ÉVERTON MOLINA DA SILVA

ESTRATÉGIA DE COMBINAÇÃO ENTRE TIMES DE ALTO DESEMPENHO E A METODOLOGIA ÁGIL SCRUM NO DESENVOLVIMENTO DE SOFTWARE

SÃO PAULO
2011

ÉVERTON MOLINA DA SILVA

ESTRATÉGIA DE COMBINAÇÃO ENTRE TIMES DE ALTO DESEMPENHO E A METODOLOGIA ÁGIL SCRUM NO DESENVOLVIMENTO DE SOFTWARE

Monografia apresentada ao Programa de Educação Continuada (PECE) da Escola Politécnica da Universidade de São Paulo para obtenção do título de MBA em Tecnologia da Informação.

SÃO PAULO
2011

ÉVERTON MOLINA DA SILVA

ESTRATÉGIA DE COMBINAÇÃO ENTRE TIMES DE ALTO DESEMPENHO E A METODOLOGIA ÁGIL SCRUM NO DESENVOLVIMENTO DE SOFTWARE

Monografia apresentada ao Programa de Educação Continuada (PECE) da Escola Politécnica da Universidade de São Paulo para obtenção do título de MBA em Tecnologia da Informação.

Área de concentração:
Tecnologia da Informação.

Orientador:
Prof. Raymundo Pedro da Silva Vasconcelos.

SÃO PAULO
2011

FICHA CATALOGRÁFICA

Silva, Éverton Molina da
Estratégia de combinação entre times de alto desempenho e a metodologia ágil Scrum no desenvolvimento de software/ E. M. da Silva. - São Paulo, 2011.
40p.

Monografia (MBA em Tecnologia da Informação) – Escola Politécnica da Universidade de São Paulo. Programa de Educação Continuada em Engenharia.

1. Métodos ágeis 2. Engenharia de software 3. Processo de software 4. Desenvolvimento de software I. Universidade de São Paulo. Escola Politécnica. Programa de Educação Continuada em Engenharia II. t.

AGRADECIMENTOS

Á Deus por suas bênçãos nessa jornada.

A minha família, meus pais, irmã e namorada pelo apoio, paciência e por sempre acreditarem em mim.

Aos amigos e colegas de trabalho, que me incentivaram para a realização deste trabalho.

Ao prof. Raymundo Pedro da Silva Vasconcelos, pela orientação e estimulo no desenvolvimento deste trabalho.

Liderança é uma função de grupo ou equipe. O trabalho do líder é criar as condições para que a equipe seja eficaz.

(Robert Ginnett)

RESUMO

Este trabalho visa apresentar a estratégia de combinação entre Times de Alto Desempenho e a metodologia SCRUM em equipes que trabalham com desenvolvimento de software, possibilitando o desempenho máximo da equipe com maior eficácia. O trabalho utiliza a metodologia SCRUM como elemento central no atendimento das grandes demandas de desenvolvimento de software, transformando o desejo do cliente em software funcional, com aumento da qualidade de software, em conformidade com a especificação e entregue no prazo determinado. Esta abordagem tem como objetivo garantir que o desenvolvimento de software seja um processo cada vez mais eficaz e que haja uma melhoria continua sobre os resultados obtidos, sendo assim, propõe a adoção de Times de Alto Desempenho no atendimento de alguns aspectos considerados fundamentais para que um time seja considerado de alto desempenho, aspectos estes que não estão presentes em times SCRUM, um exemplo é a atenção especial na seleção dos membros que irão integrar a equipe, assegurando uma equipe formada por profissionais com conhecimentos complementares e com alto nível de conhecimento.

Palavras-Chave: Scrum. Times de alto desempenho. Metodologias ágeis. Desenvolvimento de software.

ABSTRACT

This work purpose is to demonstrate the strategy of combining High Performance Team and the SCRUM method with teams that work on software development providing high performance and more effectively. This paper uses the SCRUM methodology as a main element to attend the high demand for software development, transforming the customer's wishes in functional software, with quality driven software, software in accordance with the requirements specification and delivered in time. The objective of this approach is to ensure more efficiency in software development process and also to ensure a continued improvement over the results, therefore, proposes the adoption of High Performance Teams to attend some essential aspects in a team to be considered a High Performance Team, aspects that are not presents in SCRUM teams, as a special attention to select team members, ensuring a team of professionals with complementary skills and a high level of knowledge.

Keywords: Scrum. High performance teams. Agile methodologies. Software development.

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

TELM – Team Effetivenness Leadership Model

TI – Tecnologia da Informação

ROI – Return On Investment

PDCA – Plan, Do, Check, Act

BPMN – Business Process Modeling Notation

UML – Unified Modeling Language

SUMÁRIO

1 INTRODUÇÃO

Atualmente é essencial que um novo sistema seja desenvolvido rapidamente para aproveitar as novas oportunidades e responder às pressões competitivas. Desenvolvimento e entrega rápida são, portanto, muitas vezes o requisito mais crítico para sistemas de software (SOMMERVILLE, 2007).

Neste contexto onde as entregas devem ser rápidas e os requisitos do sistema podem mudar rapidamente durante o processo de desenvolvimento, surgiram metodologias ágeis de gestão de projetos de software que contam com uma abordagem iterativa para especificação, desenvolvimento e entrega de software. Com a utilização de métodos tradicionais, como por exemplo, os processos em cascata de desenvolvimento, existem equipes independentes para análise de requisitos, especificação, desenvolvimento, teste, implantação, entre outras, todos os passos de uma etapa devem ser concluídos para que o próximo inicie, sendo assim uma simples mudança de especificação no inicio do projeto que já está em fase de teste ou implementação, por exemplo, pode vir a impactar a entrega do projeto todo.

Métodos ágeis têm uma abordagem iterativa e baseada em entregas parciais de produtos e/ou funcionalidades executáveis que acrescentem valor para o negócio, ou seja, se faz a especificação, desenvolvimento, teste e implantação de apenas uma funcionalidade ou parte do produto, isso fornece mais flexibilidade e controle no desenvolvimento de software onde qualquer alteração de especificação impactará somente parte do projeto, pois o mesmo não foi totalmente especificado.

Porém para que um *Time de Alto Desempenho* seja bem sucedido é fundamental a presença de profissionais qualificados e capacitados para levar a equipe ao desempenho máximo e realizar as entregas dos objetivos rapidamente e com qualidade. Neste contexto *Times de Alto Desempenho* podem ser adotados para a formação de uma equipe ágil, pois são guiados pelo sonho (objetivo), envolvendo toda a organização na busca pelo mesmo, levando em consideração as responsabilidades da corporação, do time e do indivíduo na mesma direção fornecendo a sinergia necessária para o alto desempenho, enquanto equipes em

geral (ágeis ou não), são guiadas com o foco somente no processo de trabalho do time (Robert Ginnett, 2005).

1.1 Objetivos

Times de Alto Desempenho assim como a metodologia *SCRUM* apresentam características similares, como o foco nas pessoas, colaboração e comunicação. Ambas destinam-se a melhora da eficácia em equipes através do esforço em conjunto, comunicação eficaz e outras características, entretanto, times *SCRUM* por estarem contidos dentro de uma metodologia de desenvolvimento de software não atendem a todos os aspectos necessários para que sejam considerados *Times de Alto Desempenho*, um exemplo a ser considerado é a presença de membros com alto grau de conhecimento e ao mesmo tempo em que esses conhecimentos sejam complementares, o que exige um processo seletivo minucioso dos integrantes. Sendo assim, o objetivo deste trabalho é apresentar a adoção de *Times de Alto Desempenho* dentro da metodologia *SCRUM* de gestão de projetos com intuito de endereçar os pontos fracos da metodologia e atendendo as demandas de desenvolvimento de software com entrega rápida e com qualidade.

Apresentar a técnica e processo de formação de um *Time de Alto Desempenho* assim como demonstrar que a adoção desses times garante a sinergia, motivação, melhora da comunicação entre os membros do time além de compartilhamento de sonhos e desafios levando a equipe ao desempenho máximo possível. Mostrar como é o processo seletivo, quais as características individuais e competências o responsável por formar uma equipe deve buscar nos candidatos, qual o ambiente e recursos necessários para a entrega do objetivo por parte do time, como a combinação de *Times de Alto Desempenho* com a metodologia ágil *SCRUM* através de suas técnicas garantem maior velocidade de desenvolvimento, além de paralelismo, entregas parciais com produto funcional e software de qualidade sem burocracia com o time inteiro focado em um único objetivo.

1.2 Justificativa e Motivações

Desenvolvimento de software é um processo complexo onde um bom gerenciamento, criatividade, organização, ferramentas e metodologias são fatores determinantes para o seu sucesso ou insucesso. Além disso, esses fatores podem levar à agilidade no desenvolvimento, e hoje em dia ser ágil no lançamento de novos produtos de software é fazer diferença e estar à frente de concorrentes.

As metodologias ágeis provaram ser de fácil implementação e que reduzem a resistência a mudanças, a adoção destas metodologias fornece alto grau de liberdade, velocidade de desenvolvimento, maior qualidade de software e diminuição de burocracia como extensas documentações e geração excessiva de evidências (Martins, 2007). Porém existem pontos fracos, ou dificuldades na adoção dessas metodologias, um exemplo é a não presença de um time com alto nível de conhecimento que facilite a execução de tarefas interdisciplinares. Neste caso *Times de Alto Desempenho* podem estar ajudando com a formação de equipes eficazes desde que as condições favoráveis ao desenvolvimento dos times estejam presentes (Bejarano, et al., 2006).

1.3 Metodologia

Pesquisa Bibliográfica: Estudo das referências de sustentação dos conceitos sobre *Times de Alto Desempenho* assim como estudo das referências sobre a metodologia ágil *SCRUM* a fim de correlacioná-las na busca por competitividade no desenvolvimento de software de forma rápida e eficaz.

Proposição do Método: Nesta fase serão feitas as correlações entre o método e os conceitos de *Times de Alto Desempenho* que atendem e respondem as fraquezas da metodologia *SCRUM* na busca por competitividade, agilidade e eficácia no desenvolvimento de software.

Conclusão: Nesta última fase são apresentadas as conclusões com afirmações demonstradas no trabalho sobre os ganhos no desempenho e velocidade, assim

como uma série de outras vantagens no desenvolvimento de software através da adoção de *Times de Alto Desempenho* e da metodologia *SCRUM* na gestão de projetos de software.

1.4 Estrutura do trabalho

A apresentação deste trabalho segue a seguinte seqüência de capítulos:

Capítulo 1: introdução ao trabalho, seus objetivos, as motivações e justificativas que levaram ao desenvolvimento desta dissertação, assim como a abrangência e a metodologia utilizada.

