento e implantação de método para desenvolvimento de software. Envolve as atividades de escolha do método mais adequado a ser adaptado à realidade da empresa, seleção dos documentos a serem produzidos e atividades a serem realizadas durante o processo de desenvolvimento.

Como pôde ser visto, o processo de mudança deve ocorrer de forma gradual e contínua. Cada etapa implantada deve ser devidamente avaliada e o processo deve contar com a participação de todos os envolvidos. A ordem estabelecida não é rígida e pode ser modificada de acordo com as necessidades da empresa.

4.3 – Implantação do método de Gerenciamento

A empresa utiliza duas ferramentas para auxiliar no processo de gerenciamento de software: o *Mantis Bug Tracking* e o *Agilo*. Antes da implantação do processo de gerenciamento de software a empresa já utilizava o *Mantis*. O *Mantis* é uma ferramenta gratuita, de código aberto, desenvolvida em PHP, trabalha com banco de dados MySQL, PostgreSQL e servidor *Web*.

Os casos reportados são separados em Projetos ou Sub-Projetos, possuindo uma hierarquia desde os casos menos urgentes aos casos que precisam ser resolvidos com maior urgência.

São lançados casos na ferramenta desde especificidades da manutenção do software, bem como *bugs* decorrentes do seu funcionamento e recursos solicitados pelos clientes.

Cada caso novo é testado e então atribuído a um membro da equipe de desenvolvimento. Cada membro da equipe possui um papel, configurado na ferramenta, com acessos restringidos de acordo com a atividade que desenvolve. O relator cria as solicitações, altera e monitora os casos. O desenvolvedor executa e também cria as solicitações, adicionando informações sobre as solicitações e é responsável pelo monitoramento das mesmas. O gestor distribui as solicitações, monitora e gera os relatórios de produtividade. O administrador configura os usuários, os projetos, dentre outras especificações da ferramenta.

Cada solicitação lançada no *Mantis* possui um *status*. Abaixo, cada *status* dos casos abertos por meio da ferramenta e sua especificação:

- Novo: Novo caso registrado por um relator
- Retorno: Um caso fechado indevidamente ou atribuído da mesma forma pode ser alterado e ao mesmo podem ser adicionadas informações.

- Admitido: O gestor passou a ter conhecimento sobre o caso.
- Confirmado: O desenvolvedor está ciente do caso e trabalhando no mesmo.
- Atribuído: O caso foi encaminhado pelo gestor ao desenvolvedor.
- Resolvido: O caso foi executado pelo desenvolvedor.
- Fechado: O relator que registrou o caso confirma o atendimento e valida a implementação.

Os clientes possuem acesso ao *Mantis*, instalado no Servidor da empresa, para que possam relatar os casos e os mesmos são testados pela equipe de suporte, sendo, posteriormente, encaminhados à equipe de desenvolvimento.

O *Mantis* possui a opção de gerenciar os usuários, restringir os acessos e controle de projetos.

Na opção Resumo, os casos são visualizados de forma sintética e o desempenho e a produção dos desenvolvedores e relatores também podem ser avaliados.

Relatórios podem ser impressos por meio do *Mantis* para que o gestor possa avaliar visualmente, facilitando o processo de tomada de decisões.

Projetos são criados e os usuários podem ser vinculados a estes projetos. Um usuário pode ser relator em um projeto e desenvolvedor em outro, dentre outras formas de aproveitamento da ferramenta.

Para a implantação do método *Scrum* na empresa foi necessária a utilização de uma ferramenta para dar suporte ao processo. A ferramenta escolhida para a execução dos princípios do *Scrum* na empresa foi o *Agilo*, ferramenta de código aberto e desenvolvida em *Phyton*, escolhida por sua simplicidade de instalação e operação.

Os casos relatados no *Mantis* são exportados no formato CSV para que o arquivo gere os requisitos e *user stories*, que farão parte do *Product Backlog* do *Scrum*.

