UNIVERSIDADE PRESBITERIANA MACKENZIE

BRUNO PEREIRA GUERRA

METODOLOGIAS ÁGEIS NO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE: ALTERNATIVAS PARA AUMENTAR O ÍNDICE DE SUCESSO EM PROJETOS

São Paulo

2013

BRUNO PEREIRA GUERRA

METODOLOGIAS ÁGEIS NO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE: ALTERNATIVAS PARA AUMENTAR O ÍNDICE DE SUCESSO EM PROJETOS

Trabalho de Conclusão de Curso apresentado ao Programa de Pós-graduação Lato Sensu da Universidade Presbiteriana Mackenzie, como requisito para a obtenção do Título de Especialista em Gestão de Projetos.

São Paulo
2013

Dedico este trabalho, sobretudo, à família,
pela compreensão aos momentos de ausência.
E também à minha esposa, especialmente
pelo esforço dedicado na revisão ortográfica.

Você obtém o que você mede e mais nada.
(Robert Austin)

RESUMO

Trata-se de uma pesquisa sobre Metodologias Ágeis aplicada ao processo de Engenharia de Desenvolvimento de *Software*, com enfoque no *framework* conhecido como *Scrum*, cuja abordagem é uma alternativa às Metodologias Tradicionais e visa a obtenção de resultados mais satisfatórios em atendimento aos indicadores. São eles: 1) Escopo (o que é entregue em um projeto de *software*); 2) Tempo (o prazo necessário para que o escopo desejado seja entregue com percepção de benefício para o solicitante); 3) Custo (o investimento necessário para que o escopo desejado seja entregue no tempo acordado). Também será apresentada uma consolidação de pesquisas e dados estatísticos dos principais órgãos de referência no assunto Gestão de Projetos (PMI®) e Pesquisa em Desenvolvimento de Software (*The Standish Group*), os quais indicam Metodologias Ágeis como alternativa plausível para obtenção de resultados mais assertivos.

Palavras-chave: Gestão de Projetos; Metodologias Ágeis; *Scrum*

ABSTRACT

This is a research on Agile Methodologies applied to process engineering Software Development, focusing on framework known as Scrum, whose approach is an alternative to traditional methodologies and aims to obtain more satisfactory results in compliance indicators. They are: 1) Scope (what is delivered in a software project), 2) time (the time required for the desired scope is delivered with perceived benefit to the applicant); 3) Cost (investment required for the desired scope is delivered at the agreed time). Also presented will be a consolidation of research and statistical data of the main organs of reference on the subject Project Management (PMI ®) and Research in Software Development (The Standish Group), which indicate Agile as plausible alternative for achieving more assertive results.

Keywords: Project Management, Agile, Scrum

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS, SIGLAS E SÍMBOLOS

ASD	*Adaptive Software Development*
Capes	Coordenação de Aperfeiçoamento de Pessoal de Nível Superior
DSDM	*Dynamic Systems Development Method*
FDD	*Feature-Driven Development*
GP	Gestão de Projetos
IEEE	Instituto de Engenheiros Eletricistas e Eletrônicos
PMBOK	*Project Management Body of Knowledge*
PMI	*Project Management Institute*
RAD	*Rapid Application Development*
RUP	*Rational Unified Process*
TI	Tecnologia da Informação
UML	*Unified Model Language*
XP	*Extreme Programming*

SUMÁRIO

1. INTRODUÇÃO

1.1. OBJETIVOS

1.1.1 Objetivo Geral

A pesquisa limita-se ao tema Metodologias Ágeis no que tange ao processo de engenharia de desenvolvimento de *software*, com destaque para a técnica *Scrum*.

O problema levantado é o alto grau de insucesso em projetos de desenvolvimento de *software*. As Metodologias Ágeis são tratadas nesse estudo como a alternativa para a obtenção de melhores resultados.

Portanto, o objetivo geral é apresentar Metodologias Ágeis, sobretudo o *Scrum*, para obtenção de resultados mais assertivos, como alternativa às Metodologias Tradicionais no processo de desenvolvimento de *software*.

1.1.2 Objetivos Específicos

Os objetivos específicos que pretende-se alcançar com a pesquisa são:

- Definir a técnica proposta;
- Entender seus objetivos;
- Pesquisar para que foi criada;
- Pesquisar quem a utiliza;
- Pesquisar seus resultados;
- Definir seus pontos fortes;
- Definir seus pontos fracos;
- Analisar criticamente as opiniões dos criadores;
- Analisar criticamente as opiniões dos disseminadores;
- Analisar criticamente as opiniões dos principais utilizadores;
- Levantar pesquisas sobre insucesso em projetos de forma geral e projetos de desenvolvimento de *software*.

1.2 JUSTIFICATIVA

1.2.1 Problematização e Diagnóstico da situação

Quando se trata da área de Tecnologia da Informação (TI) é bastante comum encontrar profissionais da área com a sensação de "estar devendo", de estar aquém do esperado, ou correndo atrás do prejuízo em seus campos de atuação independentemente de sua competência, capacidade e motivação momentânea. Esta sensação também independe do tipo ou ordem do relacionamento, seja de um funcionário para com seu gestor ou do gestor para com seus funcionários, do empreendedor para com seus possíveis clientes ou dos consumidores para com o que esperam em termos de inovações, da sociedade para com os prestadores de serviços em geral, do indivíduo para com sua família e amigos, entre tantos outros tipos de relações possíveis.

