UNIVERSIDADE SÃO JUDAS TADEU
GERÊNCIA DE PROJETOS COM ÊNFASE NAS PRÁTICAS DO PMI

PATRÍCIA FONTOURA DE SENE

GERENCIAMENTO ÁGIL DE PROJETOS

SÃO PAULO

2010

PATRÍCIA FONTOURA DE SENE

GERENCIAMENTO ÁGIL DE PROJETOS

Trabalho de Conclusão de Curso apresentado ao Curso de Pós-Graduação Gerência de Projetos com Ênfase nas Práticas do PMI como requisito parcial para a aprovação. Universidade São Judas Tadeu.

Orientador (a): Prof.(ª) Nelson J. Rosamilha, PMP

SÃO PAULO

2010

DEDICATÓRIA

À Isa Karasek, que sempre se orgulhou das minhas pequenas e grandes vitórias.

AGRADECIMENTOS

Agradeço à minha amiga Gitânia que, mesmo sem saber, me estimulou a dar o primeiro passo nesta caminhada.

EPÍGRAFE

"Suba o Primeiro degrau com fé. Não é necessário que você veja toda a escada. Apenas de o primeiro passo." Martin Luther King.

RESUMO

Os Métodos Ágeis surgiram quando os Métodos Tradicionais de desenvolvimento de software começaram a ser questionados devido, principalmente, ao grande volume de documentação. Eles ganharam mais atenção com a publicação de um manifesto em 2001 e após com o surgimento do Gerenciamento Ágil de Projetos, que representa um conjunto de valores, princípios e práticas, que auxiliam a equipe de projeto a entregar produtos de valor em ambientes desafiadores. Com a elaboração deste estudo concluiu-se que as metodologias ágeis buscam o estabelecimento de novos paradigmas para o desenvolvimento de software, onde o desenvolvimento ágil de software é o foco principal dessas metodologias. As metodologias ágeis visam à simplicidade na implementação de um projeto, pois suas regras são simples de serem seguidas. O sucesso na adoção de um modelo ágil está diretamente relacionado ao comprometimento do time, pois sua forma de condução dos trabalhos possui um elevado grau de informalidade.

PALAVRAS-CHAVE: Metodologias Ágeis; Gerenciamento de Projetos; Desenvolvimento Ágil de Software.

ABSTRACT

Agile methods have emerged when the traditional methods of software development began to be questioned, mainly due to the large volume of documentation. They gained more attention with the publication of a Agile Alliance in 2001 and after the emergence of Agile Project Management, which represents a set of values, principles and practices that help the project team to deliver value products in challenging environments. With the preparation of this study concluded that agile methodologies seek to establish new paradigms for software development, where the agile software development is the main focus of these methodologies. The agile methodologies aim at simplicity in the implementation of a project, its principles are simple rules to follow. The successful adoption of an agile model is directly related to the involvement of the team, because their way of conducting the work has a high degree of informality.

KEYWORDS: Agile Methodologies, Project Management, Agile Software Development.

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

O Gerenciamento de Projetos de Software vem apresentando uma busca incessante por um método científico que proporcione resultados produtivos e qualitativos, e que possa ser utilizado similarmente a uma receita de bolo, independentemente de cultura, tecnologia e tipo de negócios. Por trás desta busca existem os anseios mais íntimos e importantes das empresas: fazer projetos de software com lucro, com a plenitude da satisfação do cliente, com a fidelização do cliente e com crescimento técnico profissional.

A cada ano que passa a importância do software e, consequentemente, sua complexidade vem crescendo significativamente. Processos orientados a documentação podem ser fatores limitadores ao desenvolvimento de software, além de muitas organizações não possuírem recursos ou inclinação para processos pesados de produção de software. Por esta razão, as organizações pequenas acabam por não usar nenhum processo, o que pode levar a efeitos desastrosos na qualidade do produto final, além de dificultar a entrega do software nos prazos e custos predefinidos.

Neste contexto, as metodologias ágeis têm sido apontadas como uma alternativa às abordagens tradicionais para o desenvolvimento de software. Trata-se de uma abordagem de gerenciamento de projetos geralmente empregada no desenvolvimento de software. A metodologia ágil auxilia as equipes a responder aos imprevistos durante a construção de softwares por meio de um trabalho interativo, denominado *sprints*.

Os métodos ágeis surgiram como oposição ao desempenho insatisfatório dos projetos de desenvolvimento de software conduzidos com uso de métodos tradicionais de desenvolvimento e gerenciados de acordo com práticas e princípios do gerenciamento clássico de projetos. Tiveram seu desenvolvimento focado, principalmente, em projetos onde há dificuldades no estabelecimento de requisitos, ou onde os requisitos mudam de forma dinâmica no decorrer do projeto.

Iniciado em princípio num âmbito técnico, com a instauração de novos modelos e métodos de desenvolvimento de software, passou, posteriormente, ao âmbito gerencial, dando-se início ao Gerenciamento Ágil de Projetos. E considerando o ritmo acelerado das mudanças nos projetos de desenvolvimento de software, existe, no mercado, uma forte tendência para o desenvolvimento e gerenciamento ágil das aplicações.

Desenvolvimento ágil é um termo usado para agrupar metodologias interativas e incrementais de desenvolvimento de software, tais como *Extreme Programming* (XP), *Scrum*, *Crystal*, *Dynamic Systems Development Method* (DSDM), *Lean Development*, e *Feature Driven Development* (FDD). Especialmente durante os últimos anos, o desenvolvimento ágil consolidou sua presença em empresas como a Siemens, CapitalOne, Lockheed Martin, Motorola, Microsoft, Yahoo, Google, GE e Cisco Systems. Essas e muitas outras empresas adotaram processos ágeis para entregar melhorias significativas em matéria de inovação, qualidade, produtividade e vantagem competitiva.

Este trabalho explora teorias e abordagens alternativas para o gerenciamento de projetos de desenvolvimento de software. Esta exploração é feita avaliando-se abordagens que foram consideradas influenciadoras nos resultados positivos de projetos, como é o caso dos Métodos Ágeis. É apresentada uma visão geral das metodologias ágeis e do gerenciamento ágil de projetos de software, partindo inicialmente da metodologia clássica de desenvolvimento e de gerenciamento de projetos.

1.1 OBJETIVO

O objetivo principal deste trabalho é comprovar os benefícios do Gerenciamento Ágil de Projetos de Desenvolvimento Software, através a avaliação das metodologias ágeis.

Para alcançar o objetivo acima explicitado, faz-se necessário:

- Estudar as metodologias para desenvolvimento de software;
- Estudar os métodos e técnicas ágeis de desenvolvimento de software;
- Construir resultados e quadros comparativos entre os métodos de desenvolvimento estudados;
- Entender as características dos projetos de software gerenciados com métodos ágeis;
- Formular a conclusão deste trabalho.

1.2 JUSTIFICATIVA

Atualmente, é comum encontrar software de baixa qualidade e projetos que excedem as previsões de custos e tempo de desenvolvimento, ou ainda, sistemas que não atendem a todas as exigências esperadas. Desse modo, os processos de desenvolvimento de software e de gerenciamento de projetos desta natureza, têm ganhado maior importância como forma de prover software de qualidade e cumprir corretamente os prazos de desenvolvimento.

Neste sentido, o presente trabalho busca fazer um estudo dentre os vários modelos de processos de desenvolvimento de software, dando-se uma maior ênfase aos modelos classificados como ágeis, e ao gerenciamento ágil de projetos.

As metodologias ágeis de desenvolvimento de software se propõem a construir softwares com maior produtividade e, sobretudo, com qualidade garantida. Para isso elas encaram os projetos sobre um novo paradigma e defendem a adoção de uma série de princípios e práticas. Entre as quebras de paradigma está a forma de encarar a mudança. Ela passa a ser encarada como algo inevitável no projeto de software, pois enquanto que as metodologias tradicionais abordam o escopo como fixo, devendo ser controlado o máximo possível para não haver mudanças; as metodologias ágeis abordam o escopo como manipulável e oferecem flexibilidade para sua mudança.

1.3 VIABILIDADE

Para o desenvolvimento desta pesquisa faz-se necessário um levantamento e estudo antecipado da bibliografia disponível sobre o tema. Como o assunto é ainda recente, alguns tópicos abordados possuem escassa bibliografia editada, sendo necessário recorrer à pesquisa de outros artigos e trabalhos escritos no meio acadêmico. Sendo assim, a Internet constitui recurso de pesquisa importante para o desenvolvimento deste trabalho.

1.4 METODOLOGIA

A metodologia utilizada neste trabalho constitui um estudo bibliográfico, seguido de uma fundamentação teórica sobre os diversos tipos de abordagens de desenvolvimento de software e gerenciamento de projetos desta natureza. O conteúdo do estudo apresenta entre as metodologias de desenvolvimento e suas características, e permitem a analise das principais vantagens e desvantagens dos métodos ágeis de desenvolvimento bem como seu impacto na gerência dos projetos. O trabalho é concluído com a exposição dos benefícios de se gerenciar projetos de software com gerenciamento ágil.

2 GERENCIAMENTO CLÁSSICO DE PROJETOS

Um projeto pode ser definido como um empreendimento organizado para alcançar um objetivo específico. Segundo Davis, Aquilano e Chase (2001), tecnicamente, um projeto é definido como uma série de atividades ou tarefas relacionadas que são, geralmente, direcionadas para uma saída principal e que necessitam um período de tempo significativo para sua realização.

Vargas (2009, p. 6) apresenta duas citações sobre o que vem a ser um projeto:

> Para Cleland, um projeto é uma combinação de recursos organizacionais, colocados juntos para criarem ou desenvolverem algo que não existia previamente, de modo a prover um aperfeiçoamento da capacidade de desempenho no planejamento e na realização de estratégias organizacionais.
>
> Já para Meredith, um projeto é uma atividade única e exclusiva com um conjunto de resultados desejáveis em seu término. É também complexo o suficiente para necessitar de uma capacidade de coordenação específica e um controle detalhado de prazos, relacionamentos, custos e desempenho.

2.1 O QUE É O PMI?

Em 1969 um grupo de cinco profissionais de gestão de projetos da Philadelphia e Pensilvânia reuniram-se, nos EUA, para discutir melhores práticas de gerenciamento de projetos. Assim, Jim Snyder fundou o PMI - *Project Management Institute*, uma entidade internacional e sem fins lucrativos, que congrega profissionais que atuam em áreas relacionadas à gestão de projetos.

A premissa do PMI é que as ferramentas e as técnicas da gerência de projeto são terra comum desde a aplicação difundida em projetos da indústria do software até indústria de construção. Em 1981, os diretores do PMI autorizaram o desenvolvimento do Guia PMBOK® - *Project Management Body of Knowledge*, que

se transformou em um guia de projetos contendo os padrões e as linhas mestras das práticas utilizadas para o gerenciamento de projetos (PMBOK, 2004).

2.2 O QUE É UM PROJETO?

Segundo a definição do PMI, apresentada na 3ª edição do Guia P PMBOK® (2004, p. 5), "Um projeto é um esforço temporário empreendido para criar um produto, serviço ou resultado exclusivo".

Sendo assim, um projeto é uma iniciativa que é única de alguma forma:

- Seja no produto que gera;
- Seja no cliente do projeto;
- Seja na localização;
- Seja nas pessoas envolvidas;
- Seja em outro fator.

Um projeto possui um início e um fim bem definido, o que dá a característica de temporário. Em outras palavras, possui um objetivo claro, que quando atingido, caracteriza o final do projeto (PMBOK, 2004).

Os atributos de um projeto, segundo os autores Mattos e Guimarães (2005; p. 132), são:

- Propósito único;
- Temporário;
- Necessidade de recursos, frequentemente de várias áreas;
- Necessidade de financiador e cliente;
- Envolve incertezas, decorrentes de premissas assumidas como verdadeiras no início do projeto e que podem não ocorrer. São usualmente consideradas na análise de riscos.

2.3 O QUE É A GESTÃO DE PROJETOS?

A Gerência de Projetos surgiu como um processo de gerenciamento para lidar com a complexidade do trabalho em grupo baseado em conhecimento e pelas demandas de novos métodos de gerenciamento. Nos Estados Unidos, o pai da gerência de projetos é Henry Gantt. Ele é chamado o pai das técnicas de planejamento e de controle e é conhecido pelo uso do gráfico de barra como uma ferramenta de gestão de projeto. O trabalho de Henry Gantt é precursor de muitas ferramentas da gestão de projetos, tais como a WBS - *Work Breakdown Structure* ou EAP - Estrutura Analítica do Projeto (WIKIPEDIA, 2010).