Capítulo 2: descreve a origem e o porquê do trabalho em equipe, os conceitos e principais características sobre *Times de Alto Desempenho*, TELM, métodos ágeis e da metodologia *SCRUM.*

Capítulo 3: descreve sobre o modelo proposto para formação e gerenciamento de *Times de Alto Desempenho* que atuará no desenvolvimento de software utilizando a metodologia de gestão de projetos *SCRUM.*

Capítulo 4: conclusões.

2 TIMES DE ALTO DESEMPENHO E A METODOLOGIA ÁGIL SCRUM NO DESENVOVIMENTO DE SOFTWARE

Empresas vêm apresentando mudanças no modelo organizacional nas últimas décadas devido às potencialidades que o trabalho em equipe fornece. No modelo de trabalho em equipe a estrutura formada por departamentos e funções deixam de existir (Bejarano, et al., 2006). Em uma equipe formada para o desenvolvimento de um novo software, por exemplo, podem estar presentes engenheiros de software, especialistas no produto, analistas de sistemas, programadores, testadores entre outros.

Entretanto, apesar das diversas vantagens que o trabalho em equipe provê, para que uma equipe seja considerada de alto desempenho existe uma série de obstáculos que devem ser superados, e segundo Bejarano (2006) observa-se também que o desenvolvimento de uma verdadeira mentalidade de trabalho em equipe é algo complexo para ser administrado, faz-se então a necessidade de reconhecer e empenhar esforços para superar as dificuldades na formação destas equipes, já que o trabalho em equipe não é natural para maior parte das pessoas.

Com diversas características presentes em *Times de Alto Desempenho* existem às equipes que trabalham com a metodologia *SCRUM* de desenvolvimento de software, como exemplo é possível citar a presença de equipes multidisciplinares, caracterizada pela presença de programadores capazes de desempenhar mais de um tipo de tarefa, outra característica é a atenção especial dada à comunicação entre os membros da equipe. O *SCRUM* é baseado no sistema Toyota de Produção que surgiu na década de 70 após a crise do petróleo, na época um grande problema dos japoneses era a enorme falta de recursos que os impedia de adotar a produção em massa de Henry Ford, no modelo Toyota a produção se dá em pequenos lotes o que elimina a necessidade de grandes estoques entre outros desperdícios como superprodução e excesso de recursos.

2.1 Times de Alto Desempenho

O trabalho em equipe tem o potencial de aumentar a produtividade assim como diminuir os custos em projetos através da reunião de talentos, estimulo da criatividade e criação de um ambiente para solução de problemas. Baseado nesses potenciais, as organizações têm feito mudanças em seus modelos organizacionais nas últimas décadas, ao invés de organizar o trabalho por funções e departamentos as empresas estão adotando estruturas baseadas em equipes (Bejarano, et al., 2006).

Times de Alto Desempenho são equipes caracterizadas por possuírem liberdade para estabelecer métodos de trabalho, isso faz com que os membros dos times sintam-se úteis e que contribuam com a corporação para a realização das metas que são definidas com base na missão, visão e valores da corporação. A missão e visão são capazes de dar a equipe um incrível senso de direção, ou seja, aonde a empresa quer chegar (Pham, et al., 2005). Diretamente ligado ao desempenho máximo das equipes está também o sentimento de autoridade que motiva os membros da equipe a poder agir e tomar decisões (Bejarano, et al., 2006), outra característica importante que deve ser considerada em *Times de Alto Desempenho* é a diferença entre o "presto contas a meu chefe" e "prestamos contas a nós mesmos" (Katzenbach e Smith 2001).

Segundo Davis (2007) *Times de Alto Desempenho* são necessários para as organizações quando a complexidade do trabalho é alta e o prazo para a realização do mesmo é curto, sendo assim um time com profissionais especialistas, com conhecimentos complementares e trabalhando em harmonia pode fazer com que o sucesso no trabalho seja alcançado.

A estratégia da construção do *Time de Alto Desempenho* é feita com enfoque específico para o atendimento de um objetivo, sendo assim, quando o objetivo é alcançado o time é imediatamente desfeito.

2.1.1 O Modelo de Liderança para Efetividade em Times

O Modelo de Liderança para Efetividade em Times (TELM) é capaz de identificar pontos de mudança em times que não estão desempenhando bem, além de apontar o caminho para soluções pragmáticas. O modelo lembra a abordagem da teoria de sistemas, com entradas à esquerda, ou seja, o indivíduo, a equipe e fatores organizacionais, com processos ao centro, isto é, o que se pode dizer sobre a equipe observando os membros da equipe no trabalho e por fim as saídas, ou seja, como a equipe cumpriu seus objetivos (Ginnett, 2005).

Segundo Ginnett (2005) criador do modelo TELM, todo time é composto pela sobreposição de três peças essenciais, o líder, os seguidores e a situação, sendo assim para liderar é necessário saber mais sobre times do que sobre liderança mesmo que poucos tenham sido treinados para isso. Ginnett também faz menção sobre a visão de liderança, com o conceito de líder nato, este conceito é um mito dissipado por décadas de liderança que dava a idéia de que um líder forte pode levar o trabalho ou projeto ao sucesso independentemente de seus seguidores ou situação. Neste modelo de líder nato acredita-se que liderança é uma habilidade mais de nascimento do que desenvolvida e que somente os atributos pessoais para liderança como força, inteligência e pragmatismo podem ser ensinados e quando aplicados podem criar equipes fortes com bons resultados.

Porém, segundo Ginnett (2005), em 1947 um estudo que visava definir traços de liderança provou o contrário, os lideres com os atributos necessários eram em geral fracos e os lideres sem os atributos eram freqüentemente fortes. Estudos mais recentes concluíram que liderança tem mais a ver com a forma como nos comportamos em torno de outras pessoas, bem como a forma colaborativa de trabalho para ganho mútuo e que um líder sem seguidores cooperativos não pode realizar as tarefas ou satisfazer as partes interessadas (Ginnett, 2005).

Ginnett ressalta a competição que existe em nossa sociedade, onde o "eu versus você" está implícito, a maioria das pessoas vão direto ao ganho pessoal ao invés de pensarem no ganho mútuo. A situação e os seguidores no modelo de liderança não podem ser subestimadas, porém poucos dão valor a seguidores e situação, não se vê universidades formando seguidores por exemplo. Mas somos todos seguidores, todos reportamos a alguém e é através das ações de seguidores

que os lideres chegam ao seu objetivo, quando um líder não leva a situação e os seguidores em consideração os resultados tendem a ser menos produtivos ou até fatal, como no caso citado por Ginnett (2005) onde um desastre acorreu em 1979 com a empresa aérea United Airlines quando um avião teve que pousar em uma floresta por falta de combustível. O capitão, que na época era o mais antigo capitão da frota United, ignorando o co-piloto e as advertências do engenheiro de vôo sobre o nível do combustível ele se concentrou somente em determinar a causa de um ruído ouvido no inicio do vôo. Depois de circular por mais de uma hora o combustível acabou e o avião caiu. Este foi realmente um erro de liderança por parte do capitão, mas num sentido mais amplo, foi um erro da equipe, uma combinação letal de falta de planejamento e coordenação entre o líder, os seguidores e da situação.

Em TELM, as mais importantes funções de liderança - Sonhos, Design e Desenvolvimento - são aplicadas através de entradas e contexto. Alcançar a eficácia da equipe requer atenção para as três funções críticas em liderança de equipes. Para criar um novo time eficaz primeiro deve-se definir a direção (Sonhos), que leva ao Design (Entradas), que por fim leva a o Desenvolvimento (Processo).

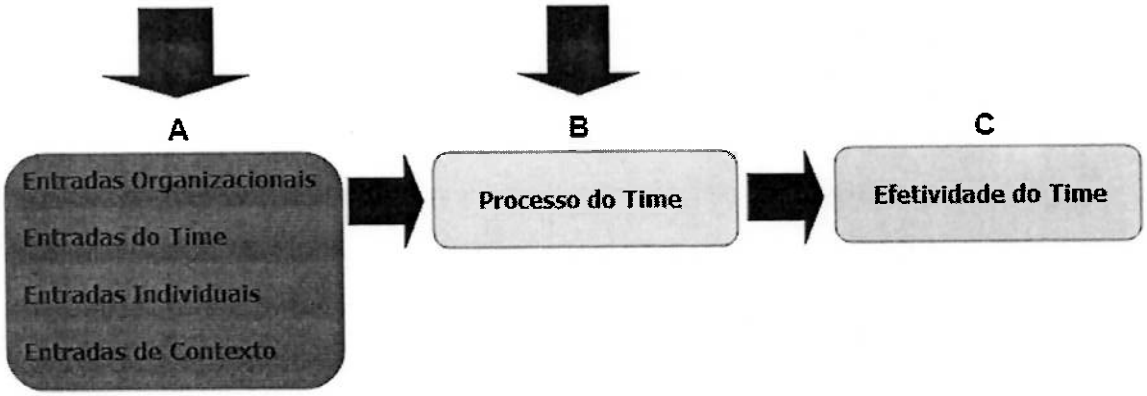

Figura 1 - Modelo TELM (Ginnett, 2005)

Sendo assim, a função de Design (A) é onde a maioria do trabalho é feito, é o componente que merece mais atenção, porém seja o componente mais omitido. Quando isto acontece, os custos de desenvolvimento muito provavelmente serão elevados. Líderes que querem *Times de Alto Desempenho* e resultados poupando dinheiro ou esforço inicial cometem um erro grave (Ginnett, 2005).

Para Ginnett (2005) os componentes do modelo TELM não são novos e sim o modo com que estão estruturados, para ele existem três saídas que são essenciais para que a equipe seja considerada eficaz, são elas:

- Resultados aceitáveis para as partes interessadas
- Capacidade futura da equipe: Deve melhorar ao longo do tempo.
- Satisfação pessoal: Isso não significa fazer do trabalho o ambiente para diversão, mas sim alcançar um ponto de equilíbrio onde as pessoas sintam prazer no que estão fazendo. Esta não é uma responsabilidade do líder, a equipe faz isso acontecer.

2.2 Metodologias Ágeis

O desenvolvimento ágil se tornou popular, diversas empresas como Google, Yahoo, Microsoft, entre outras estão adotando metodologias ágeis de desenvolvimento. Para Shore e Warden (2008) o desenvolvimento ágil não é recomendado somente para o aumento de produtividade, pois seus benefícios vêm do modo diferenciado de trabalhar e não do trabalho rápido. A equipe precisará de tempo para aprender o desenvolvimento ágil e enquanto aprendem serão mais lentos e não mais rápidos como o esperado. Sendo assim empresas que pensam em adotar o desenvolvimento ágil devem analisar se o desenvolvimento ágil irá realmente ajudar a terem mais sucesso.

A idéia tradicional de sucesso em projetos de desenvolvimento de software é a entrega no prazo, de acordo com o orçamento e especificações definidas no início do projeto, o Standish Group (1995) fornece algumas definições:

- *Bem sucedido*: Projeto completado a tempo, dentro orçamento e com todas as funcionalidades presentes na especificação.
- *Desafiado*: Projeto completo e funcionando, porém fora do orçamento, fora do prazo e sem algumas funcionalidades que deveriam estar presentes, pois estavam na especificação.
- *Defeituoso*: Projeto cancelado em algum momento durante o desenvolvimento.