Em desenvolvimento de *software* a situação não é muito diferente. Apesar de inúmeras técnicas, metodologias, formas de fazer e abordagens diferentes, o que os números apresentam é um resultado maior de "não sucesso" do que "sucesso", como por exemplo, os números divulgados pela *The Standish Group* em seu *Chaos Report*, os quais mostram que em 2009 24% dos projetos falharam, o que quer dizer que foram cancelados ou nunca usados, sendo que 44% mudaram, ou seja, não atenderam a pelo menos um dos três pilares fundamentais de um projeto: Escopo, Prazo e Custo, e que 32% foram concluídos com sucesso. Observando-se outros aspectos menos tangíveis e mais abstratos, como a satisfação do cliente, as aspirações pessoais e/ou profissionais planejadas frente as concretizadas também se chega a resultados relevantes para o "não sucesso". (2009 *apud* SOUZA, 2010).

Conceituando complexidade, em uma escala de quatro níveis (na ordem: simples, complicado, complexo e anárquico) onde na complexidade simples, a tecnologia é mais conhecida e os requisitos são mais certos até a complexidade anárquica em que a tecnologia é menos conhecida e os requisitos são menos certos (ou mais incertos). Para a questão do Desenvolvimento de *Software*, um forte candidato a principal problema é o nível de complexidade assumida para os projetos, em que a maioria são na realidade complexos e tratados ou abordados como se fossem simples. (SCHWABER, 2004).

1.2.2 Limites da pesquisa

A pesquisa se limita ao tema Metodologias Ágeis para desenvolvimento de *software*, com um foco maior para *Scrum*. O objetivo é se aprofundar na técnica proposta, entender seus objetivos, para que foi criada, quem utiliza, pesquisar seus resultados, refletir sobre seus pontos fortes e fracos, analisar criticamente as opiniões dos criadores, disseminadores e principais utilizadores e, por fim, correlacionar com o problema levantado que é o alto grau de insucesso em projetos de desenvolvimento de *software*.

1.2.3 Sobre o tema

Apesar de ser um tema bastante atual e estar em voga, o *Scrum* já vem sendo utilizado há mais de duas décadas, desde o início dos anos 90. (DALONSO, 2010).

É uma abordagem empírica que aplica algumas ideias da teoria de controle de processos industriais para o desenvolvimento de *softwares*, reintroduzindo as ideias de flexibilidade, adaptabilidade e produtividade. (SOARES,2004).

Metodologias Ágeis se opõem ao modelo cascata (do inglês, *waterfall model*) que deriva de outras engenharias tradicionais (Civil, Elétrica, Naval etc) e foi a primeira a ser utilizada em Engenharia de *Software*, na década de 70. Este modelo que adota a visão sequencial de tarefas é recomendado apenas em situações em que os requisitos do *software* são estáveis e os requisitos futuros são previsíveis. (SOARES, 2004).

Em resumo, de um modo geral, Metodologias Ágeis podem ser definidas pelos itens abaixo:

A) Principais conceitos:
   - Iterativa (iterações são chamadas de *sprints*) e incremental;
   - Reuniões de acompanhamento diárias;
   - Não são orientadas à documentação;
   - Enfoque nas pessoas;
   - Adaptativas ao invés de preditivas;
   - Enfoca comunicação;
   - Tornou-se popular em 2001 com a criação do Manifesto Ágil.

B) O marco de início foi o Manifesto Ágil (BECK et al, 2001), com os seguintes conceitos chaves:
   - Indivíduos e interações ao invés de processos e ferramentas;
   - *Software* executável ao invés de documentação;
   - Colaboração do cliente ao invés de negociação de contratos;
   - Respostas rápidas a mudanças ao invés de seguir planos.

C) São divididas em três fases principais (SOARES, 2004):
   - Pré-planejamento;
   - Desenvolvimento;
   - Pós-planejamento.

D) Principais diferenças frente a outras metodologias (SOARES, 2004):
   - Feedback constante;
   - Abordagem incremental;
   - Comunicação entre pessoas é encorajada.

1.3 METODOLOGIA

O trabalho monográfico propõe-se a um estudo estritamente teórico, com base nos seguintes autores (entre outros): Jeff Sutherland; Ken Schwaber; Kent Beck; PMI®.

Segundo o Capes (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior), as áreas de conhecimento envolvidas estão divididas em duas áreas de avaliação (A e B), conforme abaixo (CAPES, 2012):

A) Matemática / Probabilidade e Estatística:
   - Análise de Dados;
   - Probabilidade e Estatística Aplicadas.

B) Ciência da Computação:
   - Análise de Algoritmos e Complexidade de Computação
   - Metodologia e Técnicas da Computação;
   - Engenharia de *Software*;
   - Sistemas de Informação;
   - Sistemas de Computação;
   - Arquitetura de Sistemas de Computação.