Um projeto está sujeito a uma tripla limitação estabelecida nas metas de escopo, de prazo e de custos. Para que as atividades de um projeto sejam realizadas e seus objetivos sejam atingidos, os projetos demandam de esforço de gerenciamento. Como uma disciplina, a gerência de projetos foi desenvolvida de diversos campos de aplicação que vão desde a construção civil e a engenharia mecânica até projetos militares (MATTOS, GUIMARÃES, 2005).

Portanto, a Gerência de Projetos (ou Gestão de Projetos) é a aplicação de conhecimentos, habilidades e técnicas para projetar atividades que visem atingir ou exceder as expectativas das partes envolvidas (MATTOS, GUIMARÃES, 2005).

Gerenciar, administrar, coordenar ou gerir um projeto, envolve desde iniciá-lo até finalizá-lo, passando pelas etapas de planejamento, execução e atividades de controle. A Gestão de Projetos envolve um processo de estabelecimento de um plano que será implementado para atingir os objetivos propostos. Em síntese, o gerenciamento de projetos consiste primeiramente no planejamento do trabalho e depois na execução do plano (GIDO, CLEMENTS, 2007).

O gerenciamento de projetos pode ser entendido como o planejamento, a direção e o controle de recursos (pessoas, equipamentos e materiais) para atender às restrições técnicas, de custos, e de tempo do projeto.

O quadro abaixo apresenta os principais mitos do gerenciamento de projetos e os conceitos revisados por Kerzner (1998).

Quadro 1: Transição de conceitos de gerenciamento de projetos

MITOS	**CONCEITOS REVISADOS**
Gerenciamento de projetos requer mais pessoas e adiciona custos indiretos à empresa.	Gerenciamento de projetos permite ao projeto realizar mais trabalho em menos tempo com menos pessoas.
A lucratividade pode diminuir em decorrência dos custos de controle.	A lucratividade irá aumentar devido à presença de controle.
O gerenciamento de projetos aumenta o número de mudanças no escopo.	O gerenciamento de projetos permite maior controle sobre as mudanças de escopo.
O gerenciamento de projetos cria instabilidade organizacional e aumenta os conflitos entre departamentos.	O gerenciamento de projetos torna a organização mais eficiente e melhor efetivamente a relação entre os setores por meio do trabalho em equipe.
O gerenciamento de projetos cria problemas.	O gerenciamento de projetos possibilita a solução de problemas.
Somente grandes projetos necessitam de gerenciamento.	Todos os projetos se beneficiam diretamente do gerenciamento de projetos.
O gerenciamento de projetos cria problemas de poder e autoridade.	O gerenciamento de projetos reduz conflitos por poder.
O gerenciamento de projetos tem como objetivos os produtos.	O gerenciamento de projetos tem como objetivo as soluções.
O custo do gerenciamento de projetos pode tornar a companhia menos competitiva.	O gerenciamento de projetos aprimora os negócios da empresa.

Fonte: Kerzner, 1998.

Germano (2008) descreve nove pecados mortais no planejamento de projetos:

- Sem planejamento algum: o problema mais comum é não ter planejamento algum, mas isto pode ser facilmente evitado, pois uma pessoa não precisa ser um especialista na área para criar um planejamento eficiente. Mas, o planejamento é essencial.
- Falha ao contabilizar todas as atividades de projeto: ou seja, não planejar o suficiente. Alguns planos de projetos são criados assumindo que ninguém no time de desenvolvimento ficará doente, terá treinamentos, férias ou sairá da empresa. As atividades principais são geralmente subestimadas em grande nível. Planejamentos criados assumindo condições irreais como estas levarão projetos ao desastre.
- Falha ao planejar por riscos: para projetos de software, esquivar-se constantemente de falhas é tão importante quanto simular o sucesso. Em muitos contextos, a palavra ‘risco’ não é mencionada a menos que o projeto

já esteja em grandes problemas. Em software, o gestor do projeto que não utiliza a palavra ‘risco’ todos os dias e não incorpora o gerenciamento de riscos em seu plano provavelmente não está realizando seu trabalho.

- Utilizar o mesmo plano para todos os projetos: algumas empresas crescem envoltas em uma forma particular de conduzir os projetos de software, que é conhecida como ‘o jeito que fazemos as coisas por aqui’. Quando esta abordagem é utilizada, a empresa tende a ir bem enquanto o projeto se parecer com os antigos projetos.
- Aplicar indiscriminadamente planejamentos pré-fabricados: os planos pré-fabricados podem ajudar a evitar problemas, mas não são substitutos para a tentativa de otimizar o planejamento de um projeto para suas principais necessidades. Nenhum especialista de fora pode compreender as necessidades específicas de um projeto tão bem quanto às pessoas diretamente envolvidas nele.
- Permitir que um planejamento fuja à realidade do projeto: uma abordagem comum para o planejamento é criar um plano no início do projeto e então desprezá-lo, não seguindo o que foi planejado. À medida que as condições do projeto mudam, o plano torna-se invariavelmente irrelevante, e, ao chegar à metade, o projeto correrá livre de formas, sem nenhuma relação real entre o planejamento inicial e a realidade do projeto. O planejamento do projeto deve evoluir através do projeto.
- Planejar em muitos detalhes muito cedo: alguns gestores bem intencionados tentam mapear todo o valor das atividades de um projeto muito cedo. Mas um projeto de software consiste em um grupo de decisões constantemente desconhecido, e cada fase de um projeto cria dependências para as futuras decisões. Deste modo, tentar planejar atividades distantes em muitos detalhes é um exercício burocrático e prejudica o planejamento.
- Planejando para alcançar no futuro: para projetos que trabalham em atraso, um erro comum é planejar para recuperar o tempo perdido no futuro. Mas, à medida que a equipe progride para as fases mais adiantadas do projeto, a velocidade não aumenta; ela irá diminuir à media que encontrar as consequencias dos erros cometidos mais cedo e investirá tempo tentando corrigir tais erros.

- Não aprender com os erros de planejamentos antigos: projetos de software podem durar muito tempo, no final de um longo projeto, pode ser difícil recordar todas as decisões anteriores que afetaram sua conclusão. Uma forma fácil de conter esta tendência e prevenir futuros pecados mortais é conduzir uma retrospectiva *post mortem*. Tal processo estruturado pode não apagar os erros passados de um projeto, mas pode certamente ajudar a prevenir os erros de projetos futuros.

2.4 O GERENCIAMENTO DE PROJETOS SEGUNDO O PMI

A proposta de gestão de projetos através da abordagem tradicional de gerenciamento de projetos, estruturada segundo uma visão de processos conforme proposto pelo PMI, é bastante aceita dentro da gerência tradicional de projetos. A definição de gerenciamento de projetos está baseada na gestão funcional, com a estruturação do gerenciamento por meio de processos, encarando o projeto como uma sequencia de atividades linearmente distribuídas. Esta definição trata do ciclo de vida do projeto da maneira pela qual ele é visto pelas principais metodologias de desenvolvimento, com suas fases bem definidas.

O PMI descreve, no Guia PMBOK® (2004, p. 8), o gerenciamento de projetos como "[...] a aplicação de conhecimentos, habilidades, ferramentas e técnicas às atividades do projeto, a fim de atender aos seus requisitos". O PMI destaca ainda que o "gerenciamento de projetos é realizado através da aplicação e da integração dos seguintes processos de gerenciamento de projeto: iniciação, planejamento, execução, monitoramento e controle e encerramento" (PMBOK , 2004, p. 8).

Ainda segundo o PMI, gerenciar um projeto inclui identificar as necessidades, estabelecer objetivos claros e alcançáveis, balancear demandas conflitantes de qualidade, escopo, tempo e custo, adaptar especificações, planos e abordagens às diferentes preocupações e expectativas das diversas partes interessadas (PMBOK, 2004).

Na visão do PMI (2004, p. 37-38) "o gerenciamento de projetos é realizado através de processos, usando conhecimento, habilidades, ferramentas e técnicas do gerenciamento de projetos que recebem entradas e geram saídas". Neste caso, Processo é definido como "um conjunto de ações e atividades inter-relacionadas realizadas para obter um conjunto pré-especificado de produtos, resultados ou serviços".

Existem duas categorias de processos em um projeto (PMBOK, 2004):

- Processos de gerenciamento de projetos: comuns a maioria dos projetos, visando um objetivo integrado: iniciar, planejar, executar, monitorar e controlar e encerrar um projeto.
- Processos de orientados ao produto: especificam e criam o produto do projeto.

Os processos de gerenciamento de projetos estão agrupados em 5 grupos, de acordo com o PMI (2004, p. 41), que os descreve como:

- Processos de iniciação: definem e autorizam o projeto ou fase do projeto;
- Processos de planejamento: definem e refinam os objetivos e planejam as ações necessárias para atingir os objetivos e o escopo para os quais o projeto foi concebido;
- Processos de execução: integram pessoas e outros recursos visando a execução do plano de gerenciamento do projeto;
- Processos de monitoramento e controle: medem e monitoram o progresso do projeto regularmente para identificação das variações em relação ao plano, possibilitando a tomada de ações corretivas para atendimento aos objetivos do projeto.
- Processos de encerramento: formalizam a aceitação final do produto, serviço ou resultado, conduzindo o projeto ou fase a um final ordenado.

Nestes cinco grupos de processos, estão distribuídos os 44 processos de gerenciamento de projetos, distribuídos em nove áreas de conhecimento segundo o PMI descreve em seu guia (2004, p. 10), sendo estas:

- Gerenciamento da integração: área que engloba os processos requeridos para assegurar que todos os elementos do projeto sejam adequadamente

coordenados e integrados, garantindo que o seu todo seja sempre beneficiado.

- Gerenciamento de escopo: área que engloba os processos necessários para assegurar que, no projeto, esteja incluído todo o trabalho requerido, e somente o trabalho requerido, para concluí-lo de maneira bem-sucedida.
- Gerenciamento de tempo: área que engloba os processos necessários para assegurar a conclusão do projeto no prazo previsto. É uma das aeras mais visíveis do gerenciamento de projetos.
- Gerenciamento de custos: área que engloba os processos requeridos para assegurar que os produtos ou serviços do projeto estarão em conformidade com o solicitado pelo cliente, ou contratante.
- Gerenciamento de recursos humanos: área que engloba os processos requeridos para fazer uso mais efetivo do pessoal envolvido com o projeto.
- Gerenciamento das comunicações: área que engloba os processos requeridos para assegurar que as informações do projeto sejam adequadamente obtidas e disseminadas.
- Gerenciamento de riscos: área que visa planejar, identificar, qualificar, quantificar, responder e monitorar os riscos do projeto.
- Gerenciamento das aquisições: área que engloba os processos requeridos para adquirir bens e serviços de fora da organização promotora. Também conhecido como gerenciamento de suprimentos ou contratos.

O PMBOK® 3ª edição (PMBOK, 2004) aborda 42 processos divididos em noves áreas de conhecimento que formam um fluxo contínuo de processos. O quadro abaixo resume o mapeamento dos processos de gerenciamento de projetos com sua associação aos grupos de processos e áreas de conhecimento.

Quadro 2: Processos de gerenciamento de projetos

	Grupos de Processos de Gerenciamento de Projetos				
	Grupo de processos de iniciação	**Grupo de processos de planejamento**	**Grupo de processos de execução**	**Grupo de processos de monitoramento e controle**	**Grupo de processos de encerramento**
Gerenciamento da integração do projeto	Desenvolvimento do termo de abertura do projeto Desenvolvimento da declaração de escopo preliminar do projeto	Desenvolvimento do plano de gerenciamento do projeto	Orientação e gerenciamento da execução do projeto	Monitoramento e controle do trabalho do projeto Controle integrado de mudanças	Encerramento do projeto

Gerenciamento do escopo do projeto		Planejamento do escopo Definição do escopo Criação da EAP (Estrutura Analítica do Projeto)		Verificação do escopo Controle do escopo	
Gerenciamento de tempo do projeto		Definição da atividade Sequenciamento de atividades Estimativa de recursos da atividade Estimativa de duração da atividade Desenvolvimento do cronograma		Controle do cronograma	
Gerenciamento de custos do projeto		Estimativa de custos Orçamento		Controle de custos	
Gerenciamento da qualidade do projeto		Planejamento da qualidade	Realização da garantia da qualidade	Realização do controle da qualidade	
Gerenciamento de recursos humanos do projeto		Planejamento de recursos humanos	Contratação ou mobilização da equipe do projeto Desenvolvimento da equipe do projeto	Gerenciamento da equipe do projeto	
Gerenciamento das comunicações do projeto		Planejamento das comunicações	Distribuição das informações	Relatório de desempenho Gerenciamento das partes interessadas	
Gerenciamento de riscos do projeto		Planejamento do gerenciamento de riscos Identificação de riscos Análise qualitativa de riscos Análise quantitativa de riscos Planejamento de respostas a riscos		Monitoramento e controle dos riscos	
Gerenciamento de aquisições do projeto		Planejamento das compras e aquisições Planejamento das contratações	Solicitação das respostas de fornecedores Seleção de fornecedores	Administração de contrato	Encerramento de contrato

Fonte: PMBOK, 2004, p. 70.