Artefatos como requisitos, *user stories*, tarefas e *bugs* podem ser gerados a partir da utilização do *Agilo*. Tais componentes, caracterizados como Tickets, possuem um relator, neste caso pode ser o *Scrum Master* ou o *Product Owner*.

Após a reunião de planejamento inicial, são retirados do *Product Backlog* os casos de maior urgência, montando, assim, a lista de requisitos denominada *Sprint Backlog*, que pode ser visualizada na ferramenta.

A ferramenta possui uma *timeline*, na qual podem ser visualizados os tickets criados e alterações realizadas ao longo dos *Sprints*. Os *Sprints* são criados pelo *Scrum Maste*r e acompanhados pelo mesmo. Ao término do *Sprint* é gerado o *Sprint Burndown Chart*, no qual podem ser visualizados os dados sumarizados do *Sprint*.

O administrador possui o domínio de criar usuários e gerenciá-los, dando-lhes permissões.

Cada funcionário que utiliza a ferramenta possui a visualização de seus *Tickets*, trabalhando para o cumprimento de suas atividades e realização das tarefas diárias. A cada *user story* pode ser associada uma ou mais tarefas. Cada usuário inclui a carga horária provável para a realização de cada tarefa, determinando o tempo dispensado a cada cumprimento e encerramento de um *user story*.

A cada reunião diária, os membros da equipe relatam as atividades que desenvolveram no dia anterior com relação ao projeto, o que pretende ser feito e o que falta ser executado. Cabe ao *Scrum Master* acompanhar através do *Agilo* a realização destas tarefas, podendo avaliar de fato o cumprimento das atividades.

O *Agilo* possui configuração de acesso para o *Product Owner* e o *Scrum Master*, possibilitando aos mesmos, avaliar se os membros da equipe estão trabalhando para o cumprimento das suas atividades e se o tempo previsto para finalizar uma tarefa está de acordo com o que estão trabalhando de fato.

Sprint Backlog for Dezembro — Cannot modify sprints that have ended.

ID	Summary	Remaining Time	Owner
47	0001711: Alterar a maneira de dar conformidade no módulo controle e avaliação	5	Hugo
57	Analisando.	5	Hugo
48	0001717: Data de expiração de senha	5	Ricardo
54	Aguardando dados.	5	Ricardo
49	0001709: Exames de laboratório		Ricardo
50	0001699: Distribuição por unidade	3	Hugo
53	Criando janela que permita gerar todas as agendas de uma vez	3	Hugo
51	0001698: TFD: Comprovante de viagem		Hugo
52	0001697: TFD: relatório de acompanhamento	3	Hugo
60	Trabalhando	3	Hugo
55	001715: Alterar a quantidade do preparo		Hugo
56	0001722: Problema no modulo regulador.		Hugo
58	0001725: Extrato do Prestador com valores diferentes	3	Ricardo
59	Trabalhando	3	Ricardo

Figura 6: *Sprint Backlog* – Agilo

O gráfico mostrado na figura 7 (*Sprint Burndown Chart*) gerado a partir da ferramenta, possibilita a visualização das tarefas ao longo dos dias no intervalo do *Sprint*.