1.4 ESTRUTURA DO TRABALHO

Este trabalho está estruturado em cinco capítulos, cujos objetivos estão resumidos a seguir:

– Capítulo 1, Introdução: este capítulo delimita os objetivos, geral e específicos, do trabalho como um todo. É explicado o problema, motivador da pesquisa pelo assunto, e quais os limites de abrangência do trabalho. É apresentado brevemente também o assunto principal, qual metodologia utilizada e como é estruturada a monografia;

– Capítulo 2, Acerca de um Projeto de *Software*: um capítulo com definições teóricas básicas sobre os assuntos relacionados a um projeto de *software* e o seu gerenciamento;

– Capítulo 3, Metodologias de Desenvolvimento de *Software*: outro capítulo com referencial teórico, porém, esse voltado para o problema (baixo índice de sucesso em projetos de *software*) motivador do trabalho, detalhando as metodologias, da Abordagem Tradicional para desenvolvimento de *software*, que são teoricamente responsáveis indiretas pelo problema e também as metodologias da Abordagem Ágil, que são propostas como alternativas para, senão solucionar, ao menos melhorar o problema apresentado. Dentre as metodologias da Abordagem Ágil apresentadas, é dado um destaque e detalhamento maior para o *Scrum;*

– Capítulo 4, Projetos de Desenvolvimento de *Software* em Números: este capítulo é dedicado a apresentar dados estatísticos de fontes, tidas como referência no assunto, que justifiquem o problema apresentado, bem como permitam afirmar se a alternativa proposta como solução ao problema levantado é plausível;

– Capítulo 5, Conclusão: capítulo dedicado a reflexão sobre os conceitos e dados estatísticos apresentados.

2. ACERCA DE UM PROJETO DE *SOFTWARE*

2.1. GERENCIAMENTO DE PROJETOS

O Gerenciamento de Projetos, tem por finalidade definir objetivos (**escopo**) e atingir objetivos; para isso deve otimizar a utilização dos recursos: **tempo**, **custo** financeiro, humano, matéria-prima, energia, espaço físico durante o tempo de um projeto. Normalmente, é responsabilidade de um único gerente de projeto, que nem sempre participa diretamente das pequenas atividades ou tarefas em que é dividido um projeto, mas se responsabiliza por manter o andamento das várias atividades, gerenciando o risco geral de insucesso (GONÇALVES, 2006).

Escopo, tempo e custo não estão destacados inadvertidamente, tratam-se dos pilares (PMBOK, 2008) a serem equilibrados durante um projeto pelo gerente de projeto. Alguns autores os chamam de restrição tripla de um projeto.

2.1.1. PMI®

Quando o assunto é gerenciamento de projetos, o Project Management Institute (PMI®), ou traduzindo Instituto de Gerenciamento de Projetos, fundado em 1969 e sediado na Filadélfia, Pensilvânia EUA, é referência mundial. Trata-se de uma associação, sem fins lucrativos, de profissionais de gerenciamento de projetos com abrangência global, que atualmente possui mais de 500 mil membros espalhados em cerca de 185 países. (PMI®-SP, s.d.).

O PMI® baseia seu trabalho em um conjunto de valores fundamentais e acredita que o gerenciamento de projetos é uma competência crítica que pode impactar positivamente no resultado das organizações e da sociedade. Através do profissionalismo que transparece, inspira responsabilidade, comprometimento e comportamento ético para com as partes interessadas. O voluntariado e parcerias eficazes contribuem para manutenção e sucesso do Instituto. A valorização dos membros contribuem para a valorização de uma forma geral do profissional de gerenciamento de projetos e evolução e manutenção do conceito bem como tornar seus conceitos, crenças e valores fundamentais referência no assunto. (PMI®-Brasil, s.d. C).

2.1.2. PMBOK®

O *Project Management Body of Knowledge* (PMBOK® Guide) é o guia do conjunto de Conhecimento em Gerenciamento de Projetos. Este documento é desenvolvido através do trabalho dos praticantes, pesquisadores e acadêmicos que utilizam as práticas mais difundidas, testadas e aprovadas bem como aquelas inovadoras que se encontram em fase inicial de exploração e estudo acerca da disciplina Gerenciamento de Projetos.

O PMI® é o responsável pela publicação que atualmente encontra-se na 5ª edição, lançada oficialmente em 31 de dezembro de 2012.

2.1.3. Certificação

É sabido que, o PMI® disponibiliza ao público seis certificações distintas, sendo elas (PMI®-Brasil, s.d. A):

- **Certificação PMP** – Profissional de Gerenciamento de Projetos (PMP)®: é a principal e mais reconhecida certificação para Gerentes de Projetos;
- **Certificação CAPM** – Técnico Certificado em Gerenciamento de Projetos®: focada para iniciantes em gerenciamento de projetos que exige um menor nível de conhecimento;
- **Certificação PgMP** – Profissional de Gerenciamento de Programas®: voltada para gerentes de projetos que gerenciam múltiplos projetos complexos com ligação direta com os resultados estratégicos e organizacionais das empresas;
- **Certificação PMI-SP** – Profissional em Gerenciamento de Cronograma do PMI®: certificação voltada para especialização no desenvolvimento e manutenção de cronogramas de projetos;
- **Certificação PMI-RMP** – Profissional em Gerenciamento de Riscos do PMI®: certificação voltada para especialização na avaliação e gerenciamento dos riscos do projeto;
- **Certificação PMI-ACP** – Profissional Certificado em Métodos Ágeis do PMI®: certificação para profissionais que fazem uso de metodologias e práticas Ágeis para gerenciamento de projetos.