O sucesso na gestão de um projeto está relacionado ao alcance dos seguintes objetivos:

- Entrega dentro do prazo previsto;
- Entrega dentro do custo orçado;
- Entrega com nível de desempenho adequado;

- Aceitação pelo cliente;
- Atendimento de forma controlada às mudanças de escopo, e;
- Respeito à cultura da organização.

A pessoa responsável pela gestão do projeto é o gerente do projeto, que consequentemente, também é responsável pelo sucesso ou insucesso do projeto. O gerente do projeto deve ser designado no início do projeto, tendo o apoio da alta administração da organização. Ele deve ter sua competência reconhecida pelos demais interessados no projeto, embora não necessite ter conhecimento técnico aprofundado, uma vez que suas competências são mais voltadas para o entendimento geral e não específico (PMBOK, 2004).

2.5 O GERENCIAMENTO DE PROJETOS DE SOFTWARE

As regras do gerenciamento de projetos também podem ser aplicadas aos projetos de *software*. A gerência de projetos de desenvolvimento de *software* deve ter em mente que sua atividade deve objetivar a qualidade, produtividade e a redução de riscos através do planejamento e execução do desenvolvimento do produto (PMBOK, 2004).

O modelo de gerência de projetos processual destaca a forma com que os processos são divididos, facilitando a comunicação entre as fases do projeto. O modelo incorpora os conceitos de gerência de projetos do PMI em um modelo de gerência de projeto de *software*.

A Gerência é um item fundamental dentro de um projeto de software, e deve estar presente durante todo o ciclo de vida do desenvolvimento de um sistema. A capacidade gerencial em um projeto de software deve levar em considerações fatores como (ENGENHARIA, 2009):

- Gestão da qualidade: alicerçada sobre a qualidade do produto e qualidade do processo. No primeiro é necessário verificar se as necessidades pré-estabelecidas pelos clientes estão implementadas no produto. No segundo, é

importante analisar se o processo está organizado, estruturado para produzir subprodutos de qualidade;

- Gestão de configuração: conjunto de ações (ou tarefas) que possibilitam identificar todos os artefatos gerados pelo processo de software, estabelecendo um relacionamento entre eles. A gestão de configuração também é responsável por subsidiar a atividade de gerenciamento em projetos, pois esta necessita controlar as diferentes versões geradas para um mesmo produto;
- Gestão de expectativa: caracterizada pelo estabelecimento de uma relação harmoniosa entre clientes internos e externos;
- Gestão de risco: é a gestão para minimizar as condições que comprometem ou impendem à realização de alguma parte projeto;
- Gestão de escopo: contextualizada por delimitar de maneira recorrente, as necessidades, os produtos e as pessoas;
- Gestão de aquisição: caracterizada pelo estabelecimento de um conjunto de atividades e critérios que possibilitam a obtenção de bens e serviços fora dos limites organizacionais.

3 METODOLOGIAS ÁGEIS

Até o ano de 2001, a maior parte dos modelos de sucesso para a realização de projetos de software era baseada no modelo Cascata, onde existe uma grande fase de planificação que dá todo o suporte ao desenvolvimento. Nestes modelos a mudança de requisitos é muito difícil de ser aceita em fases posteriores ao planejamento, pois exige uma regressão para a fase de concepção do plano. Os métodos ágeis para desenvolvimento de software formaram uma alternativa aos métodos tradicionais, considerados pesados e caracterizados pelo foco excessivo em documentação.

Os Métodos Ágeis de Desenvolvimento de Software surgiram, nos anos 90, como uma resposta às cobranças por inovação em prazos cada vez mais reduzidos, constantes necessidades de mudanças de requisitos e mau desempenho de grande parte dos projetos de desenvolvimento de software. Nesta época, as metodologias tradicionais começaram a ser questionadas devido ao grande volume de documentação, e um crescente número de organizações passou a adotar um ou mais método ágil para produzir software com menos documentação, sob condições de mudanças rápidas e com foco na satisfação do cliente (MOL, 2007).

As metodologias ágeis estabelecem práticas que auxiliam a ampliar a capacidade de aprendizado e adaptação dos projetos.

Segundo Vale (2008a, p. 30),

> A adaptação e o aperfeiçoamento contínuo geram um processo evolutivo que emerge das relações sistêmicas dos organismos com o seu meio ambiente. Em projetos ágeis, seus processos também emergem das relações sistêmicas estabelecidas entre as pessoas envolvidas no projeto (stakeholders) e o seu meio de negócio. A capacidade de adaptação é elemento essencial de diferenciação do modelo ágil em relação aos processos tradicionais para desenvolvimento de software. Os pequenos ciclos de avaliação e aprendizado que o compõe desafiam equipe e cliente a evoluírem seu processo por meio da detecção, análise, correção e absorção sistemática de conhecimento sobre erros, ineficiências ou falsas suposições encontrados durante seu tempo de vida.

Na concepção de metodologias ágeis, as correções sistemáticas de pequenos erros cometidos devido às falsas suposições são denominadas Adaptive Software Development (ASD), e são consideradas a chave para o funcionamento dos seus ciclos de especulação, colaboração e aprendizado (VALE, 2008a).

O termo Metodologias Ágeis surgiu a partir de um manifesto criado por dezessete profissionais veteranos na área de software, representantes de diversas metodologias, que se reuniram no ano de 2001 nos EUA para discutir o que havia em comum entre estas metodologias. Esse manifesto foi chamado de Manifesto for Agile Software Development, ou simplesmente Manifesto Ágil. Os participantes deste manifesto foram: Kent Beck, Mike Beedle, Arie van Bennekum, Alistair Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C. Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland e Dave Thomas.

Segundo o Manifesto Ágil (AGILEALLIANCE, 2001):

> Estamos evidenciando maneiras melhores de desenvolver software fazendo-o nós mesmos e ajudando outros a fazê-lo. Através desse trabalho, passamos a valorizar:
>
> - Pessoas e interação MAIS QUE processos e ferramentas;
>
> - Software em funcionamento MAIS QUE documentação abrangente;
>
> - Colaboração com o cliente MAIS QUE negociação de contratos; e
>
> - Responder a mudanças MAIS QUE seguir um plano.
>
> Ou seja, mesmo tendo valor os itens à direita, valorizamos mais os itens à esquerda.

As metodologias ágeis visam o auto-aperfeiçoamento. Nesse sentido, o Manifesto Ágil (AGILEALLIANCE, 2001) afirma que um “time de projeto deve refletir, em intervalos regulares, sobre como ser mais efetivo, e assim afinar e ajustar seu comportamento apropriadamente”.

Um dos princípios do Manifesto Ágil diz que (AGILEALLIANCE, 2001) "as Mudanças de requisitos são bem vindas, mesmo em estágios tardios do

desenvolvimento. Processos Ágeis devem admitir mudanças que trazem vantagens competitivas para o cliente".

Segundo Vale (2008b, p.22),

> Projetos Ágeis são semelhantes a campeonatos esportivos. Um campeonato é disputado jogo a jogo. Cada jogo terá uma história diferente e apresentará um resultado diferente. A equipe vencerá se conseguir focar em resultados, se adaptar com a realidade de cada jogo e ao mesmo tempo aperfeiçoar sua forma de jogar ao longo do campeonato. Projetos tradicionais esperam o campeonato terminar para refletir e melhorar. O modelo ágil, por outro lado, trata cada uma de suas iterações como as equipes esportivas tratam seus jogos. A primeira atitude de um técnico esportivo depois de um jogo é reunir a equipe para falar sobre os pontos positivos e negativos do último jogo. No contexto do modelo ágil, tais reuniões são chamadas de Reuniões de Retrospectiva.

De acordo com Câmara (2008, p. 19),

> Métodos ágeis são propostas incomuns de técnicas para projetos de software. XP, como o nome sugere, é algo um pouco mais radical. Propostas estranhas de técnicas esquisitas fazem os gerentes de projetos temerem utilizá-las em grandes projetos. Para atrapalhar ainda mais as poucas iniciativas de se quebrar paradigmas, o radicalismo de alguns pensadores de XP afastam a compreensão sobre as reais vantagens e os benefícios para o negócio que podemos alcançar com estes métodos.

No Manifesto Ágil foram descritos 12 princípios que as metodologias de desenvolvimento ágil devem seguir (vide quadro abaixo).

Quadro 3: Princípios metodologias ágeis

Princípios	**Descrições**
Satisfazer o cliente através de entregas iniciais e contínua de software que agregue valor àquele.	Se um cliente necessita de um software de 10 módulos, o manifesto ágil prega que deve ser entregue para o cliente os módulos conforme ficarem prontos. Isso cria a possibilidade do cliente começar a ter um retorno para o seu negócio mais cedo, através desses módulos prontos.
Aceitar mudanças nos requisitos mesmo que tarde no projeto.	Processos ágeis têm que serem abertos às mudanças para que o cliente tenha vantagens competitivas.
Entregar software funcional frequentemente. Preferencialmente em uma escala de tempo pequena.	Quanto antes o cliente puder usar o software, antes ele terá condições de entender melhor as necessidades do negócio. Assim é possível detectar mudanças mais cedo.
Pessoas do negócio e desenvolvedores têm que trabalhar juntos durante o projeto.	Isso ajuda os desenvolvedores a entender melhor as necessidades do negócio.

Construa projetos em torno de indivíduos motivados. Dê-lhes o ambiente e o apoio que necessitam, confie no trabalho deles.	Ter pessoas motivadas no projeto potencializa a probabilidade do projeto ser um sucesso.
O método mais eficaz e eficiente de se comunicar com a equipe de desenvolvimento é o face-a-face.	Preferir comunicação verbal à comunicação escrita.
Software que funcione é a primeira métrica de progresso.	Para evitar que o progresso do projeto seja medido por atividades que não tenham um retorno direto para o negócio do cliente. Muitas vezes os projetos já estão 60% prontos, porém essa porcentagem é somente o planejamento do projeto, se o projeto for cancelado nessa etapa o cliente não tem nenhum módulo que funcione, ou seja, não tem nada que realmente de retorno direto para o seu negócio.
Processos ágeis promovem desenvolvimento sustentável.	Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
Ter atenção contínua para a excelência técnica e para os bons designs promovem a agilidade.	Obviamente, quanto melhor estruturado estiver o software, mais fácil será alterá-lo.
Simplicidade.	Sempre evite complexidade desnecessária. Não insira complexidade com o objetivo de suportar uma necessidade futura, postergue a inclusão da complexidade para quando a necessidade futura chegar.
As melhores arquiteturas, requisitos e designs surgem de equipes que se auto-organizam.	Equipes que se auto-organizam são equipes que estão empenhadas no projeto, isso facilita a excelência técnica.
Em intervalos regulares, a equipe reflete como ser mais eficiente, então faz os ajustes necessários.	A equipe pode fazer otimizações na sua forma de trabalhar para que o projeto se beneficie.

Fonte: AGILEALLIANCE, 2001.

Com base nestes princípios, pode-se afirmar que os métodos ágeis são (AGILEALLIANCE, 2001):

- Incrementais: pequenas versões e ciclos rápidos de desenvolvimento;
- Cooperativos: estimulam a proximidade com o cliente e a interação entre os programadores;
- Diretos: aprendizado e documentação simplificada;
- Adaptativos: acomodação hábil de mudanças durante o projeto.

3.1 CARACTERÍSTICAS DAS METODOLOGIAS ÁGEIS

As metodologias ágeis têm como pontos fortes a frequente participação do

cliente e grande flexibilidade. De acordo com as bases das Metodologias Ágeis, as pessoas têm papel fundamental no desenvolvimento dos projetos, sendo que para isso é essencial que em todas as equipes exista uma boa comunicação e haja motivação e preocupação de cada indivíduo com a qualidade. O cliente passa a fazer parte da equipe de desenvolvimento, uma vez que a todo o momento é questionado sobre prioridades e testes de versões. É também frequentemente chamado a intervir, iteração a iteração, tendo um papel decisivo na definição dos novos requisitos, contrariando a prática de quase tudo deve ser planificado e acordado no inicio do projeto. Nenhum projeto é totalmente previsível, portanto ser ágil, é ter conhecimento desta realidade e aceitar que os requisitos habitualmente mudam.