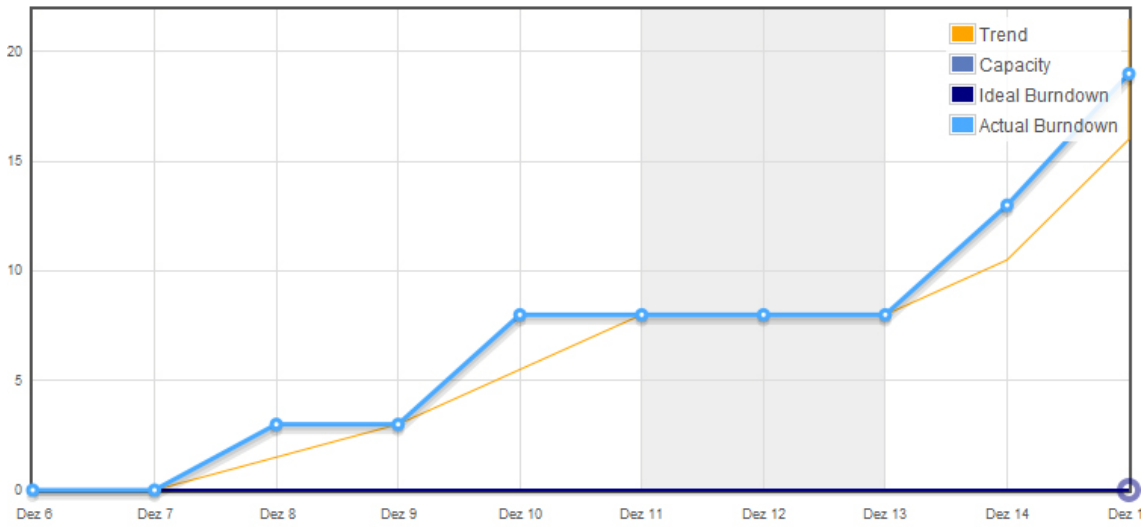

Figura 7: *Sprint Burndown Chart* - Agilo

5 – Resultados Alcançados

A metodologia utilizada para a avaliação do processo foi feita por meio da aplicação de questionários antes e depois da implantação do *Scrum* na empresa. Os funcionários e gestores envolvidos no processo foram orientados a responderem às questões de forma imparcial e com base na sua visão pessoal acerca do projeto proposto.

O primeiro questionário (Apêndice A), aplicado no início do processo, teve como objetivo estimar a situação de desenvolvimento de *software* na empresa antes do processo de adoção do método ágil, possuindo questões de avaliação a respeito do desenvolvimento e gerenciamento de projetos na CTEC.

O questionamento inicial avalia o processo de gerenciamento de *software* na empresa. A maioria respondeu que o desenvolvimento de *software* na empresa era gerenciado parcialmente.

A questão seguinte analisou o planejamento das mudanças de prazo, escopo e custo. Todos responderam que estes quesitos raramente eram planejados de maneira formal, tendo por base a experiência do desenvolvedor, podendo o projeto ou recurso extrapolar os custos e tempo estimados pelo cliente.

Com base nos prazos, de acordo com as respostas, quase sempre os mesmos não eram cumpridos quando são planejados.

O ponto seguinte diz respeito à avaliação individual de como a implantação de um método de gerenciamento auxiliaria no processo de desenvolvimento. De acordo com o ponto de vista dos gestores, possibilitaria um controle mais efetivo da empresa em relação aos desenvolvedores. Pontos destacados nas respostas dos demais são: auxílio ao acompanhamento das tarefas e cumprimento dos prazos, desenvolvimento mais rápido, com redução dos riscos inerentes, na organização das tarefas de cada profissional envolvido no projeto e no controle de qualidade do *software*.

Dentre as possíveis dificuldades que poderiam ser encontradas durante a implantação do processo estão a conscientização dos profissionais envolvidos, as reuniões diárias próprias do *Scrum,* a mudança da rotina de

trabalho dos funcionários, a adequação dos desenvolvedores aos métodos e modelos novos e a resistência dos mesmos ao processo de implantação.

A maioria respondeu que haveria sim a resistência dos funcionários ao processo, pelo cumprimento das tarefas em determinados prazo, aumentando a regularidade das supervisões. Tal mudança alteraria a forma habitual de trabalho.

Na avaliação do método utilizado para dimensionar os prazos para a realização das tarefas foi predominante nas respostas o método Empírico, baseado na experiência dos profissionais envolvidos e de acordo com a complexidade das tarefas a realizar e de acordo com a necessidade de urgência para a realização das tarefas para o cliente.