Esta última certificação, focada em Métodos Ágeis, é vista pelo mercado e por alguns especialistas como uma resposta do PMI® à forte demanda, principalmente do setor de

Tecnologia da Informação (TI) e Desenvolvimento de *Software*, por Metodologias diferenciadas com capacidade e flexibilidade a mudanças tão exigidas por este segmento. A Certificação PMI-ACP foi anunciada em fevereiro de 2011 e a realização da prova disponibilizada aos interessados em janeiro de 2012. (PMI®-SP, 2012 B)

Apesar de parecer uma estratégia oportunista (negativamente falando), observando vê-se como uma adaptação e adequação a demanda baseada em pesquisas do próprio PMI® e de outros órgãos como o *The Standish Group* (*Chaos Report*) que constatou a movimentação do mercado de desenvolvimento de *software* para utilização de Métodos Ágeis em seus projetos. O detalhamento destas pesquisas é abordado no capítulo 4. Trata-se de um grande desafio mesclar um guia (*guides*) de conhecimento (*knowledge*) e padrões (*standards*) do PMBOK® com uma metodologia voltada para o "como fazer" (*how-to*) em uma certificação com um selo de um Instituto com considerável prestígio internacional frente a outras entidades certificadoras.

2.1.4. O que é um projeto

Segundo o PMBOK® (2008), Gerenciamento de Projetos é: Aplicação de conhecimentos, habilidades, ferramentas e técnicas às atividades do projeto a fim de satisfazer os seus requisitos de forma efetiva e eficaz.

E o que é um projeto? Segundo o PMBOK®:

> Um projeto é um esforço temporário empreendido para criar um produto, serviço ou resultado exclusivo. A sua natureza temporária indica um início e um término definidos. O término é alcançado quando os objetivos tiverem sido atingidos ou quando se concluir que esses objetivos não serão ou não poderão ser atingidos e o projeto for encerrado, ou quando o mesmo não for mais necessário. Temporário não significa necessariamente de curta duração. Além disso, geralmente o termo temporário não se aplica ao produto, serviço ou resultado criado pelo projeto; a maioria dos projetos é realizada para criar um resultado duradouro. Por exemplo, um projeto para a construção de um monumento nacional criará um resultado que deve durar séculos. Os projetos também podem ter impactos sociais, econômicos e ambientais com duração mais longa que a dos próprios projetos.
> Cada projeto cria um produto, serviço ou resultado exclusivo. (PMBOK®, 2008, p.11).

Como uma definição tão aparentemente exata e relativamente precisa pode descrever "coisas" tão diferentes como uma construção de engenharia de tráfego urbano, ponte, viaduto, um prédio, ou o esforço de socorro após um desastre natural ou ataque terrorista, a expansão de vendas de um determinado produto em qualquer que seja o mercado, ou vencer um campeonato esportivo, e por último o desenvolvimento de um complexo *software* para gestão empresarial de uma grande empresa de atuação mundial? O que todos têm em comum? A resposta é: são projetos!

Este trabalho monográfico pretende apartar projetos de desenvolvimento de *software* desta definição generalista e entrar em algumas de suas nuances, afim de entender o elevado índice de insucessos.

2.1.5. Gerente de projeto

São habilidades inerentes a um gerente de projeto: organização, foco a um objetivo pré-estabelecido, e a compreensão do papel estratégico pelo qual as organizações obtém sucesso, aprende e muda. O gerente de projetos é responsável por criar um contexto favorável ao grupo para que todos acreditem no objetivo compartilhado e lutem por ele. Possui a habilidade de extrair o melhor de sua equipe, conforme a característica individual de cada integrante do projeto e com isso cria-se confiança e um eficaz canal de comunicação tornando-o um agente de mudanças. É capaz de lidar com a pressão e age com naturalidade diante de mudanças e ambientes dinâmicos. Tem habilidade para transformar atividades complexas e interdependentes em tarefas e sub-tarefas que podem ser documentadas, monitoradas e controladas. Esse é um papel de grande importância nas empresas e a demanda por estes profissionais é crescente dada a relevância do tema Gerenciamento de Projetos para empresas das mais diversas áreas e segmentos de atuação. (PMI®-Brasil, s.d. B)

2.2 ENGENHARIA DE *SOFTWARE*

É possível encontrar diversas definições, umas com mais outras com menos detalhes. Com destaque para duas, sendo a primeira a definição do dicionário Michaelis para Engenharia: "...**1** Arte de aplicar os conhecimentos científicos à invenção, aperfeiçoamento ou utilização da técnica industrial em todas as suas determinações ...", e a segunda, a definição oficial do Instituto de Engenheiros Eletricistas e Eletrônicos (IEEE) *Standard Computer Dictionary*, que define engenharia de *software* como sendo "a aplicação de uma abordagem sistemática, disciplinada e mensurável para desenvolvimento, operação e manutenção de *software*; isto é, a aplicação da engenharia ao *software*." (2002 *apud* TELES, 2005).

Para Teles, esta segunda abordagem visa alcançar na área de desenvolvimento de *software* o mesmo nível de previsibilidade, determinismo e acerto presente em outros ramos da engenharia. (TELES, 2005).