Um projeto pode ser bem sucedido mesmo estando fora do prazo, orçamento e com funcionalidades que não estão de acordo com a especificação, pois esse projeto pode ser adorado pelo público-alvo. O mesmo pode ocorrer para projetos considerados bem sucedidos onde não conseguem atrair público alvo (Shore; Warden, 2008).

Segundo Machado (2009) o que faz as empresas adotarem as metodologias ágeis no desenvolvimento é a mudança constante nos ambientes empresariais que ocorrem em um ritmo muito rápido e o setor de T.I. na luta para acompanhar essas mudanças acabam adotando-as. Na verdade agilidade se resume na habilidade de criar e responder a mudanças com respeito ao resultado financeiro de um projeto num ambiente turbulento de negócios (Machado, 2009).

As metodologias ágeis prometem aumentar a satisfação do cliente com entrega do software no prazo e com alta qualidade, para que isso seja possível são utilizadas diversas práticas ágeis (Shore; Warden, 2008), como por exemplo:

- *Programação em pares*: Dobrando a capacidade mental disponível durante a programação.
- *Trabalho energizado*: Reconhece que os desenvolvedores fazem um trabalho melhor quando motivados e energizados.
- *Espaço de trabalho informativo*: Fornece a equipe a capacidade de ver o que está funcionando e o que não está.
- *Análise da causa raiz*: Utilizado como ferramenta para identificar as causas dos problemas.
- *Retrospectivas*: Analisa e melhora todo o processo de desenvolvimento.
- *Envolvimento do cliente*: Ajuda a equipe a entender o que ele realmente quer e o que deve ser desenvolvido.
- *Reuniões rápidas:* Ajuda a manter os membros da equipe informados sobre o andamento do projeto.
- *Padrões de Codificação:* Fornecem um modelo para unir as equipes.

Com a adoção das metodologias ágeis existe também o risco de rejeição por parte dos envolvidos, pois elas forçam uma mudança cultural na forma de pensar e tiram as pessoas da zona de conforto.

2.2.1 SCRUM

O termo *SCRUM* tem sua origem no jogo de Rugby, é a jogada que ocorre após uma falta para reiniciar o jogo onde todos os jogadores se posicionam juntos de forma aglomerada competindo a posse da bola e trabalham em conjunto para levar a bola ao campo adversário e fazer o ponto, ou seja, concretizar seu objetivo, o *Goal* (Machado; Medina, 2009).

Segundo Machado e Medina (2009) Hirotaka Takeuchi e Ikujiro Nonaka foram os primeiros a associar o termo SCRUM ao desenvolvimento de software no livro "*O jogo do desenvolvimento de novos produtos*", eles destacam a importância de se trabalhar em equipe, como uma unidade para atingir um objetivo comum. O *SCRUM* foi formalizado por Ken Schwaber e Jeff Sutherland em 1995, para eles *SCRUM* são times altamente integrados onde cada membro tem seu papel bem definido, porém todos estão focados no mesmo objetivo que é a entrega do produto.

O *SCRUM* possui uma curva de aprendizado baixa, é objetivo, flexível, possui metas claras, alto grau de cooperação e comprometimento. A abordagem do *SCRUM* diferentemente do modelo cascata de desenvolvimento não necessita que toda a análise do projeto esteja pronta para que o desenvolvimento seja iniciado, em *SCRUM* a análise é iniciada assim que alguns requisitos estiverem disponíveis, e assim que alguma análise estiver pronta se inicia o desenvolvimento, ou seja, o projeto trabalha em um pedaço pequeno de cada vez (Martins, 2007).

2.2.1.1 O Processo do SCRUM

Segundo Martins (2007) o projeto *SCRUM* inicia-se com uma visão, que no inicio pode não ser muito clara, mas que será mais bem entendida à medida que o projeto evoluir. Com a visão definida é montada uma lista priorizada, com todas as funcionalidades que devem ser desenvolvidas e de acordo com o valor que agrega ao negócio, está lista é chamada de ***Backlog* de Produto**.

Em todo inicio de uma iteração, ou ***Sprint***, como é chamada no *SCRUM,* existem Reuniões de Planejamento onde são definidas as funcionalidades

presentes no *Backlog* de Produto que vão estar presentes na iteração, essa lista de funcionalidade é conhecida como **Backlog do *Sprint***. A partir daí a equipe é deixada sozinha para que desenvolva o incremento de produto, no decorrer das duas ou quatro semanas que representam o *Sprint* a equipe realiza reuniões diárias com duração de no máximo 15 minutos onde o objetivo é sincronizar o trabalho de todos os membros do time. Diariamente a equipe também atualiza as tarefas que foram realizadas e as tarefas que serão realizadas até o próximo dia no quadro *SCRUM,* utilizado para o acompanhamento das tarefas de uma iteração, assim como o **gráfico de *Burndown*** que mostra a quantidade de trabalho restante no *Sprint*.

Ao final da iteração a equipe apresenta o incremento de produto ao cliente e interessados no negócio que vão avaliar a funcionalidade e solicitar alterações, estas serão adicionadas no *Backlog* de Produto e as prioridades serão revisadas. Também é feita uma **Reunião de Retrospectiva** para avaliar as ações que deram resultado positivo e negativo propondo mudanças na forma de trabalhar. O projeto só é encerrado quando todos os itens do *Backlog* de Produto estiverem desenvolvidos e o produto atendendo os objetivos.

A figura 2 representa o ciclo *SCRUM* de Desenvolvimento:

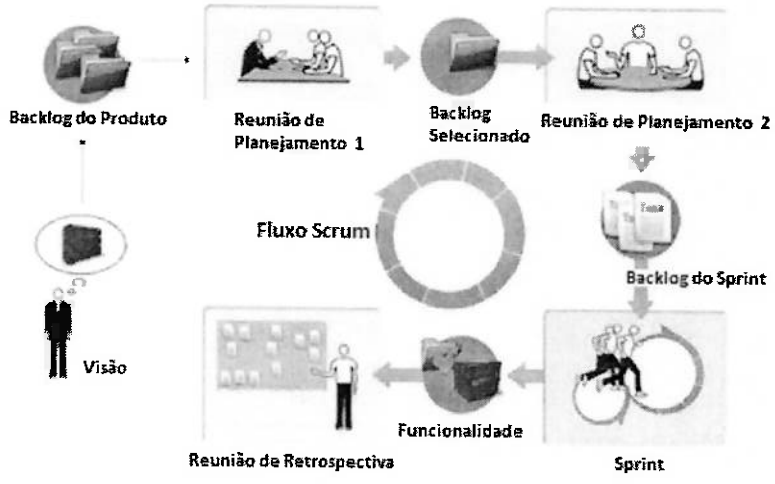

Figura 2 - O ciclo SCRUM de Desenvolvimento (Machado; Medina, 2009)

2.2.1.2 Papéis e Responsabilidades

Segundo Martins (2007) o *SCRUM* trabalha basicamente com os papéis e responsabilidades descritas a seguir:

- **Dono do Produto ou *Product Owner*:** É o responsável por representar os interesses das pessoas que apostaram no projeto, ele é quem consegue o recurso financeiro, define os objetivos do projeto e os objetivos de ROI, além de ser o responsável pelo *Backlog* do produto. Esse papel é normalmente desenvolvido pelo gerente de projetos, um membro da equipe de marketing ou um cliente interno da empresa.

- **Mestre Scrum ou *Scrum Master*:** *Scrum Master* é um líder e não um gerente, ele é o responsável por fazer com que os valores e práticas da metodologia sejam seguidos pelo time, responsável pela remoção de obstáculos que possam estar criando impedimentos para o andamento do trabalho do time, responsável por ensinar o Dono do Produto a maximizar o ROI e atingir seus objetivos através do *SCRUM,* além de também ser responsável por manter a informação sobre o progresso de trabalho da equipe atualizado e disponível para todos os interessados. Ou seja, o *Scrum Master* é o facilitador do time, que tem a missão de fazer o time funcionar.

- **Equipe do Projeto ou Scrum Team:** São os responsáveis por transformar os itens do *Backlog* de Produto em itens do *Backlog do Sprint* e transformá-los em funcionalidades de produto prontas para serem entregues. É a equipe que desenvolve o software e trabalha de forma participativa, é auto-gerenciada e formada geralmente por de 5 a 10 membros com diferentes especializações.

- **Interessados no Projeto ou Stakeholders:** São todos os interessados no produto que será desenvolvido, como clientes, usuários finais e outras equipes. São geralmente representados pelos Donos do Produto.

2.2.1.3 Fases

No *SCRUM* existem três fases: **pré-*game***, ***game*** e **pós-*game***, onde a primeira e última são constituídas em processos bem definidos. Já fase intermediária (*game*) o processo é empírico e a maioria dos processos nesta fase são tratados como uma caixa preta que requerem apenas controles externos. É na fase do *game* que o produto é construído em múltiplas iterações. O produto pode ser modificado a qualquer momento durante a iteração por conta de ações da concorrência, complexidade, prazo, qualidade, entre outros motivos.

2.2.1.3.1 Pré-Game

Esta fase é composta por duas partes: o planejamento e arquitetura do produto. No planejamento é definida uma nova entrega de software com base nas informações conhecidas até o momento, junto com a estimativa de prazo e custo, o processo de planejamento estabelece as expectativas dos interessados no produto. Na parte de arquitetura são definidas as bases para a construção do produto (Martins, 2007).

Para Martins (2007) o planejamento de uma nova entrega em *SCRUM* deve conter as seguintes variáveis, que podem mudar durante o projeto e por isso devem ser consideradas no planejamento dos vários *Sprints*:

- *Requisitos do cliente*: são as funcionalidades que precisam estar presentes no produto.
- *Prazo*: Prazo disponível para construir o produto.
- *Concorrência*: São as características dos produtos concorrentes que precisam ser consideradas na construção do novo produto.

- *Qualidade*: Nível requerido de qualidade.
- *Visão*: Mudanças ou funcionalidades que precisam ser consideradas para que o produto visionado seja construído.
- *Recursos*: pessoal ou outros recursos disponíveis.

O planejamento tem os seguintes passos:

- Desenvolvimento da lista de *Backlog* do Produto.
- Definição da data de entrega de cada incremento de software.
- Seleção da entrega que será abordado primeiro.
- Definição da equipe do projeto.
- Avaliação do risco e ações de controle.
- Revisão e ajuste dos itens do *Backlog* de Produto.
- Avaliação das ferramentas de desenvolvimento e infra-estrutura.
- Estimativa de custo da entrega do incremento de software.
- Obtenção da aprovação por parte da direção para o projeto e orçamento.