No que diz respeito ao orçamento e cronograma dos projetos, incluindo a definição de marcos e pontos de controle, todos responderam que estes não são estabelecidos e mantidos.

Avaliando os riscos dos projetos e a identificação do seu impacto, probabilidade de ocorrência e prioridade de tratamento, todos responderam que não são documentados nem determinados os riscos e as características citadas.

Com relação à avaliação dos recursos humanos e planejamento dos mesmos considerando o perfil e conhecimento, a maioria respondeu que os membros da equipe são alocados para as suas atividades de acordo com sua habilidade para tal.

Todos responderam que os recursos e o ambiente de trabalho necessários para a execução dos projetos são planejados.

Avaliando as metas dos projetos, considerando as restrições e os recursos disponíveis, todos responderam que os mesmos são considerados.

Um outro questionário (Apêndice B) foi aplicado após o processo de implantação do *Scrum* e aplicação dos princípios do método ágil à rotina de trabalho dos integrantes da equipe de desenvolvimento e suporte.

Inicialmente, foi avaliado o processo de manutenção do *software,* se o houve mudança com relação ao mesmo. A maioria disse que sim e que houve a manutenção da qualidade do nível anterior.

Com relação aos prazos de atendimento, a maioria disse que os mesmos permaneceram inalterados e no que diz respeito ao nível de

satisfação dos clientes, também permaneceram com o mesmo nível de satisfação.

O questionário segue a sequência considerando o desenvolvimento de software. A maioria disse que a mudança foi significativa e representa uma melhoria na qualidade do processo.

Dentre as mudanças notadas durante a aplicação da metodologia, estão, por parte dos gestores, um controle das tarefas e a carga horária dispensada por cada desenvolvedor para a realização das mesmas. Por parte dos funcionários, destaca-se a divisão das tarefas em intervalos de tempo (*Sprint*) e alimentação da ferramenta de acordo com a urgência dos problemas a serem resolvidos. Um ponto positivo é que o tempo é definido pelo desenvolvedor, ou seja, ele direciona seu trabalho conforme sua avaliação para o cumprimento de suas atividades.

As dificuldades encontradas para a implantação do *Scrum* na CTEC, como já haviam sido previstas no questionário anterior, foram a resistência à mudança na forma de trabalhar e a adequação às atividades diárias do *Scrum.*

A pergunta seguinte completa as dificuldades discutidas anteriormente, questionando a existência de resistência por parte dos funcionários e os motivos pelos quais se deram essa resistência. A maioria disse que houve resistência, por diversos motivos, dentre eles: muitos consideraram trabalhoso o fato de alimentar a ferramenta e descrever as tarefas, atribuindo uma carga horária de trabalho para a finalização destas.

A figura 6 exibe as informações do *Sprint Backlog* retiradas do *Agilo.*

A criação do *Sprint* e do *Sprint Backlog* define as prioridades e os desenvolvedores decidem no que vão trabalhar e o tempo demandado para a realização das tarefas.

As dificuldades encontradas durante o processo de implantação foram, basicamente, a falta de interesse dos funcionários em utilizar a ferramenta e a resistência às mudanças, desde as reuniões diárias às reuniões de planejamento.

Algo que modifica a rotina dos integrantes da equipe, já acostumados a trabalhar de determinada forma, deve ser implantado de forma gradual. Alguns funcionários resistiram também pelo fato de ter que alimentar a ferramenta e

informar quantas horas faltam para que termine as tarefas as quais lhe pertencem.

Os primeiros *Sprints* não tiveram muito sucesso em seu resultado, por conta das dificuldades citadas acima. A partir do terceiro *Sprint* foi que os envolvidos no processo puderam interagir e alimentar o *Agilo* conforme o *Scrum.*

6 – Conclusão

Muitas empresas não possuem controle de seus processos, riscos, custos e prazos. Alguns projetos fracassam pela inexistência de gerenciamento ou pobre gerenciamento que planeje a alocação dos recursos humanos e financeiros, bem como o controle dos riscos e acompanhamento do desenvolvimento.