3. METODOLOGIA DE DESENVOLVIMENTO DE *SOFTWARE*

Em um artigo de autor desconhecido, é feita uma analogia bastante interessante que se aplica ao Desenvolvimento de *Software*:

> "*software* se chama "*soft*" justamente porque não é "*hard*". Hardware – computadores, micro-processadores, periféricos, etc – esses podem ser desenvolvidos segundo a engenharia clássica. *Software*, por outro lado, se aproxima muito mais de "arte" do que de "engenharia". Existem técnicas, mas nenhuma delas garante um bom *software* tanto quanto apenas técnicas de culinária não garantem um bom jantar." (BLOG DA LOCAWEB, 2009)

Este capítulo aborda as principais técnicas e metodologias, dentro das abordagens Tradicional e Ágil, com enfoque para o *Scrum* que é o objeto principal de estudo deste trabalho.

3.1. Abordagem Tradicional

Na Abordagem Tradicional de Desenvolvimento de *Software*, o custo da mudança aumenta exponencialmente ao longo do ciclo de vida do projeto, ou seja, quanto mais adiantada a fase em que o projeto se encontra mais onerosa será contemplar qualquer mudança. (PMI®-SP, 2012 A).

Uma característica muito forte é a formalização e elaboração de documentos que demonstram a exatidão e lógica do que está sendo elaborado (exatamente como na Engenharia clássica) e por isso qualquer mudança impacta diretamente o que já foi construído, elaborado e documentado nas etapas anteriores e que no caso da necessidade de uma mudança precisarão ser revistos, revalidados ou até mesmo reconstruídos, gerando despesas e aumentando o custo e orçamento do projeto, fora o desperdício de esforço dedicado a algo que será "jogado fora".

3.1.1. Cascata

O modelo cascata (*waterfall*) é um termo comum na grande maioria dos materiais de Engenharia de *Software* consultados durante a pesquisa. Há registros de metodologias semelhantes do final da década de 1950, mas foi em 1970 que Winston W. Royce escreveu um artigo que descreve formalmente o modelo utilizado até hoje.

O modelo cascata descreve uma metodologia dividida em fases onde o produto final de uma fase serve como entrada para a fase seguinte. A seguir a imagem (Figura 1) do artigo original de Winston W. Royce, que demonstra o sequenciamento das fases: Requerimentos do Sistema (sistema como um todo e não sistema de informação), Requerimentos de *Software*, Análise, Prototipação de programas, Codificação, Testes e Operação. (ROYCE, 1970)

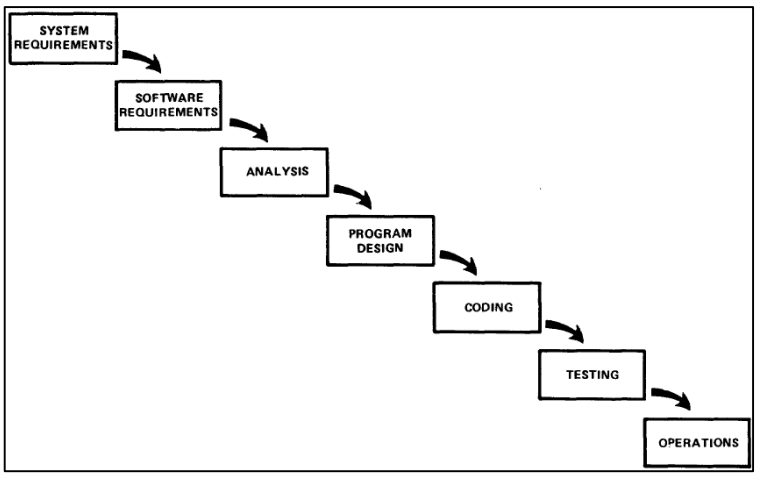

Figura 1 - Passos para implementação de *software* para atendimento da(s) necessidade(s) de um cliente
Fonte: (ROYCE, *1970)*

Há, atualmente, inúmeras variações e adaptações deste modelo conforme a necessidade de cada organização. Como fatores comuns entre elas, é possível citar algumas características como: linearidade, formalidade onde cada etapa tem seus passos muito bem definidos, reforço da necessidade de um planejamento prévio como pré-requisito para um bom resultado, necessidade de acompanhamento e gerenciamento no decorrer do processo e pontos de validação. Apesar de para cada uma dessas características podermos tirar pontos positivos e negativos, em uma visão ampla o modelo cascata agrega valor, afirmação que se justifica pelo simples fato de ser utilizado até hoje, sem mencionar que é a forma mais difundida e utilizada quando se trata de Engenharia de *Software*. Por outro lado, o dito popular "em time que está ganhando não se mexe" não pode ser aplicado pois o conformismo e a acomodação são prisões para o conhecimento e para as ideias, assim como para as técnicas melhores, mais eficientes e aprimoradas. O próprio precursor do método, curiosamente no mesmo artigo abre espaço para expansões e diz: "Eu acredito no conceito, mas a implementação descrita acima é arriscada e convida a falhas." (1970 *apud* BLOG DA LOCAWEB, 2009).