Outras características importantes exibidas pela aplicação de metodologias ágeis foram descritas por Neumann, Baureis e Stock (2009, p. 65):

- Orientação ao valor proporcionado ao cliente;
- Desenvolvimento das competências individuais;
- Times pequenos;
- Autodisciplina sustentável;
- Intensa colaboração;
- Reduzido custo de transferência da informação;
- Tempo reduzido para feedback;
- Aprendizado e adaptação constantes.

Abaixo, são exibidos outros aspectos que também caracterizam os métodos ágeis (GERENCIAMENTO, 2009):

- Estratégia de Negócio: Para ajudar a equipe é preciso fazer com que ela entenda do negócio que esta por trás do projeto. Por este motivo, é muito importante o conceito de responsabilidade de cada pessoa dentro da equipe ágil;
- Gestão de Risco: Para analisar os riscos fazemos sempre suposições sobre o futuro, sendo este o mais próximo possível. No caso das equipes ágeis esse plano de riscos é feito para no máximo uma visão de duas semanas. Normalmente é feito semanalmente, ou por períodos curtos chamados de *sprints*;

- Estimativa de Tempo: Para estimar o tempo em equipes ágeis normalmente trabalha-se uma semana para ver o progresso real, e então pode-se entender e calcular a velocidade semanal em cima de uma taxa de cada programador ou membro da equipe, fazendo assim uma previsão científica de quanto tempo o trabalho irá levar;
- Qualidade: Os testes são realizados imediatamente após a realização da tarefa e não depois de tudo estar completo. Isso gera uma maior dinamicidade, criando uma forma mais livre e um produto final já testado;
- Comunicação: As reuniões de controle demoram no máximo 15 minutos no início ou no final do dia. Podem receber a participação do cliente;
- Motivação: Todos os membros da equipe ágil participam do *brainstorming* e *kick-off* do projeto, entendendo a importância e reconhecendo de seus respectivos deveres dentro da equipe. É importante a ação continua de motivação por parte do gerente de projeto, para que a equipe não tenha uma queda de desempenho nos *sprints*;
- Satisfação do cliente: Notamos que uma das coisas mais fáceis de se atingir nesse tipo de método de gerenciamento é a satisfação do cliente. Como o mesmo está muito presente e participa ativamente das atividades do projeto, e as entregas têm prazos menores, pode-se dizer que o cliente terá confiança plena que o produto sairá conforme ele deseja;
- Fases de Transição: Ocorre quando libera-se uma versão de funcionalidades para que possa ser testada e avaliada pelo cliente, e aprovada antes que a equipe ágil se mova ou transpasse para a próxima fase de desenvolvimento;
- Lições Aprendidas: Em vez de se esperar até o final do projeto, é possível se beneficiar com esses dados desde o início do projeto, tendo-se como base os *sprints* de trabalho. Fazendo-se isso, a garantia de sucesso e a rapidez para tomadas de decisões são aumentadas.

3.1.1 Características das Equipes Ágeis

É fundamental que algumas características estejam presentes entre as pessoas de uma equipe ágil. Estas características foram apresentadas por Tomás (2009, p. 6):

- Competência: independente da metodologia (ágil ou tradicional) diz respeito a talento inato, habilidades específicas relacionadas a software e a um conhecimento global do processo que a equipe decidiu aplicar;
- Foco comum: os membros podem ter diferentes competências e habilidades, mas todos eles devem ter o mesmo foco, que é entregar um incremento de software em funcionamento ao cliente dentro do prazo prometido;
- Colaboração: a equipe precisa de colaboração entre si, com o cliente e com os gerentes de forma a conseguir analisar, avaliar e usar/comunicar informação de forma eficiente;
- Capacidade de tomada de decisão: é importante que a equipe tenha autonomia para tomar decisões de tópicos técnicos e de projeto;
- Habilidade de resolver problemas vagos: uma equipe ágil lida continuamente com ambiguidades e será confrontada com modificações. Os problemas que resolvem num dia, podem não ser significantes numa fase posterior, porque teve de se mudar a maneira de realizar algo. Contudo, a equipe aprendeu a resolver aquele tipo de problema;
- Respeito e confiança mútua: a equipe tem de funcionar como um todo;
- Auto-organização: a equipe organiza-se para o trabalho ser feito; a equipe organiza o processo para melhor acomodar seu ambiente local; a equipe organiza o cronograma de trabalho para conseguir melhorar a entrega do incremento de software.

3.2 DIFERENÇAS ENTRE MÉTODOS ÁGEIS E MÉTODOS CLÁSSICOS

A maioria das metodologias ágeis nada possui de novo em relação às metodologias tradicionais. O que as diferencia das metodologias tradicionais (clássicas) são o enfoque e os valores.

A principal idéia das metodologias ágeis é o enfoque nas pessoas e não em processos, como é a idéia dos métodos tradicionais, gastando assim menos tempo com documentação e mais tempo com implementação. Dessa forma, as metodologias ágeis são adaptativas ao invés de serem preditivas. Com isso, elas se adaptam a novos fatores decorrentes do desenvolvimento do projeto, ao invés de procurar analisar previamente tudo o que pode acontecer no decorrer do desenvolvimento.

Portanto, a diferença primordial está em evitar os problemas já conhecidos de tentar estimar e planejar software em detalhes, atividades que comprovadamente falham. Junto a isso, somam-se os atrativos de se manter uma equipe mais motivada e produtiva, envolver mais pessoas (como o cliente) na tomada de decisão e focar em trabalhos que realmente agregam valor.

Segundo Bernardino, Gomes e Vasconcelos (2005) as metodologias ágeis assim como as metodologias tradicionais possuem pontos fortes e pontos fracos. O quadro abaixo apresenta uma comparação entre as metodologias ágeis e as metodologias tradicionais em algumas áreas principais.

Quadro 4: Comparação entre metodologias ágeis e tradicionais

Principais áreas	Metodologias ágeis	Metodologias guiadas por planejamento
Desenvolvimento	Ágil, voltado ao conhecimento, arranjado e colaborativo.	Orientado ao planejamento, habilidades adequadas, acesso ao conhecimento externo.
Clientes	Dedicado, voltado ao conhecimento, arranjado, colaborativo, representativo e com poder.	Acesso ao conhecimento, colaborativo, representativo e clientes com poder.
Requisitos	Largamente emergentes, mudanças rápidas.	Conhecidos previamente, largamente estáveis.
Arquitetura	Projetada para requisitos atuais.	Projetada para requisitos atuais e futuros.
Refatoração	Barata.	Cara.
Tamanho	Pequenos times e produtos.	Grandes times e produtos.
Objetivo primário	Valor rápido.	Garantia elevada.

Fonte: Boehm, 2002 apud Bernardino, Gomes e Vasconcelos, 2005, p. 26.

3.3 DESENVOLVIMENTO RÁPIDO DE APLICATIVOS (SOFTWARES)

As metodologias e ferramentas de desenvolvimento rápido de aplicativos (*Rapid Application Development* - RAD) permitem desenvolver sistemas de forma muito mais rápida, especialmente aqueles sistemas onde a interface de usuário representa um componente importante.

O processo RAD é semelhante ao processo de prototipação. Ambos são altamente repetitivos e dão ênfase à velocidade de desenvolvimento. A prototipação muitas vezes utiliza linguagens especializadas, tais como linguagens de quarta geração e geradores de tela, enquanto que os pacotes RAD incluem diferentes ferramentas com recursos semelhantes. Com as ferramentas RAD, os desenvolvedores aperfeiçoam e ampliam a versão inicial por meio de repetições múltiplas até que ela esteja adequada para o uso operacional (TURBAN, MCLEAN, WETHERBE, 2007).

As ferramentas trabalham juntas como parte do pacote integrado, produzindo componentes funcionais para uma versão final do sistema, em vez de criar simulações ou versões em escala reduzida. Os produtos típicos de RAD são adequados ao desenvolvimento de uma grande variedade de plataformas, em especial cliente/servidor, cada vez mais utilizada, e plataformas baseadas na Web (TURBAN, MCLEAN, WETHERBE, 2007).

Além dos benefícios de velocidade e portabilidade, o RAD é usado para criar aplicativos mais fáceis de manter e modificar. No entanto, os pacotes de RAD também apresentam algumas desvantagens, pois tal como na prototipação, se não houver um critério claro para concluir o processo repetitivo de desenvolvimento, ele poderá continuar indefinidamente. Os pacotes de RAD podem possuir recursos para criar documentação do sistema, mas isso não garante que os desenvolvedores irão produzir a documentação de boa qualidade (TURBAN, MCLEAN, WETHERBE, 2007).

3.4 Principais Metodologias Ágeis

As metodologias ágeis apresentam uma série de propostas ágeis, cada uma delas possui peculiaridades e formas de encarar o desenvolvimento de *software*. As principais metodologias ágeis para desenvolvimento de *software* serão descritas a seguir.

3.4.1 Feature-Driven Development – FDD

A FDD (*Feature-Driven Development*) é dirigida pelas funcionalidades de valor para o cliente. Seu diferencial é que ela trabalha com diagramas de UML coloridos auxiliando o entendimento desses pela equipe (FELSING, PALMER, 2002).

> O FDD foi criado em 1999, em Cingapura, por um time liderado por Jeff De Luca, que incluía também Tom De Marco, Tim Lister, Jerry Weinberg e Frederic Brooks. Trata-se de um método de desenvolvimento de software específico para aplicações críticas e está baseado em processos bem definidos que podem ser repetidos. O FDD é mais um método de gerenciamento de software do que um ciclo de vida de desenvolvimento de software propriamente dito (DIAS, 2005, p. 71).

As características (*features*) são funções que o cliente valoriza e que podem ser implementadas em menos de duas semanas, e tem um formato próprio para serem descritas. Há então uma série de fases até que se cumpra a realização da característica.

Resumidamente, FDD é dividido em 5 fases que explicam sua função. São elas, conforme descrito por Câmara (2008, p. 19):

- *Shape Modeling*: é uma forma de questionar se todos compreendem o que é para fazer, analisar requisitos não-funcionais e modelo de arquitetura;
- *Feature List*: É a representação do escopo listando a compreensão do que é para ser feito e os requerimentos a serem desenvolvidos;

- *Plan by subject area*: É a modularização da lista em conjuntos de funcionalidades relacionadas, permitindo o desenvolvimento de parte do sistema autonomamente;
- *Design by feature set*: É uma orientação que determina o desenvolvimento com base no domínio do problema. Sugere-se nesta fase uma modelagem profunda e detalhada em UML;
- *Build by Chief Programmer Work Package*: É o empacotamento de pequenas funcionalidades, uma redução evolutiva que nasce na fase 2 até a fase 4. Prioriza-se este pacote, codificando suas funcionalidades e criando unit tests.

A FDD define também 4 camadas de arquitetura de software, conforme descrito por Câmara (2008, p. 19):

- UI: *User Interface*;
- PD: *Problem Domain* (lógica do negócio);
- DM: *Data Management*;
- SI: *Systems Interfaces*.

A FDD tem como característica a arquitetura baseada em funcionalidade (*feature*), onde cada funcionalidade é definida com uma fórmula simples, que permite ser repetível e confiável.

Ainda segundo o mesmo autor, fórmula da funcionalidade tem a seguinte estrutura: *<action> <result> <object>* (vide exemplo na figura abaixo).

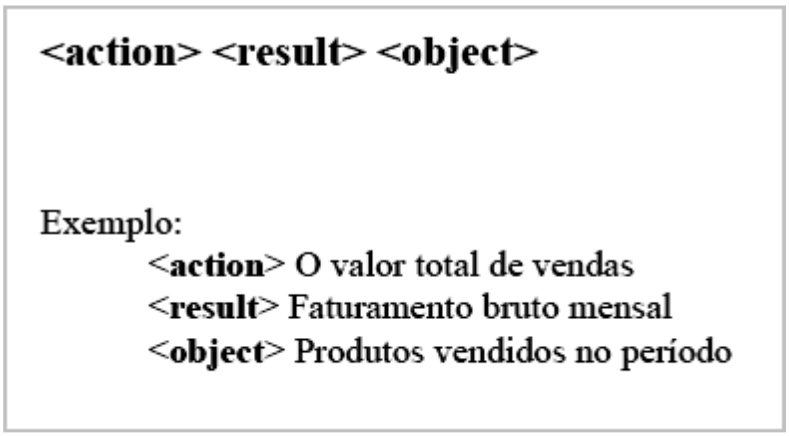

Figura 1: Fórmula funcionalidade FDD

Fonte: Camara, 2008, p.20.