A arquitetura consiste em projetar os itens do *Backlog* de Produto com vistas à implementação, esta fase inclui definições e alterações na arquitetura do produto Martins (2007). Os passos desta etapa são:

- Fazer uma análise que reflita o contexto do sistema e os requisitos.
- Identificar qualquer problema que possa existir na implementação das mudanças.
- Rever os itens do *Backlog* de Produto e identificar as mudanças necessárias.
- Fazer reunião de revisão de produto para discutir as mudanças necessárias para implementar cada item do *Backlog*.

2.2.1.3.2 Game

Segundo Martins (2007) esta fase é composta por um ou mais *Sprints,* que tem como objetivo o desenvolvimento das funcionalidades de uma nova entrega. Esta fase é composta dos seguintes processos:

- Fazer reunião com as equipes para rever o planejamento das entregas.
- Revisar padrões com os quais o sistema precisa ser compatível e fazer os ajustes necessários.
- Realizar os *Sprints* até que o produto esteja pronto para entrega.

2.2.1.3.2 Pós-Game

Segundo Martins (2007) quando a gerência decide que as variáveis de tempo, concorrência de mercado, requisitos, custos e qualidade estão adequadas para a entrega do produto, ela declara que a entrega do incremento está fechada e a fase de pós-*game* se inicia. Nesta fase o produto é preparado para entrega que inclui testes adicionais, integração, documentação de usuário entre outros.

2.3 BPMN

O BPMN (Business Process Modeling Notation) como diz o nome, é uma notação de modelagem de processo de negócio representada graficamente, ele é muito similar a UML (Unified Modeling Language), linguagem de modelagem utilizada no desenvolvimento de software que permite uma maior visualização lógica do desenvolvimento (SOMMERVILLE, 2007), porém o BPMN tenha sido criado somente para relatar processos de negócio, não chegando ao nível de software.

O BPMN é composto por diversos elementos entre eles atividades, anotações, linhas, piscinas, eventos e subprocessos. O exemplo na figura 3 abaixo

exemplifica o modelo de processo de um atendimento médico onde o exame é realizado em uma entidade externa, no caso o laboratório.

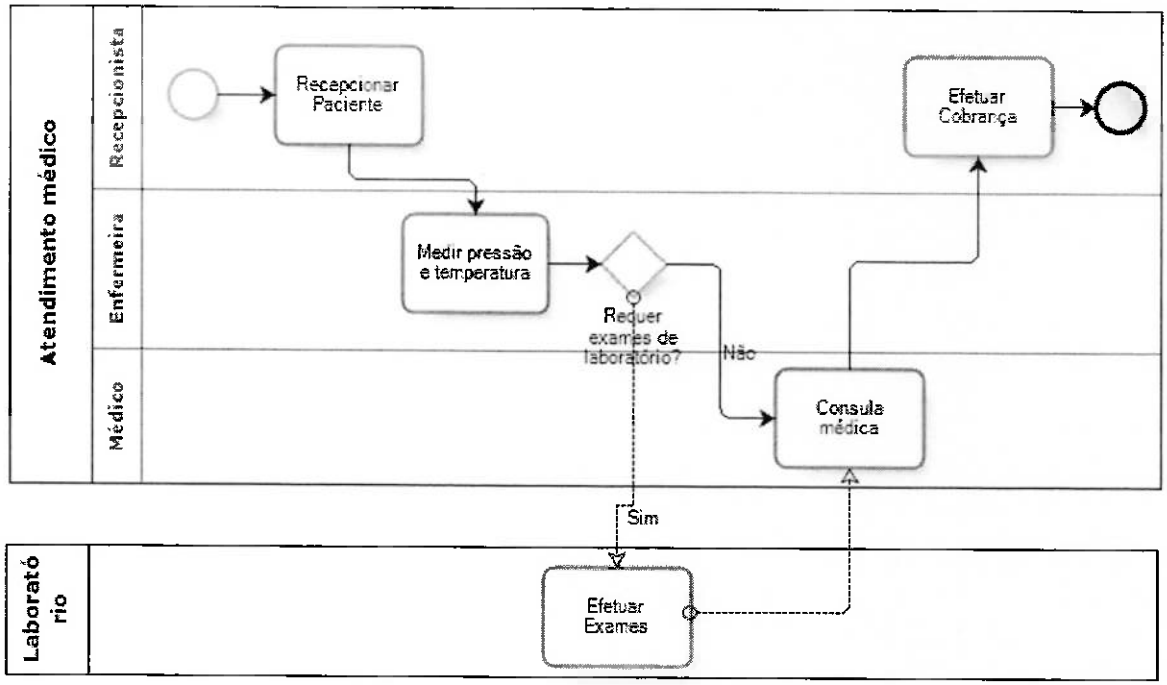

Figura 3 – Exemplo Básico de BPMN sobre o Processo de Atendimento Médico

No exemplo da figura 3 são utilizados basicamente os elementos:

- Piscina: representando o atendimento médico e laboratório
- Linhas: representando o médico, enfermeira e recepcionista.
- Tarefas: são os acontecimentos durante o processo.
- Gateway: faz a divisão do fluxo.
- Fluxo: Fazem a comunicação entre as tarefas.

3 TIMES EFICAZES E A METODOLOGIA ÁGIL SCRUM NO DESENVOLVIMENTO DE SOFTWARE

Muitas empresas têm investido somas significativas de verbas para desenvolvimento e manutenção de softwares, com a esperança de ganhar competitividade estratégica em seu mercado, porém, depois das implementações iniciais, muitas empresas percebem que seus investimentos iniciais muitas vezes não produzem o retorno e benefícios financeiros esperado, algo pode ter falhado no projeto, a entrega pode estar fora do prazo, o orçamento estourado ou o software pode não estar atendendo as especificações iniciais do projeto.

Tendo em vista os problemas comuns no processo de desenvolvimento de software, como entregas fora do prazo e orçamento, não conformidade do produto entregue com o especificado, entre outros, o objetivo é propor a adoção de *Times de Alto Desempenho* combinado ao uso do *SCRUM* como elemento de gestão atuando especificamente no processo de trabalho do time, onde a meta é melhorar a produtividade, qualidade de software, satisfação dos indivíduos que formam a equipe e realizar as entregas no prazo determinado.

A figura 4 mostra a adoção de *Times de Alto Desempenho* na metodologia ágil *SCRUM* de desenvolvimento de software.

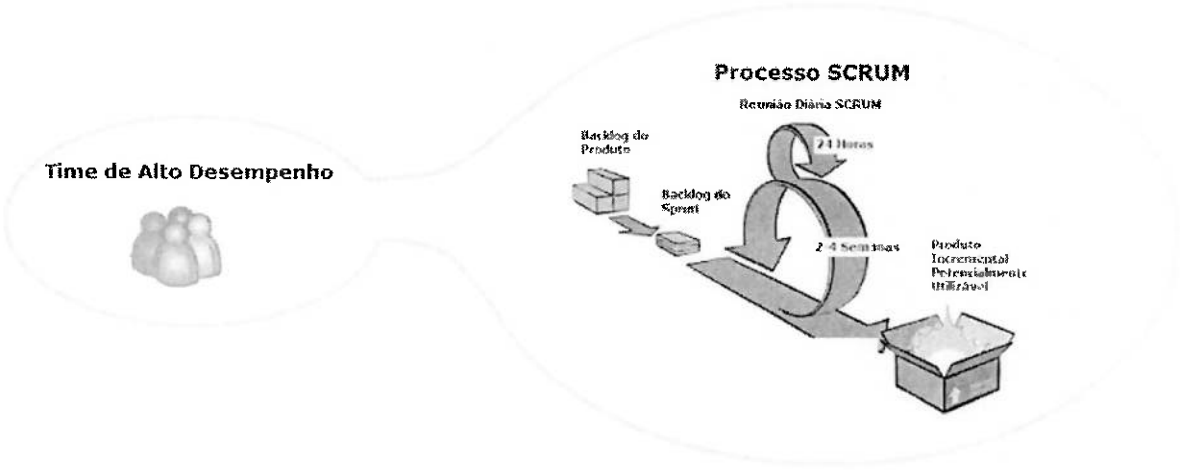

Figura 4 – A Adoção de Times de Alto Desempenho na Metodologia Ágil SCRUM.

3.1 Combinando Times de Alto Desempenho com a Metodologia Ágil SCRUM

Times auto-organizáveis, sob o selo de "ágil" tem muito do progresso para se tornar um *Time de Alto Desempenho*, porém empresas que adotam as metodologias ágeis em geral dão uma atenção limitada aos fatores humanos na formação e gestão das equipes, utilizando a metodologia somente como forma de controle e gestão no processo de trabalho de um time já existente, entretanto para que exista o desenvolvimento de software com agilidade e qualidade é necessária a presença de profissionais qualificados e capazes.

A figura 5 mostra os pontos positivos e pontos negativos de *Times de Alto Desempenho* assim como da metodologia *SCRUM,* quais os pontos em comum entre os dois e quais os pontos negativos do *SCRUM* que podem ser endereçados através da adoção de *Times de Alto Desempenho*.

SCRUM Pontos Positivos	Times de Alto Desempenho Pontos Positivos
Aumento na Velocidade de Desenvolvimento	Aumento na Velocidade de Desenvolvimento
Aumento na Comunicação entre os Integrantes do Time	Aumento na Comunicação entre os Integrantes do Time
Organização Paralela (Não Hierárquica)	Organização Paralela (Não Hierárquica)
Qualidade do Software	Autoridade aos Membros do Time
	Atenção Especial na Seleção dos Membros do Time
	Time com Nível Alto de Conhecimento
Pontos Negativos	**Pontos Negativos**
Atenção Limitada na Formação do Time	Dificuldades na Formação das Equipes
Time com Nível Alto de Conhecimento	Resistência por Parte dos Líderes
Não Existência de Prazos para Entregas	
Falta de Planejamento do Escopo	
Pouca Documentação	
Sensação de Informalidade	

Figura 5 – Pontos Positivos e Negativos da Metodologia SCRUM e Times de Alto Desempenho.

O *SCRUM* como técnica para gerenciamento de projetos de software apresenta uma série de benefícios como, por exemplo, o aumento na velocidade de desenvolvimento, aumento na comunicação entre os integrantes do time e a qualidade de software, porém não garante a verdadeira efetividade do time, pois existem diversos pré-requisitos para a formação de uma equipe *SCRUM* efetiva que devem estar presentes, como a presença de integrantes capacitados, responsáveis, criativos e dedicados na equipe, caso estes pré-requisitos não estejam presentes a equipe corre o risco de falhar e não desempenhar o suficiente e esperado.

Faz-se então a necessidade de times bem estabelecidos, ou seja, boas equipes, formadas por pessoas com as características, habilidades e aptidões necessárias para o desenvolvimento de software. A adoção de *Times de Alto Desempenho* pode suprir os pré-requisitos necessários a uma equipe *SCRUM*, pois em *Times de Alto Desempenho* ao contrário do *SCRUM* os fatores humanos demandam uma atenção especial desde a seleção dos membros do time até a realização de avaliações (questões feitas aos membros do time) sob o aspecto individual, do time e também sob o aspecto corporativo.