O objetivo desse trabalho foi acompanhar o processo de implantação de um método ágil de gerenciamento de projetos em uma empresa de Vitória da Conquista, com a finalidade de atingir o nível o nível dois do CMMI e nível G do MPS.BR, cujos processos encontram-se gerenciados ou parcialmente gerenciados. Porém, esses níveis não foram atingidos, tendo em vista que a implantação ainda está sendo feita.

A empresa encontra-se em processo de mudança, conscientização e treinamento dos profissionais envolvidos. A implantação do método ágil está sendo feita de forma gradativa.

Muitos problemas foram encontrados durante o processo, destacando-se a resistência por parte dos funcionários e aconselha-se, para a realização de trabalhos futuros, após a fase de planejamento do *Sprint,* a motivação dos funcionários e a conscientização de que a implantação é necessária como forma de agilizar os processos e controlar os mesmos, objetivando produzir um *software* de qualidade e de maior aceitação por parte dos clientes.

7 – Referências

ABNT, ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. ISO/IEC 12207 – Tecnologia da Informação – Processos de ciclo de vida de software. Rio de Janeiro: ABNT, 1996.

AMBLER, Scott W.. *Modelagem ágil, práticas eficazes para a programação extrema e o processo unificado.* 1. ed. São Paulo: Bookman, 2002.

BECK, Kent. *Programação Extrema Explicada*. São Paulo: Bookman, 1999.

DAYCHOUW, Merhi. *40 ferramentas e técnicas de gerenciamento.* Rio de Janeiro: Brasport, 2007.

http://www.dca.fee.unicamp.br/projects/sapiens/Seminars/2002/r20020823.pdf. Acesso em 18/04/2010.

http://engenhariadesoftware.blogspot.com/2007/02/ciclo-de-vida-do-software-parte-1.html. Acesso em: 18/04/2010

FERNANDES, Aguinaldo Aragon; ABREU, Vladimir Ferraz de. *Implantando a governança de ti: da estratégia à gestão de processos e serviços.* Rio de Janeiro: Brasport, 2009.

FILHO, Wilson De Pádua Paula. *Engenharia de software: fundamentos, métodos e padrões.* Rio de Janeiro: Brasport, 2000.

Guia Geral de Implementação do MPS.BR, versão 1, 2006.

JUNIOR, Roque Rabechini. *Competências e Maturidade em Gestão de Projetos, uma perspectiva estruturada.* Rio de Janeiro: Annalumbe, Fpesb, 2005.

KERZNER, Harold. *Gestão de projetos: as melhores práticas.* 2. ed. São Paulo: Bookman, 2004.

MARTINS, José Carlos Cordeiro. *Técnicas para gerenciamento de projetos de software.* 1. ed. Rio de Janeiro: Brasport, 2007.

PEREIRA, Fábio M. *Proposta de implantação de processo de gerenciamento de software*. CTEC. Vitória da Conquista, 2009.

PMI - Project Management Institute – PMI. Site oficial do PMI, http://www.pmi.org acesso em: 9 de maio de 2010.

POSSI, Marcus. *Gerenciamento de projetos, guia do profissional:* Abordagem geral e definição de escopo. 1. ed. Rio de Janeiro: Brasport, 2006.

PRIKLADNICKI, Rafael; ORTH, Afonso Inacio. *Planejamento e gerencia de projetos.* Porto Alegre: Edipucrs, 2009.

REZENDE, Dênis Alcides. *Engenharia de software e sistemas de informação.* Rio de Janeiro: Brasport, 2005.

SCHWABER, Ken *e* SUTHERLAND. Jeff. *Guia do Scrum*, 2009.

SOMMERVILLE, Ian. *Engenharia de software.* 8. ed. São Paulo: Prentice-hall, 2006.