3.4.2 Adaptative Software Development - ASD

O *Adaptative Software Development* – ASD foi criado por Highsmith em 1992. Trata-se de uma evolução do *Rapid Application Development* - RAD, o qual foi projetado para lidar com ambientes complexos e cheios de incertezas. O ASD estimula o aprendizado durante o processo de desenvolvimento e a adaptação constante a mudanças de negócio e do projeto, além de encorajar o desenvolvimento iterativo e incremental (IID) com a liberação constante de novas versões (DIAS, 2005).

O ASD é explicitamente orientado para componentes ao contrário de orientado a tarefas. Isso significa que o foco está mais em seus resultados e qualidades que em tarefas e processos usados para produzir o resultado. A forma que o ASD conduz isso é através de ciclos de três fases. Essas fases são: *Speculate* (Especular): utilizado no lugar de Planejamento; *Collaborate* (Colaborar): enfatiza a importância da equipe desenvolver um software altamente alterável; *Learn* (Aprender): realça a necessidade de reconhecer e reagir aos erros, e o fato que requisitos também podem mudar durante o desenvolvimento (DIAS, 2005).

Segundo seu criador, o ASD é indicado para equipes pequenas, mas pode ser adaptado para equipes maiores.

3.4.3 Lean Development - LD

O *Lean Development* (LD) é considerado o método ágil de maior foco estratégico devido a suas origens estarem baseadas na indústria automotiva, especialmente no Modelo Toyota de Produção. O LD é derivado dos princípios de produção enxuta, a qual reestruturou a indústria de automóveis no início de 1980. Foi criado por Bob Charette e tem como principais objetivos a redução de prazo, custo e nível de defeitos no desenvolvimento em um terço.

O LD é composto por três fases (DIAS, 2005), sendo:

- *Startup* (inicial): nesta fase é realizado o planejamento do desenvolvimento;
- *Steady state* (estado fixo): nesta fase é realizada a concepção do desenho do software, composta de uma série de ciclos pequenos de desenvolvimento;
- *Transition-renewal* (transição-renovação): após a entrega do produto, este é mantido em constante fase de renovação e transição. Nesta fase também é desenvolvida e entregue a documentação.

3.4.4 Scrum

O *Scrum* é um processo ágil que permite manter o foco na entrega de maior valor de negócio, no menor tempo possível. Isto permite a rápida e contínua inspeção do software em produção.

O *Scrum* possui um processo bem definido com uma fase de planejamento e de encerramento, conforme ilustrado na figura abaixo. Entre estas fases, há uma fase chamada de Sprint, com duração de aproximadamente 4 semanas, que ocorre várias vezes durante o desenvolvimento do projeto. São as iterações que caracterizam as Metodologias Ágeis. O *Sprint* é como uma caixa preta onde ocorre o desenvolvimento do produto.

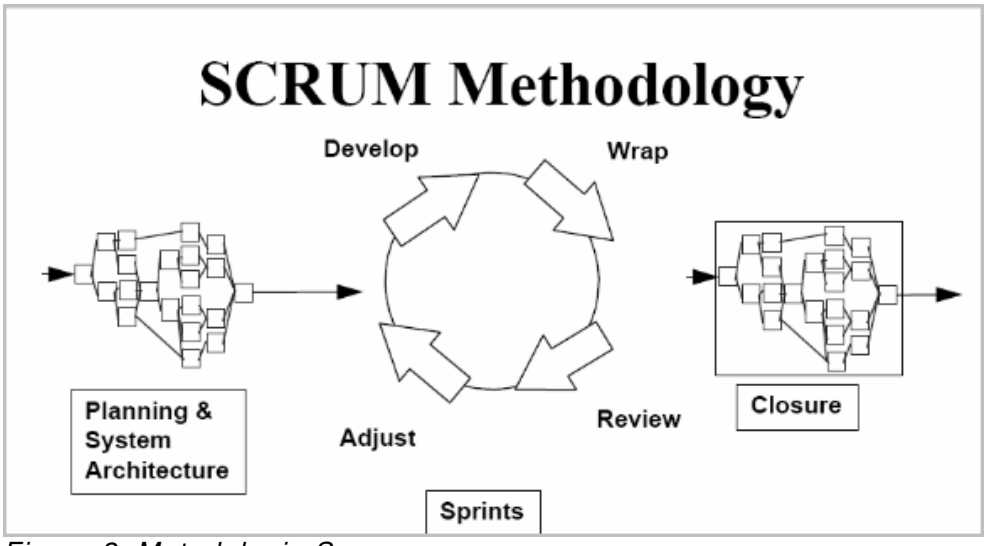

Figura 2: Metodologia Scrum

Fonte: Agile Software Development with Scrum (BEEDLE, SCHWABER, 2002).

De acordo com Câmara (2008, p. 21),

> SCRUM é um método de gerenciamento de software que pode ser usado com XP ou MSF. É baseado na teoria do controle empírico de processos e seus fundamentos são originados na indústria de manufatura japonesa. Ciclo de vida empírico é um ciclo baseado na observação dos fatos. Muitos gerentes de projetos não gostam do método reativo sugerido pelo SCRUM e preferem trabalhar com um planejamento que na minha leitura é um trabalho especulativo, pois tenta prever uma seqüência de atividades lineares. Por mais que o cronograma tenha um valor que é o planejamento prévio, ou seja, pensar antes de sair fazendo, perde muito em tentar prever atividades distantes cheias de dependências predecessoras e não facilita o controle das mudanças de requisitos.

Câmara (2008, p. 21) afirma ainda que:

> Segundo o SCRUM, o desenvolvimento deve ser trabalhado em 3 níveis: Sprint, Release e Product. O ponto central é que os requisitos são convertidos em uma lista que contém valores do cliente chamada Product Backlog. Um subconjunto desta lista é criado e chamado de Release Backlog. Este subconjunto é particionado mais uma vez transformando-se em Sprint, uma espécie de acordo de desenvolvimento de funcionalidades que após aceito pela equipe não deve ser mais alterado. Praticando SCRUM, o que mais chama a atenção é a simplicidade. Controlar projetos desta forma é participar de um jogo competitivo e saudável em que todos se auto-avaliam todos os dias (daily stand-up meeting) tornando possível resultados e técnicas de melhoria contínua.

O *Scrum* mantém uma lista de funcionalidades que deverão ser implementadas, essa lista é chamada de *Product Backlog*. Para cada iteração ou Sprint, é feita uma reunião inicial de planejamento, chamada de *Sprint Planning Meeting*, onde itens desta lista são priorizados pelo cliente, chamado no *Scrum* de *Product Owner*. A equipe define quais funcionalidades poderão ser atendidas dentro da iteração de acordo com a capacidade da mesma. Essa lista planejada para a iteração é chamada de *Sprint Backlog* (SCHWABER, 2004).

Diariamente, durante o *Sprint*, a equipe faz uma reunião chamada de *Daily Meeting* que ajuda a manter a comunicação do time sobre o andamento do projeto, disseminar conhecimento e identificar possíveis impedimentos. No final de cada *Sprint*, na reunião chamada *Sprint Review*, a equipe apresenta as funcionalidades que foram concluídas durante o *Sprint*.

Ao final do *Sprint* deve sair um produto com valor agregado, ou seja, é feito um incremento no produto. Esse ciclo se repete várias vezes até que o *Product Backlog* seja todo atendido. A figura abaixo representa o ciclo do *Sprint*.

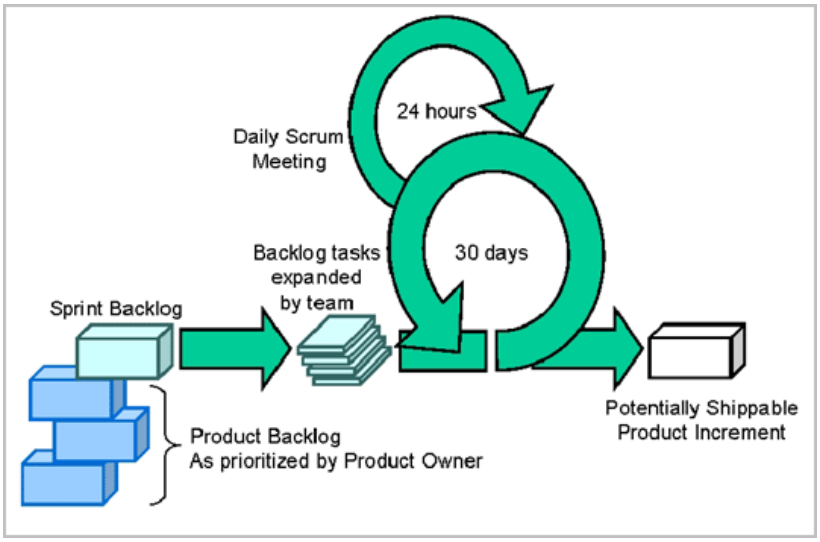

Figura 3: Ciclo do Sprint

Fonte: Agile Software Development with Scrum (BEEDLE, SCHWABER, 2002).

O *Scrum* possui 3 grupos de fases: *Pregame*; *Game*; e *Postgame*, tendo ainda atividades para cada uma destas fases (CAMARA, 2008).

Quanto aos papéis e responsabilidades do *Scrum*, este define basicamente apenas 3 papéis: *Product Owner*; *Scrum Master*; e *Scrum Team* (CAMARA, 2008).

As características de cada um desses papéis são descritas a seguir:

- *Product Owner*: Traduzido literalmente significa Dono do Produto. É o cliente do projeto, tipicamente um usuário chave que representa os interesses de todos os envolvidos com o software (SCHWABER, 2004). É quem define e prioriza as funcionalidades que compõem o Product Backlog. Deve estar acessível à equipe, tem a responsabilidade de não trazer novas funcionalidades durante o *Sprint* e de fazer a validação do produto ao final de cada *Sprint*.
- *Scrum Team*: É a Equipe responsável pelo desenvolvimento do produto de acordo com as prioridades definidas pelo *Product Owner*. Não há uma definição de papéis formal como programador, testador, arquiteto ou analista,

o que não impede que haja estas funções. Todos trabalham juntos para completar as funcionalidades que todos conjuntamente se comprometeram para o *Sprint*. A equipe deve ser idealmente pequena, de 5 a 9 pessoas, porém pode ser maior (BEEDLE, SCHWABER, 2002).

- *Scrum Master*: É o papel do gerente do projeto, mas pode ser exercido por qualquer membro da equipe. Responsável por garantir o trabalho da equipe através da remoção de obstáculos identificados e da proteção da equipe contra interferências externas. Deve disseminar e aplicar os valores e práticas do *Scrum* na equipe.

Os principais artefatos definidos pelo *Scrum* são o *Product Backlog*, que é a lista de funcionalidades definidas pelo *Product Owner*. O *Sprint Backlog* que é uma lista dos itens do *Product Backlog* que foram priorizados para serem atendidos no *Sprint*. E o *Burndown Chart* que consiste de um gráfico para acompanhamento diário do *Sprint* quanto à velocidade do time pelo consumo diário de horas.

No que se refere ao planejamento, execução e controle do *Sprint*, é preciso observar que durante o *Sprint* cada funcionalidade do software é trabalhada de forma a ser completada dentro do mesmo *Sprint*. Em se tratando de software, a funcionalidade é planejada, projetada, codificada e testada gerando software funcionando. O tamanho do *Sprint*, de 2 a 4 semanas, uma vez estabelecido não deve ter a data final alterada para que as métricas de velocidade do time sejam consistentes (SCHWABER, 2004).

3.4.5 Extreme Programming - XP

Extreme Programming (XP) consiste em um processo de desenvolvimento de *software* voltado para (TELES, 2009):

- Projetos cujos requisitos são vagos e mudam com freqüência;
- Desenvolvimento de sistemas orientados a objeto;
- Equipes pequenas, preferencialmente até 12 desenvolvedores;

- Desenvolvimento incremental (ou iterativo), onde o sistema começa ser implementado logo no início do projeto e vai ganhando novas funcionalidades ao longo do tempo.

O XP visa assegurar que o cliente receba o máximo de valor de cada dia de trabalho da equipe de desenvolvimento, sendo organizado através de um conjunto de valores e práticas que atuam harmonicamente para garantir que o cliente sempre receba um alto retorno do investimento em software.

De acordo com Bernardino, Gomes e Vasconcelos (2005) o XP possui as seguintes características:

- A maioria das regras do XP causa polêmica à primeira vista e muitas não fazem sentido se aplicadas isoladamente. É a sinergia de seu conjunto que sustenta o sucesso do XP, encabeçando uma verdadeira revolução no desenvolvimento de software.
- 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.

O quadro a seguir apresenta as considerações sobre os quatro valores (comunicação, simplicidade, feedback e coragem) de acordo com a metodologia XP, baseado na revisão de literatura de dois autores.