O *SCRUM* como metodologia de gestão em times bem estabelecidos pode ser eficiente, e a proposta é utilizar os princípios de *Times de Alto Desempenho* na formação dos times e a metodologia ágil *SCRUM* no gerenciamento do processo de trabalho do time, buscando assim atender as demandas de desenvolvimento de software de forma rápida e eficaz.

3.1.1 O Processo Seletivo

Este tópico tem como objetivo apresentar o processo de formação do *Time de Alto Desempenho* que atuará no desenvolvimento de software.

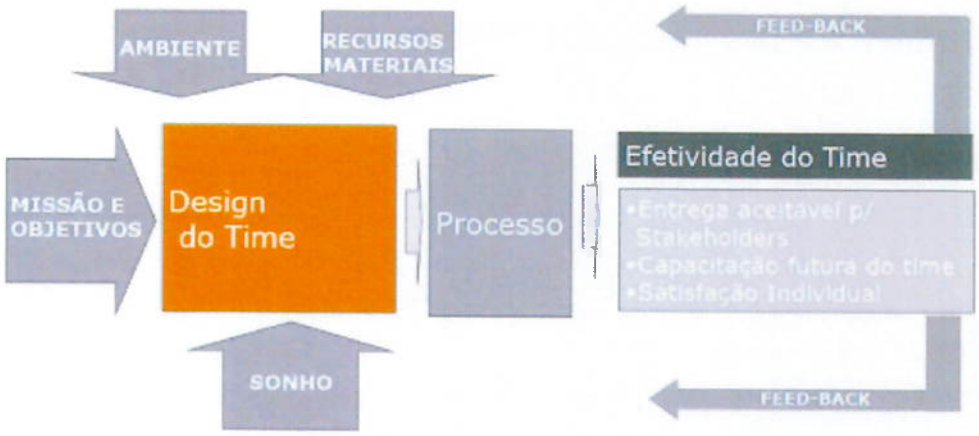

Figura 6 - Design do Time de Alto Desempenho (Robert Ginnett, 2005)

Equipes que trabalham no desenvolvimento de software necessitam de profissionais talentosos para que possam obter um bom desempenho e garantir que o produto de software seja entregue conforme o especificado e dentro do prazo. Neste contexto, profissionais de alto desempenho são necessários, pois devido à complexidade e ritmo com que as tarefas devem ser executadas somente trazendo especialistas e tendo um conjunto diversificado de competências complementares é possível alcançar o sucesso (Davis, 2007).

Segundo diversos pesquisadores falhas no processo seletivo podem ser um dos principais motivos para o insucesso em equipes (Bejarano, et al., 2006). Sendo o processo seletivo um fator determinante para o sucesso de equipes, a construção de um *Time de Alto Desempenho* deve se iniciar com uma boa seleção de profissionais, pois como nota Katzenbach e Smith (2001), o excelente desempenho é derivado de boas equipes, com profissionais capazes de trabalharem juntos por metas comuns.

Guillard (2009) aponta para algumas perspectivas sobre liderança e atitudes necessárias na seleção de pessoas para *Times de Alto Desempenho*:

- *Procurar entender as metas pessoais do candidato:* Selecionar pessoas que estão planejando suas carreiras com uma visão de longo prazo;
- *Conhecimentos complementares*: Selecionar a mistura certa de conhecimentos necessários para equipe, mapeando esses conhecimentos e selecionando as pessoas para as devidas necessidades;
- *Procurar o balanceamento entre perfis e experiências*: Essencial para promover o compartilhamento do conhecimento e experiências entre os membros do time.
- *O time participando do processo seletivo*: As pessoas com quem o candidato irá trabalhar devem querer trabalhar com ele;
- *Escolha de candidatos que demonstrem tendência para trabalho em equipe.*

Além destes pontos, uma série de competências, habilidades e aptidões são necessárias a um candidato que atuará na área de desenvolvimento de software. Para Larson e LaFasto (1989) é primordial que a seleção seja baseada na capacitação em dois níveis:

- Em habilidades e competências técnicas.
- Em habilidades e competências comportamentais.

A idéia para alcançar o balanceamento entre as habilidades e competências no time é que os responsáveis pela contratação relacionem as habilidades e competências (tanto técnicas quanto comportamentais) necessárias ao time que irá trabalhar com desenvolvimento de software, fazendo uma análise do que o time necessita, o que o time já possui e o que deve ser aprimorado no time.

A figura 7 apresenta um exemplo de tabela construída para a análise de habilidades e competências técnicas necessárias a um time de desenvolvimento de software.

Habilidades e Competências Técnicas		Precisamos ?	Temos ?	Precisamos Melhorar ?
Ferramentas e Linguagens	Habilidade para o desenvolviemnto nas linguagens C, C#, Java, Jquery, JavaScript.	Sim	Sim	Não
	Capacidade de manipulação de SGBD (SQL) e Visual Studio Team System.	Sim	Sim	Sim
Engenharia de Software	Conhecimentos sobre padrões de arquitetura de aplicações corporativas.	Sim	Não	Sim
	Conhecimentos sobre padrões de arquitetura de aplicações web.	Sim	Sim	Não
	Conhecimentos sobre programação orientada a objetos.	Sim	Sim	Não
Operacional	Conhecimentos dos principais processos da empresa (visão do todo)	Sim	Sim	Não
	Conhecimentos da metodologia SCRUM.	Sim	Sim	Sim
Produtos	Visão sobre produtos para internet	Sim	Sim	Sim
	Conhecimentos sobre produtos corporativos em geral.	Sim	Não	Sim
Qualidade	Capacidade para desenvolvimento orientado a teste	Sim	Não	Sim
	Conhecimentos sobre testes unitários	Sim	Sim	Sim
Comunicação	Habilidade para a comunicação eficaz com outras áreas da empresa ou clientes.	Sim	Sim	Não
	Conhecimento de técnicas para apresentações.	Sim	Não	Sim
Linguas	Inglês fluente	Sim	Sim	Sim
	Espanhol	Sim	Não	Sim
Relacionamento	Desenvolvimento de relacionamento intra e extra áreas	Sim	Sim	Não

Figura 7– Habilidades e Competências Técnicas Necessárias ao Time de Desenvolvimento de Software.

A figura 8 apresenta um exemplo de tabela que pode ser construída para a análise de habilidades e competências comportamentais necessárias ao time de desenvolvimento de software.

Habilidades e Competências Comportamentais		Precisamos ?	Temos ?	Precisamos Melhorar?
Relacionamentos	Confiança	Sim	Não	Sim
	Relacionamento Interpessoal	Sim	Não	Sim
	Colaboração	Sim	Sim	Sim
	Compartilhamento de Conhecimento	Sim	Não	Sim
	Facilidade de Comunicação	Sim	Sim	Sim
	Transparência	Sim	Não	Sim
Criatividade	Capacidade Analítica	Sim	Sim	Sim
	Poder de Inovação	Sim	Sim	Sim
	Capacidade de aprendizado	Sim	Sim	Não
Postura Pessoal	Bom humor	Sim	Sim	Não
	Organização	Sim	Não	Sim
	Controle de ansiedade	Sim	Não	Sim
	Auto Motivação	Sim	Não	Sim
	Agilidade	Sim	Não	Sim
	Determinação	Sim	Sim	Sim

Figura 8 – Habilidades e Competências Comportamentais Necessárias ao Time de Desenvolvimento de Software.

As tabelas foram criadas com base nas necessidades de um determinado time de desenvolvimento, uma análise sobre as tabelas acima é importante, pois pode fornecer grande apoio na escolha do novo membro para o time, sendo possível identificar os conhecimentos complementares necessários ao time e

garantindo que ocorra sinergia na equipe possibilitando o desempenho da missão e entrega dos objetivos e metas.

3.1.2 Criando o Ambiente e Avaliando os Recursos Materiais para a Entrega dos Objetivos

O objetivo deste tópico é apresentar a criação do ambiente de trabalho do time assim como avaliar os recursos matérias necessários para a entrega de software.

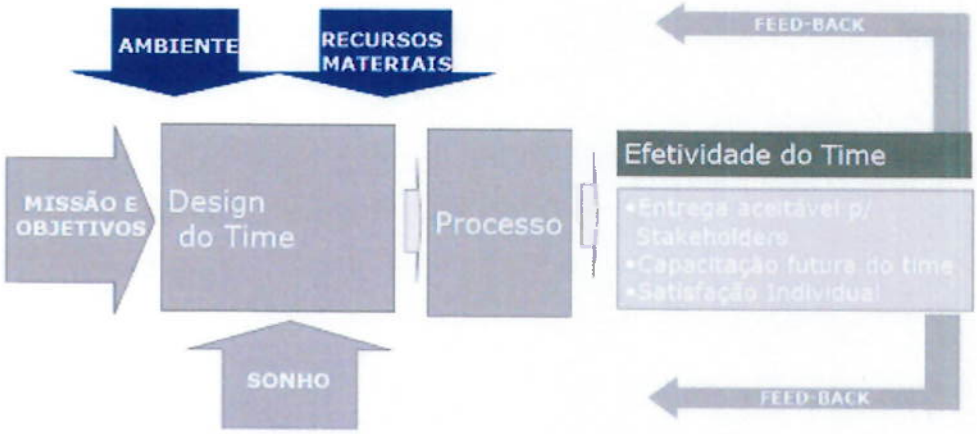

Figura 9- Ambiente e Recursos Materiais Necessários para Entrega dos Objetivos. (Robert Ginnett, 2005)

Times de Alto Desempenho assim como equipes *SCRUM* coordenam seus trabalhos através da consciência das atividades de cada membro da equipe juntamente com a comunicação e partilha de informações, e é na coordenação entre os trabalhos dos membros que equipes gastam a maior parte do tempo (Fuchs, et al., 2001).

Para o desenvolvimento eficaz do trabalho os *Times de Alto Desempenho* necessitam de um ambiente adequado, que forneça o entendimento sobre o progresso das atividades dos colegas e facilite a comunicação entre os membros da equipe (Fuchs, et al., 2001). O ambiente adequado também é importante para o *SCRUM*, pois a estrutura departamental deixa de existir e designers, engenheiros programadores, testadores, entre outros trabalham juntos (Martins, 2007). A união

dos integrantes da equipe deve ser preferencialmente no mesmo espaço físico ou sala de trabalho, promovendo naturalmente uma comunicação muito forte entre os membros do time, que é uma das chaves para o resultado do trabalho eficaz (Fuchs, et al., 2001).

A proposta é garantir um ambiente de trabalho onde os *Times de Alto Desempenho* possam trabalhar de forma eficaz utilizando o ambiente e recursos materiais pertencentes à metodologia *SCRUM*, além de apresentar uma análise sobre os recursos materiais necessários para o desempenho da missão e entrega dos objetivos e metas.