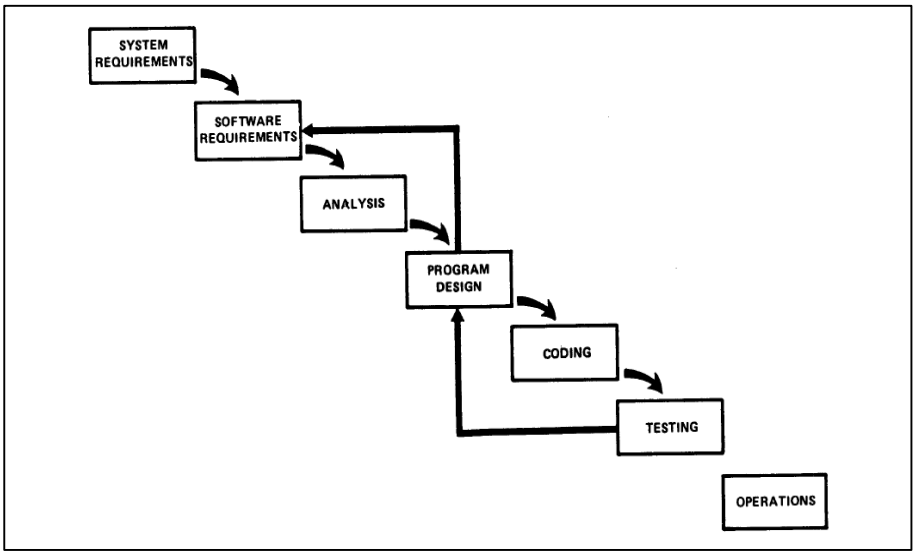

Figura 2 - As iterações nem sempre são formadas por passos sucessivos e lineares
Fonte: (ROYCE, 1970)

A Figura 2 pode ser vista como a raíz do modelo iterativo, com ciclos intermediários e não obrigatoriamente lineares, e sim mais flexíveis. (BLOG DA LOCAWEB, 2009).

3.1.2. Espiral

Este modelo é formalizado no artigo "*A Spiral Model of Software Development and Enhancemen*t" de Barry Boehm (1986). Duas características podem ser destacadas no modelo proposto por Boehm: a avaliação de riscos e a prototipação além da iteração destes dois processos se repetindo no decorrer do projeto com a finalidade de que sejam validados (protótipos) e reavaliados (riscos) conforme se apresentam à situação naquele momento.

A Figura 3 ilustra o Modelo Espiral. O processo inicia-se pelo planejamento de requisitos e planejamento de ciclo de vida (*Requirements plan life-cycle plan*), na sequência acontece a avaliação de riscos (*Risk analysis*) e prototipação (*Prototype*) pela primeira vez; na sequência teremos as seguintes fases, intercaladas por avaliação de riscos (*Risk analysis*) e prototipação (*Prototype*), entre algumas delas conforme a figura, Concepção de Operação (*Concept of operation*), Requisitos de *Software* (*Software requirements*), Validação de requisitos (*Requirements validation*), Planejamento de Construção (*Development plan*), Design do produto (*Software product design*), Validação e verificação do Design do produto (*Design validation and verification*), Planejamento de Integração e Teste (*Integration and test plan*), Desing detalhado (*Detailed design*), Codificação (*Code*), Teste Unitário (*Unit Test*),

Integração e Teste (*Integration and test*), Teste de aceitação ou Homologação (*Acceptance test*) e Implementação (*Implementation*).

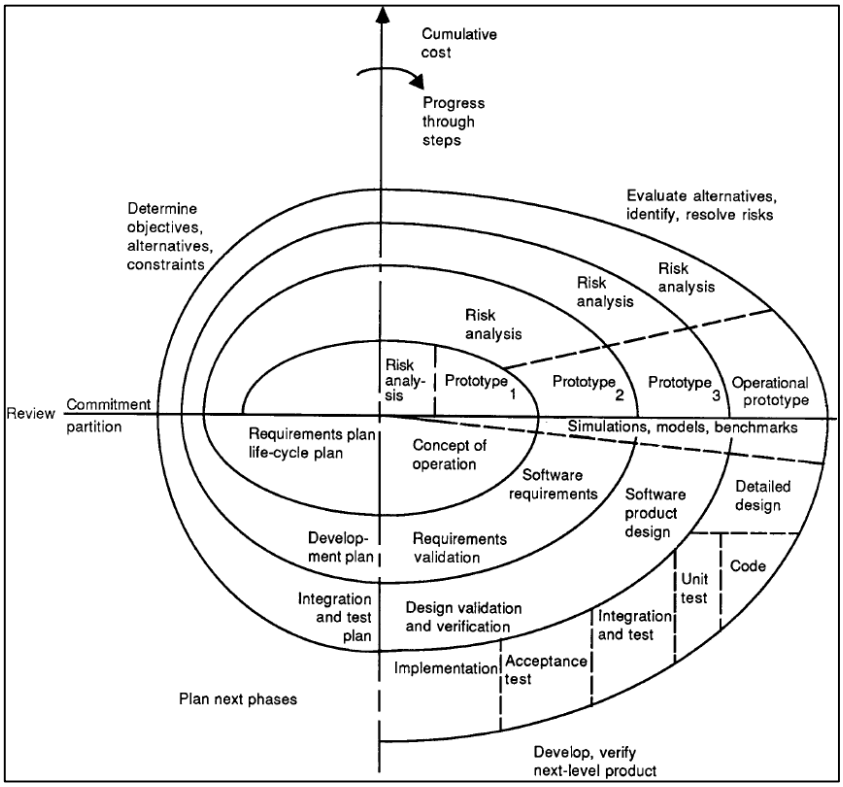

Figura 3 - Processo do Modelo Espiral
Fonte: (BOEHM, 1986)

3.1.3. RUP

Rational Unified Process (RUP) é um modelo proprietário, elaborado pela *Rational Software Corporation*, uma divisão da IBM®.