Quadro 5: Valores da metodologia XP

Valor	**Bernardino, Gomes e Vasconcelos, 2005**	**Teles, 2009**
Comunicação	A finalidade do princípio da comunicação é manter o melhor relacionamento possível entre clientes e desenvolvedores, preferindo conversas pessoais a outros meios de comunicação. A comunicação entre os desenvolvedores e o gerente do projeto também é encorajada.	Para que o cliente possa compartilhar o seu aprendizado com a equipe, é necessário que ele utilize permanentemente o valor da comunicação. A comunicação entre o cliente e equipe permite que todos os detalhes do projeto sejam tratados com atenção e a agilidade que merecem. O XP procura assegurar que a comunicação

		ocorra da forma mais direta e eficaz possível. Sendo assim, ele busca aproximar todos os envolvidos do projeto de modo que todos possam se comunicar face-a-face ou da forma mais rica que for viável.
Simplicidade	A simplicidade visa permitir a criação de código simples que não deve possuir funções desnecessárias. Por código simples entende-se implementar o software com o menor número possível de classes e métodos. Outra idéia importante da simplicidade é procurar implementar apenas requisitos atuais. A aposta do XP é que é melhor fazer algo simples, hoje e pagar um pouco mais amanhã para fazer modificações necessárias do que implementar algo complicado hoje que talvez não venha a ser usado, sempre considerando que requisitos são mutáveis.	É necessário que a equipe compreenda e utilize o valor da simplicidade, que nos ensina a implementar apenas aquilo que é suficiente para atender a cada necessidade do cliente. Ou seja, ao codificar uma funcionalidade é preciso se preocupar apenas com os problemas de hoje e deixar os problemas do futuro para o futuro. Não se deve tentar prever o futuro, pois raramente as previsões serão acertadas. Ao evitar especular sobre o que acontecerá amanhã, ganha-se tempo e permite-se que o cliente tenha acesso à funcionalidade mais rapidamente. Isso permite que o cliente utilize o software em seu negócio, gerando valor para ele e tornando viável que ele dê feedback para a equipe o quanto antes. Eventualmente, com base no feedback, a equipe poderá fazer generalizações quando elas se fizerem necessárias. Neste caso, entretanto, elas virão na forma de uma necessidade explícita e não como a especulação de algo que poderia vir a ser necessário no futuro.
Feedback	A prática do feedback constante significa que o programador terá informações constantes do código e do cliente. A informação do código é dada pelos testes constantes, que indicam os erros tanto individuais quanto do software integrado. Em relação ao cliente, o feedback constante significa que ele terá frequentemente uma parte do software totalmente funcional para avaliar. O cliente então, constantemente, sugere novas características e informações	Quando o cliente aprende com o sistema que utiliza e re-avalia as suas necessidades, ele gera feedback para a equipe de desenvolvimento. Isto é, ele realimenta a equipe com alterações nas necessidades que ainda serão implementadas e, eventualmente, naquela que fazem parte do software. O feedback é o mecanismo fundamental que permite que

	aos desenvolvedores. Eventuais erros e não conformidades são rapidamente identificados e corrigidos nas próximas versões. Desta forma, a tendência é que o produto final esteja de acordo com as expectativas reais do cliente.	o cliente conduza o desenvolvimento diariamente e garanta que a equipe direcione as suas atenções para aquilo que irá gerar mais valor.
Coragem	É necessário coragem para implementar os três valores anteriores. Por exemplo/; não são todas as pessoas que possuem facilidade de comunicação e têm bom relacionamento. A coragem também dá suporte à simplicidade, pois assim que a oportunidade de simplificar o software é percebida, a equipe pode experimentar. Além disso, é preciso coragem para obter feedback constante do cliente.	Dado que o sistema é desenvolvido de forma incremental, a equipe está continuamente fazendo a manutenção do software e criando novas funcionalidades. Em muitos casos, ela irá alterar algo que vinha funcionando corretamente, o que leva ao risco de gerar falhas no sistema. Por esta razão, a equipe precisa ser corajosa e acreditar que, utilizando as práticas e valores do XP, será capaz de fazer o software evoluir com segurança e agilidade.

Fonte: Adaptação do pesquisador.

O XP se baseia em 13 práticas (TELES, 2009):

- Cliente Presente;
- Jogo do Planejamento;
- Stand Up Meeting;
- Programação em Par;
- Desenvolvimento Guiado pelos Testes;
- Refactoring;
- Código Coletivo;
- Código Padronizado;
- Design Simples;
- Metáfora;
- Ritmo Sustentável;
- Integração Contínua;
- Releases Curtos.

O quadro abaixo apresenta as principais características de cada uma das práticas do XP.

Quadro 6: Práticas do XP

Prática	**Descrição**
Cliente presente	O XP trabalha com a premissa de que o cliente deve conduzir o desenvolvimento a partir do feedback que recebe do sistema. Para que a troca de feedback possa ocorrer e o cliente possa obter o máximo de valor do projeto, é essencial que ele participe ativamente do processo de desenvolvimento. Além disso, a sua presença viabiliza a simplicidade do processo em diversos aspectos, especialmente na comunicação.
Jogo do planejamento	O XP utiliza diversas formas de planejamento para assegurar que a equipe esteja sempre trabalhando naquilo que é o mais importante para o cliente. Por esta razão, todo projeto em XP é dividido em releases e iterações, de modo que cliente e equipe tenham diversas oportunidades de se reunir para revisar o planejamento. Releases são módulos do sistema que geram um valor bem definido para o cliente. Iterações, por sua vez, são períodos de tempo de poucas semanas (em torno de duas, em média) no qual a equipe implementa um conjunto de funcionalidades acordado com o cliente. No início de cada release e de cada iteração ocorre o jogo do planejamento. Trata-se de uma reunião onde o cliente avalia as funcionalidades que devem ser implementadas e prioriza aquelas que farão parte do próximo release ou da próxima iteração. No XP, as funcionalidades são descritas em pequenos cartões e são chamadas de estórias. Durante o jogo do planejamento as estórias são estimadas, para que o cliente possa conhecer o custo da implementação de cada uma delas. A estimativa é feita utilizando-se uma unidade especial que recebe o nome de ponto. O ponto representa uma unidade de tempo que varia ao longo do desenvolvimento de acordo com a velocidade da equipe, onde a velocidade indica o quanto a equipe foi capaz de implementar na iteração anterior.
Stand Up Meeting	A equipe de desenvolvimento se reúne a cada manhã para avaliar o trabalho que foi executado no dia anterior e priorizar aquilo que será implementado no dia que se inicia. Trata-se de uma reunião rápida que recebe o nome de stand up meeting, que em inglês significa reunião em pé.
Programação em par	No XP, os desenvolvedores implementam as funcionalidades em pares, ou seja, diante de cada computador, existem sempre dois desenvolvedores que trabalham juntos para produzir o mesmo código. Esta prática, que recebe o nome de programação em par permite que o código seja revisado permanentemente, enquanto é construído. Também contribui para que a implementação seja mais simples e eficaz, já que os desenvolvedores se complementam e têm mais oportunidades de gerar soluções inovadoras.
Desenvolvimento guiado pelos testes	O XP é destinado à construção de sistemas com alta qualidade, o que leva à necessidade de diversos mecanismos de validação para assegurar que o software está correto. Um destes mecanismos é a programação em par, tal como foi citado anteriormente. Além dela, o XP também utiliza a técnica de desenvolvimento guiado pelos testes. Os desenvolvedores escrevem testes para cada funcionalidade

	antes de codificá-las. Fazendo isso, eles aprofundam o entendimento das necessidades do cliente (o que aprimora a análise), se preocupam com as interfaces externas dos métodos e classes antes de codificá-los (o que melhora o design), sabem até onde codificar cada funcionalidade (o que ajuda a manter o sistema simples) e passam a contar com uma massa de testes que pode ser usada a qualquer momento para validar todo o sistema.
Refactoring	Para que o sistema possa evoluir de forma incremental, a equipe deve fazer com que ele expresse os seus objetivos facilmente e esteja sempre claro e fácil de compreender. Freqüentemente, isso levará a equipe a modificar partes do sistema que estejam funcionando para facilitar a sua manutenção. O refactoring é o ato de alterar um código sem afetar a funcionalidade que ele implementa. É utilizado para tornar o software mais simples de ser manipulado e se utiliza fortemente dos testes descritos anteriormente para garantir que as modificações não interrompam o seu funcionamento.
Código coletivo	No XP o sistema não é segmentado em partes, de modo que cada desenvolvedor fique responsável por uma delas. Os desenvolvedores têm acesso a todas as partes do código e podem alterar aquilo que julgarem importante sem a necessidade de pedir autorização de outra pessoa, pois o código é coletivo. Isso fornece maior agilidade ao processo e cria mais um mecanismo de revisão e verificação do código, já que aquilo que é escrito por um par hoje, acaba sendo manipulado por outro amanhã. Se alguma coisa estiver confusa no código, o par deverá fazer refactoring para torná-lo mais legível.
Código padronizado	Para que todos os desenvolvedores possam manipular qualquer parte do software de forma mais rápida, a equipe estabelece padrões de codificação, que servem também para tornar o sistema mais homogêneo e permitir que qualquer manutenção futura seja efetuada mais rapidamente.
Design simples	Para que o cliente possa obter feedback logo, a equipe precisa ser ágil no desenvolvimento, o que a leva a optar pela simplicidade do design. Ao invés de criar generalizações dentro do código, de modo a prepará-lo para possíveis necessidades futuras, a equipe deve sempre optar por um código que seja suficiente para atender às necessidades da funcionalidade que está implementando. Os desenvolvedores se baseiam na premissa de que serão capazes de incorporar qualquer necessidade futura quando e se ela surgir. Para isso, eles contam com o refactoring, os testes e as demais práticas do XP para apoiá-los.
Metáfora	Para facilitar a criação de um design simples, a equipe de desenvolvimento utiliza metáforas, já que elas têm o poder de transmitir idéias complexas de forma simples, através de uma linguagem comum que é estabelecida entre a equipe de desenvolvimento e o cliente.
Ritmo sustentável	A qualidade do design, do código, das metáforas e do sistema é determinada diretamente pela qualidade dos desenvolvedores e a capacidade que eles têm de se manter atentos, criativos e dispostos a solucionar problemas. Para garantir que a equipe tenha sempre o máximo de rendimento e produza software com

	melhor qualidade possível, o XP recomenda que os desenvolvedores trabalhem apenas oito horas por dia e evitem fazer horas-extras, visto que é essencial estar descansado a cada manhã, de modo a utilizar a mente na sua plenitude ao longo do dia.
Integração contínua	Quando uma nova funcionalidade é incorporada ao sistema, ela pode afetar outras que já estavam implementadas. Para assegurar que todo o sistema esteja sempre funcionando de forma harmoniosa, a equipe pratica a integração contínua que leva os pares a integrarem seus códigos com o restante do sistema diversas vezes ao dia. Cada vez que um par faz isso, ele executa todos os testes para assegurar que a integração tenha ocorrido de forma satisfatória. Uma integração sempre pode produzir erros no sistema. Sendo assim, a equipe utiliza os testes para descobrir eventuais defeitos o mais rapidamente possível já que descobri-los logo facilita e acelera a correção e diminui a probabilidade de pequenos problemas se transformarem em grandes dores de cabeça no futuro.
Realises curtos	Como explicado anteriormente, o XP tem como objetivo gerar um fluxo contínuo de valor para o cliente. Sendo assim, ele trabalha com releases curtos, ou seja, a equipe produz um conjunto reduzido de funcionalidades e coloca em produção rapidamente de modo que o cliente já possa utilizar o software no dia-a-dia e se beneficiar dele. Durante todo o projeto, a equipe colocará o sistema em produção diversas vezes, cada vez incorporando mais funcionalidades e gerando mais valor.

Fonte: Teles, 2009, adaptação do pesquisador.

Uma equipe que utiliza o XP geralmente é composta por pessoas que representam os seguintes papéis (TELES, 2009):

- Gerente de projeto;
- Coach;
- Analista de teste;
- Redator técnico;
- Desenvolvedor.

O quadro abaixo apresenta as características de cada uma das características da equipe.