Para Fuchs (2001) o ideal é que as equipes trabalhem fisicamente unidas, porém utilizando a metodologia SCRUM é necessário que além de estarem fisicamente unidas, as equipes tenham acesso visual ao quadro *SCRUM* de gerenciamento de tarefas, aos gráficos de Tendência do Sprint e do Produto, este espaço com o quadro também será utilizado para a realização das Reuniões Diárias por isso deve-se ter algum espaço físico livre para que a equipe possa realizar a reunião em pé de frente ao quadro. Salas de reunião também são necessárias para a realização de outras reuniões como o Planejamento do *Sprint*, Retrospectiva do *Sprint* e Revisão do *Sprint*.

Os recursos necessários ao desenvolvimento tais como: equipamentos, ferramentas, sistemas, espaços físicos e demais investimentos necessários a configuração do ambiente devem ser explicitamente planejados e adicionados ao Backlog do Produto e podem ficar sob responsabilidade do líder do time denominado Mestre Scrum que é o responsável por criar as condições e remover os impedimentos para que o time possa trabalhar.

Em termos gerais o desenvolvimento de software é dependente do acesso a ferramentas, investimento financeiro, equipamentos e sistemas devidamente atualizados, do conhecimento técnico e comportamental da equipe além de um ambiente adequado, caso contrario, se tornam obstáculos para o time impedindo que desenvolvam o trabalho de forma eficaz e performática.

A figura 10 apresenta um exemplo de tabela que pode ser construída para a análise sobre os recursos necessários para desempenhar a missão e entregar os objetivos e metas.

Recursos	Detalhamento	Precisamos	Temos
Cursos	Curso SCRUM, Team System, Hibernate, entre outros.	Sim	Não
Troca de Conhecimento	Ferramentas para gestão de conhecimento, e wiki.	Sim	Sim
Sistemas	MS Visual Studio, Bases OLTP, Bases OLAP, Project, MS Team System, entre outros.	Sim	Sim
Espaço Físico	Adequado que facilite a comunicação, silencioso e capacidade para concentração e foco.	Sim	Não
Equipamentos	Recursos de infra-estrutura de TI como computadores, impressoras servidores e notebooks.	Sim	Sim
Verba	Previamente aprovada pelos gestores.	Sim	Não
Recursos Humanos	Equipe estruturada, qualificada e treinada.	Sim	Sim
Materiais SCRUM	Post-it's, Baralho Scrum, quadro SCRUM, entre outros.	Sim	Sim

Figura 10 – Recursos necessários ao time de desenvolvimento de software para entrega dos objetivos.

O objetivo em ter um ambiente e recursos materiais adequados para o trabalho das equipes é apoiar a coordenação e comunicação do *Time de Alto Desempenho* (Fuchs, et al., 2001), ressaltando também que o ambiente e recursos são de grande importância para o desempenho da missão e entrega dos objetivos por parte dos times de desenvolvimento de software.

3.1.3 Definindo o Papel do Líder e o Modelo de Gerenciamento em um Time Eficaz

O objetivo deste tópico é detalhar o papel do líder que será representado pelo Mestre Scrum assim como detalhar o modelo de gerenciamento a ser utilizado em um *Time de Alto Desempenho*.

Segundo Hunter (2004) liderança é a habilidade de influenciar pessoas para trabalharem entusiasticamente visando atingir aos objetivos identificados como sendo para o bem comum, ele também cita que liderança não deve ser confundida com gerência já que pessoas são lideradas e coisas são gerenciadas. Liderar é conseguir que as coisas sejam feitas através de pessoas e o papel do líder é extremamente exigente, pois as pessoas contam com o apoio do líder (Hunter, 2004).

Existe uma grande tendência em ver liderança como "O líder nato", que nada mais é do que um líder forte, respeitado e às vezes temido pelos

subordinados, entretanto deve-se ver liderança como à sobreposição de três combinações: O líder colaborativo, a situação e os seguidores (Robert Ginnett, 2005). Este conceito de líder nato traz a idéia de que um líder forte pode criar o sucesso independente dos seguidores e situação, e isso pode não ser verdade.

O papel do líder em um time que utiliza a metodologia *SCRUM* é desempenhado pelo Mestre Scrum. A posição do Mestre Scrum é similar a do gerente ou coordenador de projetos e seu papel no gerenciamento dos times é dado a partir da coordenação entre os trabalhos dos membros, juntamente com a partilha das informações, comunicação eficaz, solicitação e aprendizagem a partir de comentários, sempre mantendo o foco nas metas comuns (Martins, 2007).

Segundo Martins (2007) e Machado (2009) o Mestre Scrum deve ser responsável por:

- Garantir que o time siga os valores e práticas da metodologia *SCRUM* como a transparência, trabalho energizado, análise da causa raiz, auto-organização, comunicação eficaz, prática da programação em par e revisão de código, evolução iterativa do produto e entrega de valor a cada iteração.
- Prover os recursos necessários ao longo do projeto de acordo com as necessidades e impedimentos que surgem ao longo do projeto.
- Compartilhar a missão, visão e valores com o time: Em diversas empresas a missão, visão e valores são conhecidos somente pelo nível estratégico da empresa, deixando a equipe sem rumo, sem saber aonde a empresa quer chegar e o que podem fazer pela empresa.
- Inspirar e envolver o time em um objetivo comum.
- Enxergar individualmente os membros do time, levando em consideração os perfis comportamentais.
- Ter perfil habilitador: Dar autonomia aos membros do time para a realização do trabalho e saber ouvir, buscando opinião alheia e sendo aberto a influências.
- Permitir a cada membro da equipe a flexibilidade para exercer diversos papéis e trabalhar em conexão com outras pessoas garantindo maior produtividade e desempenho.

É possível concluir então que um Mestre Scrum em *Times de Alto Desempenho* necessita saber mais sobre times, relacionamentos e influência do que sobre gerência ou liderança (Hunter, 2004). O Mestre Scrum pode ser trocado periodicamente, ou seja, pode ser exercido por diferentes membros no decorrer do projeto, desde que tenha perfil colaborativo saiba servir, dar apoio e criar as condições para o time ser efetivo, trabalhando colaborativamente para o ganho mútuo.

Além do papel diferente do líder na gestão de projetos de software, é importante que os gestores adotem um modelo similar ao modelo Três-S (*Three-S*), onde a organização fornece o sistema e a estrutura, o líder fornece o apoio e a equipe fornece o desenvolvimento, ou seja, o projeto em si (Pham, et al., 2005).

A figura 11 mostra como deve ser a pirâmide de apoio Três-S:

Figura 11– Camadas de gestão para o time de desenvolvimento de software (Pham, et al., 2005).

Pensando no processo de desenvolvimento de software aplicado à pirâmide acima, a diretoria fica com a responsabilidade de fornecer a estrutura necessária para o desenvolvimento de software. O líder, ou seja, o *Scrum Master* fica responsável pelo apoio, motivação entre outras características necessárias a um *Scrum Master* citadas anteriormente. E por fim, o time (composto por desenvolvedores, desenhistas, analistas, engenheiros, entre outros) é o responsável pelo desenvolvimento do projeto de software.

Segundo Pham (2005), esse modelo estabelece uma cultura onde cada membro do time é respeitado e tem a oportunidade de contribuir com seus

conhecimentos e habilidades, além disso, o modelo favorece o comprometimento do time, pois os papeis apóiam uns aos outros e nenhum trabalho é mais importante que outro. Isso faz com os membros se sintam úteis vendo o quão importantes são para a estratégia do time e da corporação.

3.1.4 O Processo de Trabalho do Time de Alto Desempenho e o Papel da Metodologia SCRUM no Trabalho do Time

Com o *Time de Alto Desempenho* já formado a proposta deste tópico é apresentar como o *SCRUM* irá atuar no processo de trabalho do time de desenvolvimento de software.

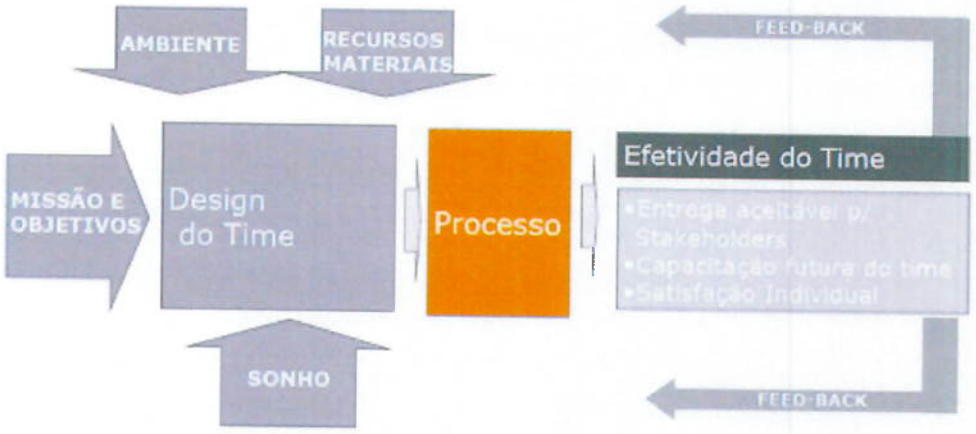

Figura 12 - O Processo de Trabalho do Time de Alto Desempenho (Robert Ginnett, 2005)

Quando uma companhia adota uma metodologia ágil, sua intenção é reduzir a freqüência com que os principais fatores que fazem os projetos de software falhar ocorram. Dados de 1995 (Standish Group, 1995), utilizando uma base com 8.380 projetos revelaram que apenas 16,2% dos projetos foram entregues dentro do prazo, custo e com as funcionalidades conforme a especificação. Aproximadamente 31% dos projetos foram cancelados antes do término e 52,7% foram entregues, porém com custos e prazos maiores, além da não conformidade das funcionalidades com a especificação do inicio do projeto. Mesmo assim, os projetos entregues dentro do prazo e custo correm grande risco de não estar com a

qualidade esperada, pois podem ter sido desenvolvidos com pressão sobre os desenvolvedores.

O *Chaos Report* da (Standish Group, 1995), famoso relatório que mostra os principais desafios da TI no mundo e sucesso nos projetos indica que a entrega de componentes de software antecipada e com regularidade muitas vezes irá aumentar a taxa de sucesso aos projetos, pois os torna mais simples e envolve o usuário antes da entrega final. O *SCRUM* se adéqua a este cenário de entregas menores e parciais, portanto sua utilização é fundamental para que se evite a ocorrência de problemas como os citados anteriormente.