As duas características principais que caracterizam este modelo são: a abordagem da orientação a objetos, projeto e documentação utilizam UML (*Unified Modeling Language*) como notação e, que por ter uma empresa (IBM®) responsável, possui *softwares* que apoiam as etapas do modelo.

O RUP, possui quatro fases que geram artefatos: Concepção, Elaboração, Construção e Transição. Estas fases são compostas por iterações.

A Figura 4 ilustra o quanto cada uma das disciplinas (Modelagem de Negócios, Modelagem de Requisitos, Análise e Design, Implementação (ou construção), Teste, Implantação, Gerenciamento de Configuração e Mudança, Gerenciamento de Projeto e

Ambiente) compõem cada uma das fases do modelo RUP: Iniciação, Elaboração, Construção e Transição.

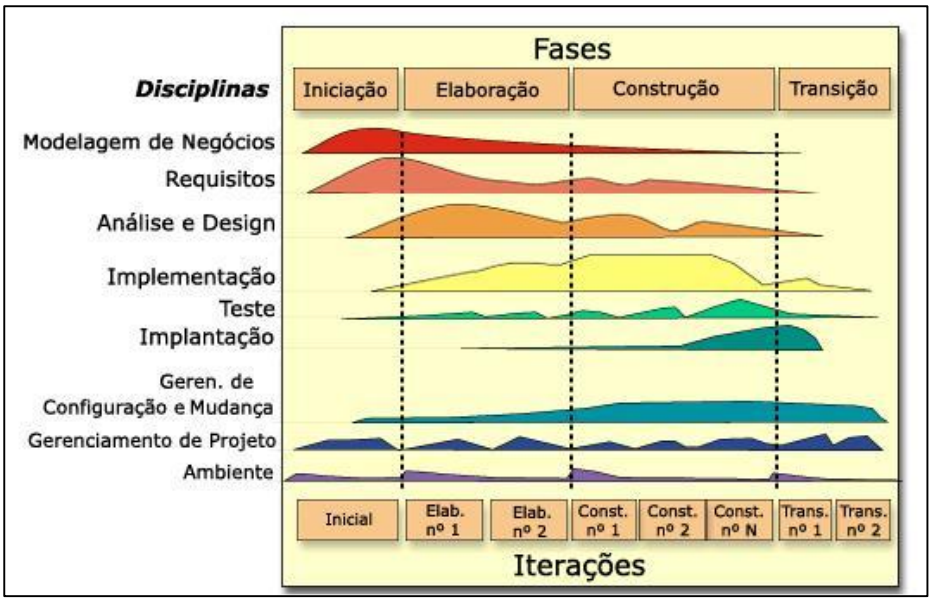

Figura 4 - Fases do RUP
Fonte: (SILVA, s.d.)

3.2 Abordagem Ágil

Serão abordadas a seguir as principais Metodologias Ágeis.

3.2.1. Manifesto Ágil

Em 2001 representantes de diferentes técnicas ágeis, como *Extreme Programming* (XP), *Scrum*, *Dynamic Systems Development Method (DSDM)*, Adaptive *Software Development*, *Crystal*, *Feature-Driven Development*, *Pragmatic Programming*, entre outros, se reuniram e escreveram o Manifesto Ágil e seus doze princípios, que constam na integra abaixo:

> **"Manifesto para o desenvolvimento ágil de *software***
> Estamos descobrindo maneiras melhores de desenvolver *software* fazendo-o nós mesmos e ajudando outros a fazê-lo. Através deste trabalho, passamos a valorizar:
>
> | | | |
> |---|---|---|
> | **Indivíduos e interação entre eles** | mais que | processos e ferramentas |
> | ***Software* em funcionamento** | mais que | documentação abrangente |
> | **Colaboração com o cliente** | mais que | negociação de contratos |
> | **Responder a mudanças** | mais que | seguir um plano |
>
> Ou seja, mesmo havendo valor nos itens à direita, valorizamos mais os itens à esquerda.

Princípios por trás do manifesto ágil

Nós seguimos os seguintes princípios:

- Nossa maior prioridade é satisfazer o cliente, através da entrega adiantada e contínua de *software* de valor.
- Aceitar mudanças de requisitos, mesmo no fim do desenvolvimento. Processos ágeis se adequam a mudanças, para que o cliente possa tirar vantagens competitivas.
- Entregar *software* funcionando com frequência, na escala de semanas até meses, com preferência aos períodos mais curtos.
- Pessoas relacionadas à negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante todo o curso do projeto.
- Construir projetos ao redor de indivíduos motivados. Dando a eles o ambiente e suporte necessário, e confiar que farão seu trabalho.
- O Método mais eficiente e eficaz de transmitir informações para, e por dentro de um time de desenvolvimento, é através de uma conversa cara a cara.
- *Software* funcional é a medida primária de progresso.
- Processos ágeis promovem um ambiente sustentável. Os patrocinadores, desenvolvedores e usuários, devem ser capazes de manter indefinidamente, passos constantes.
- Contínua atenção à excelência técnica e bom design, aumenta a agilidade.
- Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito.
- As melhores arquiteturas, requisitos e designs emergem de times auto organizáveis.
- Em intervalos regulares, o time reflete em como ficar mais efetivo, então, se ajustam e otimizam seu comportamento de acordo.” (BECK et al, 2001).