Quadro 7: Características da equipe

Função	**Descrição**
Gerente de projeto	O gerente de projeto é responsável pelos assuntos administrativos do projeto. Ele procura liberar a equipe de questões que não estejam diretamente ligadas à atividade diária de desenvolvimento. Além disso, administra o relacionamento com o cliente assegurando que o mesmo participe ativamente do

	desenvolvimento e forneça as informações essenciais para que a equipe possa implementar o sistema desejado.
Coach	O coach é o responsável técnico do projeto. O XP recomenda que um profissional tecnicamente bem preparado seja destacado para orientar a equipe de modo que ela siga as boas práticas recomendadas pelo XP. Embora também possa atuar na implementação do sistema, sua tarefa principal é assegurar o bom funcionamento do processo e buscar formas de melhorá-lo continuamente.
Analista de teste	O analista de teste é responsável por ajudar o cliente a escrever os testes de aceitação. Quando estes testes não são automatizados, o analista de teste deve fazer com que eles sejam executados diversas vezes ao longo das iterações do projeto. Ele procura fazer com que os eventuais defeitos do sistema sejam identificados tão logo apareçam. Desta forma, fornece feedback para a equipe rapidamente, de modo que ela possa fazer as correções com rapidez e evitar que os problemas se propaguem.
Redator técnico	O redator técnico ajuda a equipe de desenvolvimento a documentar o sistema. A sua presença permite que os desenvolvedores se concentrem prioritariamente na implementação do software. Embora eles possam continuar fazendo algumas documentações, o redator técnico é quem faz a maior parte do trabalho de documentação.
Desenvolvedor	O desenvolvedor é a pessoa que analisa, projeta e codifica o sistema. Em suma, é a pessoa que efetivamente constrói o software. Dentro do XP, não existem divisões entre analista, projetista, programador etc. Cada desenvolvedor exerce estes diferentes papéis em diversos momentos do projeto.

Fonte: Teles, 2009, adaptação do pesquisador.

No que se refere ao processo do XP, como demonstra a figura abaixo existem duas grandes fases: Exploração e Manutenção; Na fase de Exploração, as Estórias são ditas e documentadas e a arquitetura é desenvolvida. Na fase de Manutenção, ocorre todo o processo de desenvolvimento (programação) em uma série de iterações até a entrega do produto.

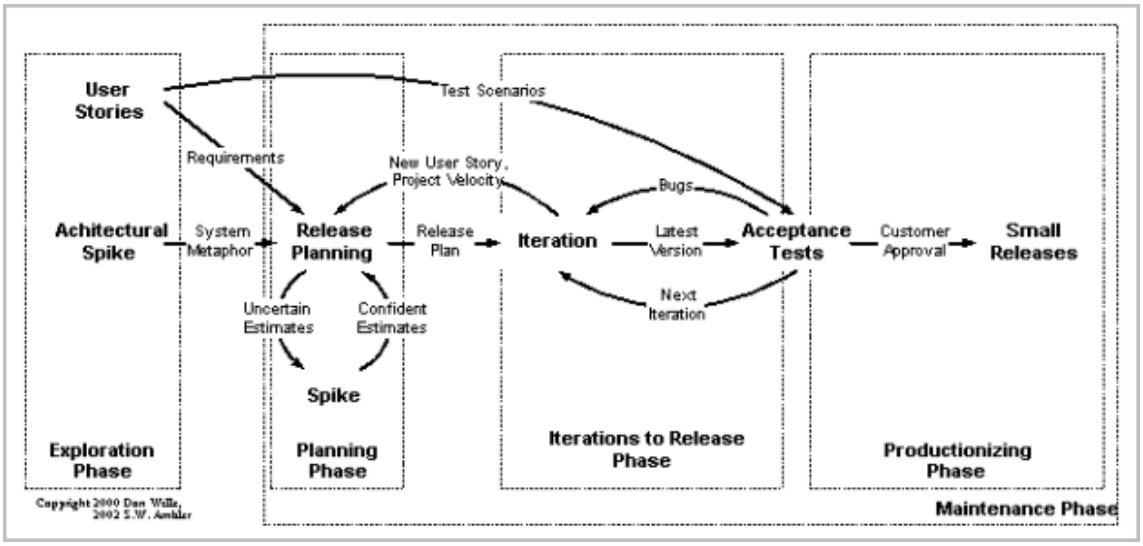

Figura 4: Processo do XP

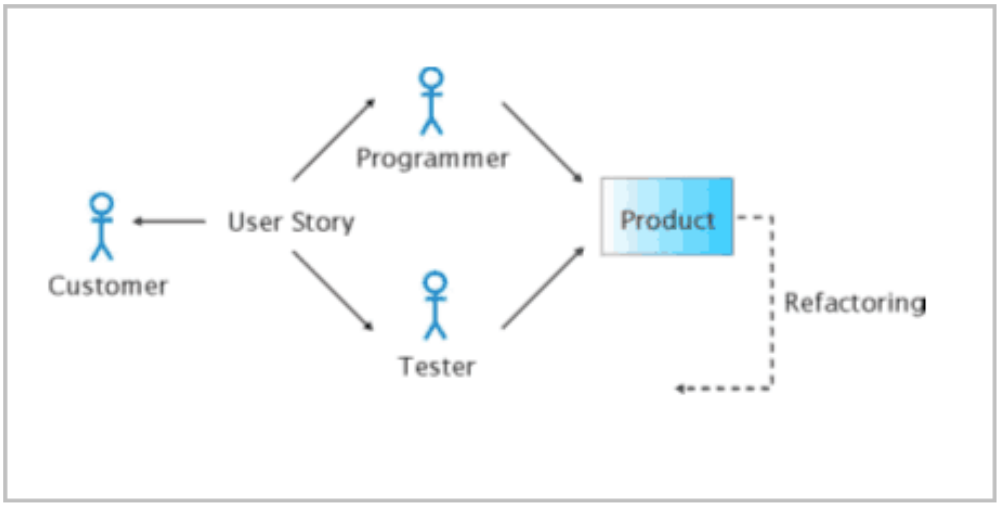

Figura 5: Iteração com o XP

Os passos gerais nas iterações do XP são: planejamento, projeto, codificação e testes. Estes quatros passos serão alternados constantemente entre eles.

3.4.6 Comparativo dos Métodos Ágeis

Strode (2005), em seu estudo apresenta um comparativo entre as seguintes metodologias DSDM, XP, Scrum e Crystal (vide quadro abaixo).

Quadro 8: Técnicas e metodologias

Técnica	DSDM	XP	Scrum	Crystal
Desenvolvimento Concorrente			x	x
Desenvolvimento Iterativo	x	x	x	x
Desenvolvimento Incremental	x	x	x	x
Protótipo Evolucionário	x			
Produtos de Softwares com releases pequenos		x		
Testes antes do desenvolvimento		x		
Integração diária do sistema completo			x	
Testes automáticos de regressão				x
Refatoração do código		x		
Testes através das iterações	x	x	x	
Programação em par		x		
Equipe trabalhando na mesma sala ou andar		x	x	x
Código de acordo com um padrão		x		
Código coletivo		x		
Daily team meetings	x		x	
Iteration planning meeting		x	x	
Planning game		x		
Reflective workshops				x
User stories		x		
Metáforas		x		
Design mais simples possível		x		
Product Backlog			x	
Sprint Backlog			x	
Release Backlog			x	
Milestones para verificar desempenho				x
Regras MOSCOW	x			

Fonte: Strode, 2005.

3.5 LIMITAÇÕES NO EMPREGO DOS MÉTODOS ÁGEIS

O trabalho de Dias (2005, p. 88-89) agrupa o estudo e descrição de um

conjunto de limitações para o emprego das metodologias ágeis. Dentre as limitações pesquisadas por ela, uma das maiores dificuldades para a adoção de metodologias ágeis no desenvolvimento de software está relacionada ao suporte a equipes distribuídas, pois a comunicação é facilitada quando a equipe de desenvolvimento e o cliente estão próximos. Quando o trabalho é realizado em localidades diferentes, o uso de documentação formal passa a ser necessário.

Outra limitação encontrada diz respeito ao tamanho da equipe de trabalho. Quando a equipe é muito grande, no caso de projetos grandes e complexos, tende-se a aumentar também a complexidade da comunicação, tornando o processo menos ágil. Portanto, o tamanho da equipe acaba por restringir a eficiência da comunicação face a face, tendendo também ao aumento de documentação.

A questão do reuso de artefatos também é limitante para o uso dos métodos ágeis, visto que a premissa de documentação mínima dificulta a reutilização dos artefatos que tem como foco o desenvolvimento de soluções para problemas muitos específicos e não de códigos genéricos e reutilizáveis.

Outro ponto negativo para a utilização dos métodos ágeis é sua adoção no desenvolvimento de projetos de Sistemas Críticos. Como estes sistemas exigem que todos os seus componentes sejam exaustivamente testados e projetados de forma a não haver falhas que impossibilitem seu uso seguro, os pressupostos relacionados à documentação e garantia de qualidade dos métodos ágeis não são válidos. Aqui é necessária a especificação formal, os testes intensivos e demais técnicas de análise dos métodos clássicos de desenvolvimento de software. Esta é uma das razões para que os métodos ágeis não se sobreponham totalmente ao desenvolvimento clássico de software, pois os sistemas críticos devem passar por um rígido controle de qualidade.

E como item final, as metodologias ágeis não são indicadas para softwares com milhões de linhas de código e alto grau de interação entre os componentes, pois estes projetos requerem grande esforço de gerenciamento e controle, além de processos estruturados e formais que garantam o entendimento do software.

3.6 DESVANTAGENS DOS MÉTODOS ÁGEIS

Uma desvantagem apontada aos métodos ágeis é o fato de estes não serem escaláveis. Estes métodos não foram desenhados para projetos muito longos, existindo contudo a abordagem *Scrum*.

Outra desvantagem dos métodos ágeis é um menor controle de custos e prazos. Nestas metodologias o projeto termina quando o cliente não levantar mais funcionalidades relevantes que deseje ver concretizadas, em oposição a ser acordado um preço e um plano. Com isso, os custos e prazos (durações) podem variar e podem ser de difícil gestão para o gestor do projeto, além de causarem um certo risco na visão do cliente (TOMAS, 2009).

3.7 VANTAGENS DOS MÉTODOS ÁGEIS

As entregas constantes constituem a principal vantagem da utilização das metodologias ágeis nos projetos de desenvolvimento de software. Outras vantagens são a integração e o teste contínuo, pois possibilitam a melhora na qualidade do software, não sendo mais necessário existir uma fase de integração de módulos, uma vez que eles são continuamente integrados e eventuais problemas são identificados e resolvidos constantemente devido aos testes contínuos.

Uma outra vantagem das metodologias ágeis está centrada no aumento do controle por parte dos gestores, uma vez que se baseia no que está realmente sendo produzido e no que vai ser feito a curto prazo. Desta forma, há mais visibilidade e adequação das medições e avaliações do estado das funcionalidades e tarefas realizadas (TOMAS, 2009). Este método aproxima os desenvolvedores e gestores pois existe uma maior e melhor comunicação. É especialmente adequado para projetos direcionados para a Web, onde os requisitos vão evoluindo e não se exigem muitos trabalhadores.

Como as equipes ágeis são pequenas, os membros da equipe não competem, pois geralmente devem ter habilidades e atribuições complementares entre si. As equipes ágeis tende a se empenham juntas nas atividades como definições de trabalho, tomadas de decisão, apresentação para os clientes e retrospectivas. Ainda que a maioria desse trabalho conjunto possa potencialmente incentivar um clima de competição, a abordagem ágil favorece o que diversos autores chamam de "conflito produtivo" – em que as equipes têm reuniões interessantes e animadas, extraem e exploram ideias de todos os membros da equipe e minimizam questões políticas.

Outros pontos fortes dos métodos ágeis são o desenvolvimento de comprometimento e responsabilidade. Práticas ágeis recomendam fortemente que todas as coisas sejam dirigidas e estejam sob controle da equipe, e o progresso e o sucesso podem ser meramente medidos pelo grau de envolvimento que tenha sido experimentado pela equipe. E como o trabalho de cada membro da equipe fica visível, cria-se uma responsabilização pessoal pelo trabalho em equipe e não por pressão superior.

4 GERENCIAMENTO ÁGIL DE PROJETOS

A gerência de projetos de *software* não é sempre tão fácil como parece. As técnicas de gerência tradicionais de projeto são incapazes de adaptar-se bem às mudanças e aos novos riscos. Isto faz com que as equipes tenham cada vez mais dificuldades para reagirem rapidamente às mudanças intrínsecas ao processo de desenvolvimento de *software*.

Para isto, as novas técnicas com métodos ágeis podem tornar mais fácil a gestão de gerentes de projeto e equipes, desde que executadas corretamente. Diferentes tipos de problemas e desafios possuem características diferenciadas que requerem diferentes tipos de abordagens. O maior desafio está em selecionar, adaptar e integrar estas abordagens, de acordo com as características presentes em um determinado ambiente.

Os projetos de desenvolvimento de *software* possuem duas vertentes: uma técnica e outra gerencial, sendo que durante muitos anos a atenção voltou-se para o aprimoramento de modelos de desenvolvimento (ênfase técnica), ficando o componente gerencial relegado ao segundo plano.