O *SCRUM* é um processo de desenvolvimento iterativo e incremental de gerenciamento de projetos onde cada S*print* é uma iteração que segue o ciclo PDCA, que planeja a funcionalidade ou software (Plan), desenvolve (Do), testa e checa a funcionalidade ou software (Check) e toma uma ação (Action) sobre o ciclo completo, entregando incremento de software ou funcionalidade. O produto pode ser entregue aos clientes de forma incremental, antecipando o lucro do projeto de uma forma controlada (Martins, 2007). A adoção do método ágil *SCRUM* no processo de trabalho tem como objetivo maximizar a produtividade do time para que conclua os projetos com todas as funcionalidades conforme a especificação, dentro do prazo e custo previsto e com qualidade.

Para que o time tenha excelente desempenho ele necessita de um entendimento profundo sobre o processo (Carrasco, 2008) e o *SCRUM* como metodologia de gestão de projetos de software traz uma abordagem simples sobre o processo de desenvolvimento de software.

O processo BPMN na figura 13 mostra o processo completo pelo qual um incremento de software ou uma nova funcionalidade normalmente é submetida.

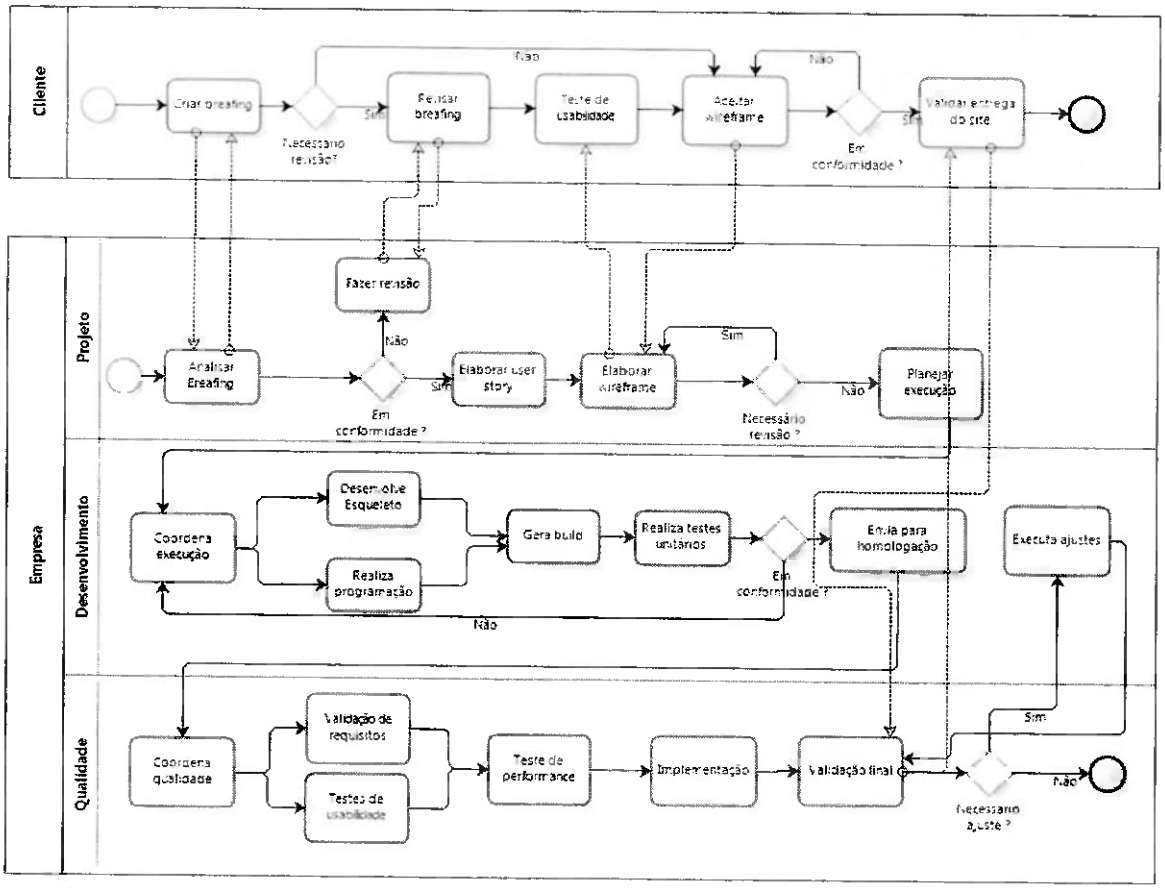

Figura 13 - BPMN do Processo de Desenvolvimento de Software

O processo representado no BPMN acima é uma iteração ou *Sprint* derivada de um *Backlog* de *Sprint*, cada *Sprint* tem duração média de 15 a 30 dias até a entrega do produto, esses dias são acompanhados por reuniões diárias para que todo o time tenha consciência das atividades de cada membro, nesse período a equipe é "deixada sozinha" para trabalhar e atingir os objetivos. Ao final da iteração o time apresenta o incremento de produto ou funcionalidade para o cliente e interessados no negócio e antes de começar uma próxima iteração realizam a reunião de retrospectiva levantando o que "deu certo" e o que "deu errado" no *Sprint* (Martins, 2007).

Além do processo enxuto, através das técnicas ágeis envolvidas no processo de desenvolvimento (presentes no BPMN da figura 13), como a orientação a testes, realização de testes unitários, testes de desempenho, entre outros, o SCRUM possibilita a redução na quantidade de débitos técnicos aumentando a velocidade de desenvolvimento e melhorando a qualidade do

software para que o produto seja entregue no prazo, com qualidade e com o custo estimado na definição inicial do projeto (Standish Group, 1995).

3.1.5 Adaptando os Times e a Mentalidade Ágil ao Modelo Organizacional da Empresa

Diversos autores falam sobre como construir, gerir e aumentar o desempenho em *Times de Alto Desempenho*, porém existe a necessidade de transformar equipes funcionais em *Times de Alto Desempenho*. A proposta deste tópico é examinar os requisitos para transformar grupos funcionais existentes de desenvolvimento de software, presentes em diversos modelos organizacionais, em *Times de Alto Desempenho* trabalhando com a mentalidade ágil.

Empresas de desenvolvimento de software que utilizam metodologias tradicionais de desenvolvimento, como o modelo cascata, normalmente possuem equipes funcionais, ou seja, existe uma divisão funcional onde os desenhistas fazem a programação gráfica, engenheiros projetam o software, programadores programam e testadores testam o software (Martins, 2007), eles trabalham separadamente, o que pode causar muitas vezes a falta de comunicação e alinhamento sobre o produto que está sendo desenvolvido, além de aumentar a possibilidade de o projeto ser falho (Fuchs, et al., 2001).

Segundo Walsh (2000) um grupo funcional bem estruturado pode proporcionar um desempenho sólido e estável além de estabelecer um nível de conforto e segurança para os usuários finais, pois já estão acostumados com a função de cada profissional dentro da empresa. Porém, nos dias de hoje, a procura por novos desafios é grande, todos procuram por novas maneiras de fazer as coisas que podem resultar em um desempenho superior. Existe também uma confusão de conceitos sobre trabalho em equipe, em um sentido muito amplo a sociedade definiu que trabalhar com outra pessoa é trabalho em equipe, porém para que uma equipe exista deve haver um conjunto de objetivos e metas comuns levando em consideração os objetivos e metas da corporação como um todo.

Uma importante mudança para que um grupo funcional se torne um *Time de Alto Desempenho* é na maneira como o time é medido, segundo Walsh (2000) o

time deve ser medido por etapas, onde os membros têm habilidades complementares e são responsáveis pelo resultado. A metodologia ágil *SCRUM* fornece todo o apoio para este tipo de medição, pois o time é medido por *Sprint*, ou seja, por entregas parciais de software funcional feitas a cada 15 ou 30 dias.

Depois de criado, não se pode deixar que o time perca o entusiasmo e caia de rendimento, isso fará com que o time tenha aparência de um grupo funcional, a idéia é tentar fazer com que os funcionários sejam atraídos a participar do time e que também sejam reconhecidos para que sintam uma sensação de realização, motivando-os a querer entrar em um novo time para um novo projeto (Walsh, 2000).

Porém, Segundo Walsh (2000) e Machado (2009) a adoção de *Times de Alto Desempenho* assim como a adoção da metodologia *SCRUM* deve ser bem analisadas, existem casos onde grupos funcionais podem ser necessários, como por exemplo, em grupos que atuam com tarefas de rotina, ou seja, operações que são repetidas diariamente, neste caso grupos funcionais proporcionam um desempenho satisfatório, pois possuem uma estrutura menos complicada e mais fácil de gerenciar. Outra saída pode ser a adoção de um modelo híbrido na gestão de projetos de desenvolvimento de software.

3.1.6 A Gestão do Conhecimento em Times de Alto Desempenho que Utilizam a Metodologia SCRUM de Gestão de Projetos

O cenário mundial indica que as empresas estão passando por uma grande e significativa mudança, elas estão passando da Era Industrial para a Era do Conhecimento e de acordo com Katzenbach e Smith (2001) as equipes são fundamentais para a gestão do conhecimento e obtenção do desempenho esperado em uma empresa do futuro. Essa nova cultura organizacional visa melhorar o acesso da informação em uma empresa, promovendo espaço para que o conhecimento seja compartilhado assim tornando as equipes as ferramentas para a execução do conhecimento através da troca de idéias e trabalho em equipe (Katzenbach e Smith, 2001).

A proposta deste tópico é apresentar a introdução da gestão do conhecimento em *Times de Alto Desempenho* que trabalham com a metodologia ágil *SCRUM* no desenvolvimento de software, através de mudanças de atitudes na empresa, valorização do capital intelectual, além de apresentar algumas opções de ferramentas e tecnologias de apoio na gestão do conhecimento que também auxiliam no gerenciamento de tarefas.

Metodologias ágeis ao contrário do modelo cascata de desenvolvimento são compostas por poucos artefatos e documentações extensas (Martins, 2007), o que pode tornar a gestão do conhecimento algo mais difícil e menos detalhado. Porém, é possível documentar em quando se utiliza métodos ágeis de desenvolvimento, a reunião de Retrospectiva (*Retrospective*) na metodologia *SCRUM,* por exemplo, pode ser utilizada para documentação dentro do *Sprint*, são nestas reuniões onde se devem registrar as lições aprendidas, o que deu certo e o que pode ser melhorado no *Sprint* para que sejam facilmente encontradas em uma próxima vez quando necessário (Martins, 2007).

Em uma equipe de desenvolvimento de software ágil um simples comentário no código de um sistema ajuda a entender como o mesmo funciona, fazendo com que uma alteração no sistema ou implementação de uma nova funcionalidade não exija a leitura de todo o código por parte do desenvolvedor para que o mesmo entenda o software.