Um Guia do Conjunto de Conhecimentos em Gerenciamento de Projetos, PMBOK, Terceira edição, 2004.

VARGAS, Ricardo Viana. *Gerenciamento de projetos, estabelecendo diferenciais competitivos.* 6. ed. Rio de Janeiro: Brasport, 2005.

Apêndice A

PROPOSTA DE IMPLANTAÇÃO DE PROCESSO DE GERENCIAMENTO DE SOFTWARE – FORMULÁRIO DE AVALIAÇÃO

Vitória da Conquista, 20 de setembro de 2010.

Este formulário visa avaliar o processo de implantação de gerência de software da Conquista Tecnologia. Esperamos que responda às perguntas de acordo com a sua visão pessoal acerca do projeto proposto desde o início de sua implantação em janeiro/2010.

1. Como você avalia o processo de desenvolvimento de software?
   [ ] Gerenciado
   [ ] Gerenciado parcialmente
   [ ] Não gerenciado

2. As mudanças de prazo, escopo e custo são planejadas?
   [ ] Sim [ ] Não

3. Com relação aos prazos, são cumpridos como planejados (se houver planejamento) inicialmente?

   [ ] Sim [ ] Não

4. Em que a implantação de um método ágil de gerenciamento auxiliaria no processo e desenvolvimento?

   _____

   _____

   _____

5. Quais as dificuldades podem ser encontradas para implantação do processo de gerenciamento no desenvolvimento de software?

   _____

   _____

   _____

6. Pode haver alguma resistência dos funcionários ao processo de mudança?
   [ ] Sim [ ] Não

7. Se sim, quais os motivos dessa resistência?

   _____

   _____

Apêndice B

PROPOSTA DE IMPLANTAÇÃO DE PROCESSO DE GERENCIAMENTO DE SOFTWARE – FORMULÁRIO DE AVALIAÇÃO

Vitória da Conquista, 20 de dezembro de 2010.

Este formulário visa avaliar o processo de implantação do processo de gerência de software da Conquista Tecnologia. Esperamos que responda às perguntas de acordo com a sua visão pessoal acerca do projeto proposto desde o início de sua implantação em janeiro/2010.

PARTE I – MANUTENÇÃO DE SOFTWARE

8. Com relação ao processo de manutenção de software, você notou alguma mudança?
   [ ] Sim [ ] Não

9. Se sim, esta(s) mudança(s) representaram uma:
   [ ] Melhoria na qualidade do processo
   [ ] Manutenção da qualidade no nível anterior
   [ ] Piora na qualidade do processo

10. Com relação aos prazos de atendimento, estes:
    [ ] Diminuíram substancialmente
    [ ] Diminuíram razoavelmente
    [ ] Permaneceram inalterados
    [ ] Aumentaram razoavelmente
    [ ] Aumentaram substancialmente

11. Com relação ao nível de satisfação de seus clientes, você diria que estes:
    [ ] Estão mais satisfeitos
    [ ] Permanecem com o mesmo nível de satisfação
    [ ] Estão mais insatisfeitos

PARTE II – DESENVOLVIMENTO DE SOFTWARE

12. Com relação ao processo de desenvolvimento de software, você notou alguma mudança?
    [ ] Sim [ ] Não

13. Se sim, esta(s) mudança(s) representaram uma:
    [ ] Melhoria na qualidade do processo
    [ ] Manutenção da qualidade no nível anterior
    [ ] Piora na qualidade do processo

14. Você notou alguma mudança com relação à metodologia de desenvolvimento?
    [ ] Sim [ ] Não

15. Se sim, que mudanças foram essas?

_____

_____

_____

16. Quais as dificuldades encontradas para implantação do processo de gerenciamento no desenvolvimento de software?

_____

_____

_____

17. Houve alguma resistência dos funcionários ao processo de mudança?
[ ] Sim [ ] Não

18. Se sim, quais os motivos dessa resistência?

_____

_____

_____