Alguns autores, como Highsmith apud Dias (2005), mencionam que o gerenciamento clássico de projetos não se mostrou plenamente efetivo para projetos de desenvolvimento de *software*, devido a estes projetos estarem inseridos em ambientes extremamente dinâmicos, sujeitos a mudanças constantes, fugindo assim aos padrões do gerenciamento clássico de projetos.

Desta forma, a alternativa de gerenciamento deu-se através do Gerenciamento Ágil de Projetos. Highsmith apud Dias (2005) indica a utilização do gerenciamento ágil para projetos de software que requerem inovação e criatividade ou que estejam sujeitos a constantes alterações de requisitos. E em determinadas situações, uma combinação do enfoque clássico com as práticas propostas pelo gerenciamento ágil é muito apropriada para o alcance de resultados mais efetivos.

4.1 O QUE É GERENCIAMENTO ÁGIL DE PROJETOS?

O movimento para o desenvolvimento ágil de *software* estendeu-se para o gerenciamento de projetos, dando origem ao Gerenciamento Ágil de Projetos com base nos mesmos valores e princípios dos métodos ágeis. Tratando-se de uma nova plataforma de gerenciamento de projetos, o Gerenciamento Ágil é um modelo aplicável a ambientes voláteis e desafiadores, sujeitos a mudanças frequentes, onde o processo padronizado (gerenciamento clássico) não é mais adequado (DIAS, 2005).

O gerenciamento ágil busca o desenvolvimento da visão do futuro e da capacidade de exploração, desfazendo-se da postura clássica de gerenciamento, calcada em planejamento prévio de ações e atividades. Neste sentido, o Gerenciamento Ágil de Projetos assume o papel de resposta em âmbito gerencial às pressões por inovação, concorrência acirrada, necessidade de redução de ciclos de desenvolvimento, implantação de novos produtos e serviços e à necessidade de adaptação a um ambiente de negócio bastante dinâmico (DIAS, 2005).

Os valores desse novo modelo de gerenciamento foram citados por Dias (2005, p.96):

- As respostas às mudanças são mais importantes que o surgimento de um plano;
- A entrega de produtos está acima da entrega de documentação;
- Priorização da colaboração do cliente sobre a negociação de contratos;
- Os indivíduos e às interações são mais importantes que os processos e as ferramentas.

4.2 CARACTERÍSTICAS DOS PROJETOS PARA UTILIZAÇÃO DO GERENCIAMENTO ÁGIL

O fato dos métodos ágeis de desenvolvimento e do gerenciamento ágil de

projetos terem a mesma origem e estarem construídos sobre os mesmos princípios, sugere que, possivelmente, o Gerenciamento Ágil de Projetos seja mais indicado para o gerenciamento de projetos de software realizados com o uso dos métodos ágeis. Entretanto, não foram encontradas evidências que comprovem ou reneguem tal afirmação na revisão bibliográfica. Apenas é consenso entre os autores do assunto, que deve ser realizada uma análise do ambiente interno (aspectos organizacionais e culturais) e do contexto externo no qual o projeto está inserido, para que se selecione o modelo de gerenciamento de projetos (clássico ou ágil) e se defina o método de desenvolvimento (clássico ou ágil).

O consenso entre os autores da área é de que em projetos em que há muitas mudanças, em que os requisitos são passíveis de alterações, onde refazer partes do código não é uma atividade que apresenta alto custo, as equipes são pequenas, as datas de entrega do software são curtas e o desenvolvimento rápido é fundamental, não podem haver requisitos estáticos, necessitando então de metodologias ágeis. Para projetos que se encaixam nestas características, os processos de desenvolvimento e gerenciamento devem ser flexíveis para serem adaptados a diversas situações. Isso para que ao final, seja feita a entrega de um produto com qualidade, validado e de acordo com o que foi solicitado.

4.3 COMPARAÇÃO ENTRE GESTÃO CLÁSSICA E GESTÃO ÁGIL DE PROJETOS

O quadro abaixo exibe uma comparação entre a gestão clássica e a gestão ágil de projetos.

Quadro 9: Comparação gestão clássica e gestão ágil

	Gestão Clássica	**Gestão Ágil**
Objetivos	Prazo, Custo e Qualidade	Prazo, Custo, Qualidade e Capacidade de Transformação
Funções do Gerenciamento	Planejamento, Controle e Tomada de Decisão	Suporte
Fundamento Teórico	Mecanicismo: Divisão do Trabalho, Especialização e Controle	Projeto como um Sistema Complexo Adaptativo

Meio Ambiente	Relativamente Estável e Previsível	Turbulento e Imprevisível
Estrutura Formal	Centralizada e Hierárquica	Descentralizada: Times Pequenos e Semiautônomos
Acesso às Informações	Restrito: Gestores possuem as Informações	Aberto: Informação é Domínio Comum
Controle	Restrito e Centralizado	Baixo e Descentralizado
Responsabilidade	Centralizada no Gerente	Descentralizada nos Times Semiautônomos
Aprendizado	Restrito	Contínuo e Adaptativo
Capacidade de Adaptação	Baixa	Elevada

Fonte: Neumann, Baureis, Stock, 2009, p. 65.

O gerenciamento clássico de projetos está baseado no planejamento detalhado do projeto e nos processos formais de monitoramento e controle. Em contrapartida, o gerenciamento ágil enfatiza a execução, objetivando uma entrega mais rápida para o cliente e resultados durante todo o projeto (DIAS, 2005).

4.4 FATORES CRÍTICOS DE SUCESSO PARA O GERENCIAMENTO ÁGIL

A aplicação do gerenciamento ágil não é trivial. Entre seus pré-requisitos mais importantes encontra-se a mudança cultural da organização. Do lado gerencial, agilidade implica na abstenção do gerente em decisões do time. Do lado dos membros, assumir a responsabilidade pelas decisões tomadas é imprescindível. A estrutura organizacional passa a ser determinante e estruturas mecanicistas tornam-se inadequadas. O cliente passa a integrar o time sendo que contratos perdem em significado. É preferível a evolução iterativa do projeto com a participação ativa do cliente, a longas fases baseadas em especificações extremamente detalhadas (NEUMANN, BAUREIS, STOCK, 2009).

5 CONCLUSÃO

Com a elaboração deste estudo concluiu-se que as metodologias ágeis buscam estabelecer novos paradigmas para o desenvolvimento de *software*, onde o desenvolvimento ágil do *software* é o foco principal. As metodologias ágeis visam à simplicidade na implementação de um projeto, pois suas regras são simples de serem seguidas. E o sucesso na adoção de um modelo ágil está diretamente relacionado ao comprometimento do time, pois sua forma de condução dos trabalhos possui um elevado grau de informalidade.

O discurso trazido pelas metodologias ágeis é leve e visa uma redução drástica da complexidade dos projetos de desenvolvimento de *software*, bem como de toda a estrutura da tecnologia da informação, reduzindo em poucos elementos gerenciáveis e trabalhando com profissionais cuja atuação é multidisciplinar.

Um dos principais problemas das metodologias ágeis é que o cliente deve estar presente durante o processo de desenvolvimento, tornando o compromisso do usuário uma peça fundamental para o sucesso do projeto. Em contrapartida, as metodologias ágeis possuem várias vantagens em relação aos métodos tradicionais de desenvolvimento de *software*, sendo o maior ponto positivo as entregas constantes do *software*. Com entregas mais rápidas, o cliente não precisa esperar muito para ver o software funcionando, o que agrega transparência sobre o progresso e o valor entregue, reduzindo também as possibilidades de que grandes falhas aconteçam.

Em âmbito gerencial, os métodos tradicionais de gestão para projetos de desenvolvimento de software não têm se adaptado aos ambientes extremamente dinâmicos. Para isto, os métodos ágeis de gestão tornaram mais fácil esta gestão, calcados em respostas rápidas a mudanças constantes e entregas do produto acima de documentação, processos e contratos.

Este trabalho permitiu concluir que o Gerenciamento Ágil de Projetos

rapidamente alcançará outros setores além do Desenvolvimento de *Software*. E para isto, o grande desafio das metodologias ágeis está no princípio de encontrar maneiras de eliminar alguns de seus pontos fracos, como por exemplo, a falta de análise de riscos, sem torná-las metodologias pesadas ao mesmo tempo. Outro desafio, é como implementar o uso dessas metodologias ágeis em grandes empresas e equipes, uma vez que normalmente essas metodologias são focadas em equipes pequenas.

A grande lição aprendida neste trabalho é que o Projeto Ágil tem a capacidade de aprender e de se adaptar com seu aprendizado.

6 REFERÊNCIAS BIBLIOGRÁFICAS

AGILE ALLIANCE: MANIFESTO ÁGIL. Estados Unidos, 2001. Disponível em: <http://agilemanifesto.org/>. Acesso em: março, 2010.

BEEDLE, MA; SCHWABER, K. Agile Software Development With Scrum. 1ª ed. Prentice Hall PTR, 2002.

BERNARDINO, CB; GOMES, AS; VASCONCELOS, A. Design interativo em processos de desenvolvimento de software. Recife: Universidade Federal de Pernambuco, 2005.

CÂMARA, F. Um cardápio de metodologias ágeis. Revista Visão Ágil. Edição 2, p. 19-21, 2008.

DAVIS, MM; AQUILANO, NJ; CHASE, RB. Fundamentos da administração da produção. 3ª edição. Porto Alegre: Bookman, 2001.

DIAS, MVB. Um novo enfoque para o gerenciamento de projetos de desenvolvimento de software. São Paulo: Dissertação de Mestrado – Universidade de São Paulo, 2005.

ENGENHARIA DE SOFTWARE. Disponível em: <http://engenhariasoftware.wordpress.com/2009/08/03/gestao-de-projetos-de-software-ou-gestao-em-projetos-de-software/>. Acesso em: dezembro, 2009.

FELSING, J; PALMER, S. A Practical Guide to Feature-Driven Development. 1ª ed. Prentice Hall PTR, 2002.

GERMANO, VH. Nove pecados mortais no planejamento de projetos. Revista Visão Ágil. Edição 4, p.27-30, 2008.

GIDO, J; CLEMENTS, JP. Gestão de projetos. São Paulo: Thomson, 2007.

KERZNER, H. In Seach of Excellence in Project Management: successful practies in high performance orgainzations. New York: Van Nostrand Reinhold, 1998.

MATTOS, JL; GUIMARÃES, LS. Gestão da tecnologia e inovação: uma abordagem prática. São Paulo: Saraiva, 2005.

MOL, AC. Uma discussão sobre os métodos ágeis de desenvolvimento de software. Minas Gerais: Monografia de Final de Curso – Universidade Federal de Minas Gerais, 2007.

NEUMANN, D; BAUREIS, D; STOCK, T. Capacidade de Transformação: Gestão Ágil de Projetos em Estruturas Organizacionais Transformáveis. Revista Mundo Project Management. Ano 5, número 26, p. 65, abril/maio 2009.

PMBOK GUIDE 3ª Edição. Um Guia do Conjunto de Conhecimentos em Gerenciamento de Projetos. 3ª edição, 2004.

SCHWABER, K. Agile Project Management With Scrum. 1ª ed. Microsoft Press, 2004.

STRODE, DE. Agile Project. 2005. Disponível em <http://muir.massey.ac.nz/bitstream/10179/515/1/02whole.pdf>. Acesso em março, 2010.

TELES, VM. Extreme Programming. 1ª ed. São Paulo: Novatec, 2009.

TOMAS, MRS. Métodos Ágeis: Características, pontos fortes e fracos e possibilidades de aplicação. Portugal. Universidade Nova de Lisboa, 2009. Disponível em: <http://dspace.fct.unl.pt/bitstream/10362/2003/1/WPSeries_09_2009Tomas.pdf>

TURBAN, E; MCLEAN, E; WETHERBE, J. Tecnologia da informação para gestão: transformando os negócios na economia digital. Tradução: Renate Schinke. Porto Alegre: Bookman, 2007.

VALE, A. Aperfeiçoamento de projetos ágeis: Parte I: uma visão geral. Revista Visão Ágil. Edição 3, p. 29-34, 2008a.

VALE, A. Aperfeiçoamento de projetos ágeis: Parte II: as três perspectivas. Revista Visão Ágil. Edição 4, p. 18-24, 2008b.

VARGAS, R. Gerenciamento de projetos: estabelecendo diferenciais competitivos. Rio de Janeiro: Brasport, 2009.

WIKIPEDIA: A enciclopédia livre. Disponível em: <http://pt.wikipedia.org/wiki/Ger%C3%AAncia_de_projetos>. Acesso em: janeiro, 2010.