Outra forma é a documentação através um sistema wiki, estes sistemas são sistemas colaborativos, normalmente web, que permitem a edição coletiva de documentos sem a necessidade de aprovação do conteúdo para que seja publicado, o que faz o wiki ser diferente de outras páginas web é o fato das páginas estarem abertas para edição por parte dos usuários que estão navegando nelas, permitindo assim a correção de erros em textos nas páginas, a inclusão de novos textos assim como a complementação de idéias que já estão publicadas. Dentro de um *Sprint* pode ser reservado um tempo para a documentação no wiki, pode ser adicionada uma Estória do Usuário (*User Store*) específica para a documentação de cada tarefa desenvolvida no wiki, ou até mesmo podem ser criadas tarefas adicionais para isso, por exemplo, se existir uma tarefa "Desenvolver Busca Simples" a mesma pode ter uma tarefa relacionada "Wiki

Busca Simples", onde estará documentado como a busca simples foi desenvolvida pelo programador.

A utilização das ferramentas e tecnologias de apoio a gestão do conhecimento citadas acima podem ser adotadas de forma simples em projetos ágeis, pois não trazem muito impacto para equipe onde o foco não é a documentação (Katzenbach e Smith, 2001). Essas ferramentas estarão auxiliando na seleção, organização e transferência do conhecimento dentro da empresa de forma que possa ser compartilhado e reutilizado.

3.1.7 Utilização do Modelo TELM no Diagnóstico de Problemas em Times de Alto Desempenho

Um dos objetivos na adoção de *Times de Alto Desempenho* é atingir a sinergia, ou seja, o que se espera é que os talentos individuais se somem de forma que o resultado seja maior do que a simples soma aritmética de dois indivíduos. Porém a formação desta equipe não é tão simples e equipes podem desempenhar aquém do esperado (Bejarano, et al., 2006).

O objetivo deste tópico é apresentar a aplicação do Modelo *TELM* no diagnóstico de problemas em times de desenvolvimento de software que não estão desempenhando um bom trabalho.

Segundo Robert Ginnett (2005) criador do Modelo *TELM,* existem três saídas que são essências para o time ser efetivo, uma delas são entregas e resultados aceitáveis aos interessados no negócio, a segunda é a capacidade futura do time e por fim a satisfação individual do membro do time. Ginnett desenvolveu um modelo para diagnosticar problemas de desempenho em times no qual o objetivo é encontrar as soluções adequadas para os problemas. O modelo será então utilizado para diagnosticar problemas em times que atuam com desenvolvimento de software.

A figura 14 mostra a representação do modelo.

Quando a medição da eficácia indicar problema com:	Olhar para estas entradas:		
	Entradas Organizacionais (O)	Design do Time (T)	Entradas Individuais (I)
Esforço (P-1)	Sistemas de Recompensa (O-1)	Tarefas (T-1)	Interesses/Motivação (I-1)
Conhecimentos e Habilidades (P-2)	Sistemas de Educação (O-2)	Design da composição (T-2)	Habilidades/Competências (I-2)
Estratégia (P-3)	Sistemas de Informação (O-3)	Normas (T-3)	Valores/Atitudes (I-3)
	Todos os itens acima são construídos sobre alicerces de:		
Dinâmica de Grupo (P-4)	Sistemas de Controle (O-4)	Autoridade (T-4)	Comportamentos Interpessoais (I-4)

Figura 14 - Modelo TELM para o Diagnóstico de Problemas (Robert Ginnett, 2005)

O modelo diz basicamente que:

- Se o problema for relacionado ao esforço deve-se examinar o sistema de recompensas da organização, assim como as tarefas do time e interesses e motivações individuais.
- Caso o problema esteja em conhecimentos e habilidades considerar o sistema de educação da organização, assim como a composição do time além das capacidades e competências individuais.
- Caso o problema esteja relacionado à estratégia devem-se considerar os sistemas de informação da organização, assim como normas do time e valores e atitudes individuais.
- E por fim caso o problema esteja relacionado à dinâmica de grupo deve-se examinar os sistemas de controle da organização, a autoridade dos times, assim como o comportamento interpessoal dos indivíduos.

O modelo aplicado ao desenvolvimento de software onde as entregas apontam problemas relacionados ao esforço, por exemplo, pode ser:

- O-1: Recompensa para metas e objetivos atingidos pela equipe, utilização da meritocracia na valorização dos funcionários, criação de sistema de bônus para premiação dos mesmos.
- T-1: Avaliação sobre as tarefas que estão sendo desempenhas pelo time, criar balanceamento entre tarefas de operação e rotineiras no

desenvolvimento de software e tarefas inovadoras que fornecem motivação aos membros do time.

- I-1: Avaliar quais os interesses individuais de cada membro do time tentando motivá-lo através da criação de tarefas de seu interesse, dar ao funcionário autoridade para a tomada de decisões.

Quando as entregas apontam problemas relacionados à estratégia, por exemplo, pode ser:

- O-3: Disponibilidade de dados para a equipe realizar o trabalho. Adotar sistemas para gestão do conhecimento e facilitação na comunicação entre os membros da equipe, assim como sistemas de histórico de projetos.
- T-3: Fazer com que todos os membros do time sigam as regras e práticas do *SCRUM.*
- I-3: São realidades que devem ser tratadas no processo seletivo, caso os valores e atitudes do individuo não estejam em conformidade com os da organização a correção pode ser a rescisão de contrato.

Nota: A combinação de letras e números (ex: O-1, T-2) indicam a localização dos componentes no modelo TELM da figura 13.

Os exemplos acima foram dados levando em consideração problemas relacionados a esforço e estratégia, para o diagnóstico completo dos problemas no time deve-se aplicar o modelo completo com Esforço (P-1), Conhecimentos e Habilidades (P-2), Estratégia (P-3) e Dinâmica de Grupo (P-4).

4 CONCLUSÕES

Este trabalho contribui apresentando a adoção de *Times de Alto Desempenho* atuando na solução de pontos fracos da metodologia ágil *SCRUM*, já que somente a adoção do *SCRUM* não garante o desempenho máximo do time e entregas dentro do escopo, prazo e custo.

A adoção de *Times de Alto Desempenho* correlacionada à metodologia *SCRUM* no desenvolvimento de software levam as organizações ao desempenho máximo, além de aumentar a velocidade no desenvolvimento, ajudar na entregar o produto conforme o especificado e dentro do custo estimado. Entretanto esses times podem levar vários anos para se desenvolverem, tornando necessário um grande empenho por parte da organização no desenvolvimento desses times, ou seja, a promoção do sistema e da estrutura necessária para o desenvolvimento dos times.

O trabalho mostra que *Times de Alto Desempenho,* através da atenção especial que dá a seleção dos novos membros, podem suprir alguns pré-requisitos necessários a uma equipe *SCRUM*, como a presença de membros com competências complementares dentro de cada time.

Através de um ambiente com recursos adequados para o trabalho dos times é possível garantir o apoio a coordenação e comunicação do time, uma vez que um ambiente *SCRUM* fornece a consciência das atividades de cada membro da equipe através da utilização do quadro *SCRUM* e realização de reuniões diárias. Em termos gerais o desenvolvimento de software depende de um ambiente e recursos materiais para desempenhar a missão e entregar os objetivos e metas.

Com a adoção de *Times de Alto Desempenho* e da metodologia *SCRUM* é necessária uma mudança na visão sobre o papel do líder, o líder não deve ser uma pessoa autoritária e que está ali para mandar e sim um líder colaborativo, que crie oportunidades para que os membros se motivem e dê autonomia aos membros do time para realização do trabalho e tomada de decisão, ou seja, um líder que conheça mais sobre relacionamentos e influência do que sobre gerência.

Também foi possível chegar à conclusão que nem sempre a adoção de *Times de Alto Desempenho* e de metodologias ágeis é recomendada, um exemplo é onde o trabalho a ser desempenhado é um trabalho de operação e rotineiro,

nesses casos grupos funcionais bem estruturados podem fornecer um desempenho sólido e estável, pois possuem uma estrutura menos complicada que *Times de Alto Desempenho*.

A documentação é o principal elemento de gestão do conhecimento no desenvolvimento de software, ela promove espaço para que o conhecimento seja compartilhado entre todos os integrantes do time. Entretanto times que trabalham com metodologias ágeis não produzem muitos artefatos e documentações extensas, sendo assim, um comentário no código ou uma simples explicação do que foi desenvolvido e como foi desenvolvido em um sistema wiki.

Por fim foi possível concluir que a aplicação do modelo TELM em *Times de Alto Desempenho* que trabalham com o desenvolvimento de software facilita a identificação de onde está o problema, para que seja endereçado mais facilmente.

4.1 Trabalhos Futuros

Este trabalho não explorou a medição dos *Times de Alto Desempenho* que é feita através dos resultados e pode ser um ponto de continuidade do trabalho. A medição é baseada em questões que medem a efetividade do time e é capaz de avaliar as entregas, a capacidade futura do time, satisfação pessoal dos membros, entre outros.

REFERÊNCIAS

BENJARANO, V. C.; PILATTI, L. A.; LIMA, I. A.; OLIVEIRA, A. C. **Equipes de Alta Performance.** 2006.

DAVIS, C. R. **High-Performance Individuals and Teams: A Study of the Effect of Duration on Achievable and Sustainable Maximum Performance Levels.** IBM, 2007.

FUCHS, L.; POLTROCK, S.; WETZE, I. **Teamspace: An Environment for Team Articulation Work and Virtual Meetings.** Boeing Co., Seattle, WA, 2001.

GINNETT, ROBERT **Leading a Great Team: Building Them From the Ground Up, Fixing Them on the Fly.** Abril, 2005.

GUILLARD, STEFAN, **Ominlab Media's Stefan Gillard on how to build a high performance team,** 2009. Disponível em: <http://www.cio.com.au/article/303748/ominlab_media_stefan_gillard_how_build_high_performance_team/>. Acesso em: 19 out. 2010.

HUNTER, JAMES C. **O monge e o Executivo: Uma História sobre a Essência da Liderança.** Editora Sextante, 2004.

KATZENBACH, J. R.; SMITH, D. K. **Equipes de Alta Performance: conceitos, princípios e técnicas para potencializar o desempenho das equipes.** Editora Campus Ltda., 2001.

PHAM, C. H.; HOUSMAND, K.; PALLA, M.; HU, B.; ESMAILI, R. **Key Foundation to Successfully Build and Manage High Performance Teams in Large Company.** Cisco Systems, Inc., San Jose, CA, USA, 2005.

MACHADO, M.; MEDINA S. G. **SCRUM – Método Ágil: uma mudança cultural na Gestão de Projetos de Desenvolvimento de Software.**

MARTINS, JOSÉ C. C. **Técnicas para Gerenciamento de Projetos de Software.** Brasport, 2007.

SHORE, J.; WARDEN, S. **The Art of Agile Development.** O' Reilly Media, 2008.

SOMMERVILLE, IAN. **Engenharia de Software.** Pearson Education, 2007.

STANDISH GROUP, **CHAOS report**, 1995. Disponível em: <http://www.projectsmart.co.uk/docs/chaos-report.pdf>. Acesso em: 12 nov. 2010.

WALSH, KEN P. **Building Bridges Between Functional Groups and High Performance Teams.** IEEE Member, 2000.