Este manifesto é considerado o marco inicial para este jeito novo de pensar na Engenharia de *Software*, que são os Métodos Ágeis, mesmo considerando que seus autores já possuíam publicações descrevendo suas experiências, propostas e descritivos de melhores práticas das particularidades das metodologias que cada um propunha.

3.2.2. SCRUM

3.2.2.1. História

A história do *Scrum* não é marcada por um criador específico. Foi citado pela primeira vez em um artigo da *Harward Bussines Review*, titulado "*The New Product Development Game*", em janeiro de 1986, escrito por Takeuchi e Nonaka. Era um artigo voltado para a indústria automobilística e tinha como base o sistema Toyota de produção (*Lean*). Nele eram destacados os resultados obtidos por equipes pequenas e multidisciplinares, e estas equipes altamente eficazes foram associadas à formação de reinicio de jogo utilizado em alguns momentos no *Rugby*, que é chamada de *Scrum*, e daí a origem do nome. (HAAS, 2010).

Sete anos mais tarde, em 1993, Jeff Sutherland, John Scumniotales e Jeff McKenna começaram a moldar o *framework*. Em 1995, Ken Schwaber formalizou a definição e adaptou à realidade de desenvolvimento de *software*, apesar de não a restringir apenas a esta área. (DALONSO, 2010). Schwaber viria a fazer, em 2001, parte do grupo responsável pelo Manifesto Ágil. (BECK et al, 2001).

3.2.2.2. Definição, o que é *Scrum*?

A definição mais adequada é a dos responsáveis pela formalização do *framework Scrum*:

> "*Scrum* é uma estrutura processual (*framework*) para suportar o desenvolvimento e manutenção de produtos complexos. O *Scrum* consiste em Equipes do *Scrum* associadas a seus papéis, eventos, artefatos e regras. Cada componente dentro do *framework* serve a um propósito específico e é essencial para o uso e o sucesso do *Scrum*." (SCHWABER; SUTHERLAND, 2011).

Todas as características que serão abordadas nos tópicos a seguir são importantes, mas é possível destacar como principal o fato de que ***Scrum* é iterativo incremental** e está previsto no processo procedimentos constantes e repetitivos de checagem no decorrer do projeto (Retrospectiva), para cada ciclo de entrega (*Sprint*) espera-se que haja produto pronto e com percepção de valor para o cliente no menor tempo possível; é altamente flexível quanto a mudança, pois os ciclos (S*prints*) são curtos e consequentemente os pontos de checagem acontecem em intervalos menores.

3.2.2.3. Papéis

– ***Product Owner***: ou Dono do Produto, é o principal conhecedor da regra do negócio. Normalmente é o próprio cliente e deve estar sempre presente e principalmente ter autonomia para tomada de decisões quanto ao rumo do projeto, quais itens do *Product Backlog* serão atendidos e em que ordem e/ou prioridade. Pode até mesmo representar a opinião de um grupo, mas deve ser centralizado na posição de uma pessoa com livre poder de decisão. (SCHWABER; SUTHERLAND, 2011).

– **Time *Scrum***: são profissionais que recebem o título de desenvolvedores, independente da especialização de cada indivíduo. A resposta final e o posicionamento são sempre do grupo. Este grupo tem como principais características: ser **auto-gerenciado**, ou seja, não possui hierarquia e não precisa que alguém diga como que irá transformar o *Product Backlog* em produto final e são **multi-disciplinares**, ou seja, a característica de cada profissional deve ser complementar para com os demais. (SCHWABER; SUTHERLAND, 2011).

– ***Scrum Master***: é o profundo conhecedor do *Scrum*, responsável por garantir a utilização das técnicas no decorrer do projeto e facilitar para que os eventos aconteçam conforme o previsto. Tem como atribuição remover os obstáculos, seja qual for a natureza desses impedimentos para o projeto, e blindar o time de interferências externas que possam prejudicar o desempenho da equipe. (DALONSO, 2010).

O *Scrum Master* tem responsabilidades para com a corporação, sendo o responsável por disseminar a cultura e técnicas do *Scrum* para com o *Product Owner*, capacitando-o a identificar e gerenciar o *Product Backlog*. Para a equipe *Scrum*, é o principal incentivador de utilização de todas as técnicas de aprimoramento da interdisciplinaridade e auto gerenciamento, evitando que problemas administrativos ou qualquer outro assunto que não relacionado ao projeto em si interfira no rendimento da equipe. (SCHWABER; SUTHERLAND, 2011).

3.2.2.4. Cerimônias

A Figura 5 ilustra o Fluxo *Scrum*, desde a concepção da ideia (Visão) até a entrega do produto acabado (Incremento do Produto). As cerimônias e artefatos serão descritos a seguir, de forma resumida. O processo inicia-se a partir de uma ideia (visão), que vira um item na lista *Product Backlog*. Utilizando esta lista como entrada, acontece o *Planning Meeting*, onde é decidido quais itens farão parte da próxima *Sprint*, ou seja, quais itens do *Product Backlog* irão compor o *Sprint Backlog*. Durante uma *Sprint*, que pode ter até quatro semanas, acontece o *Daily Meeting* (ou Reunião Diária). Ao término da *Sprint* acontecem o *Review* (ou Revisão da *Sprint*) e a Retrospectiva da *Sprint*, além do produto final entregue ao cliente.

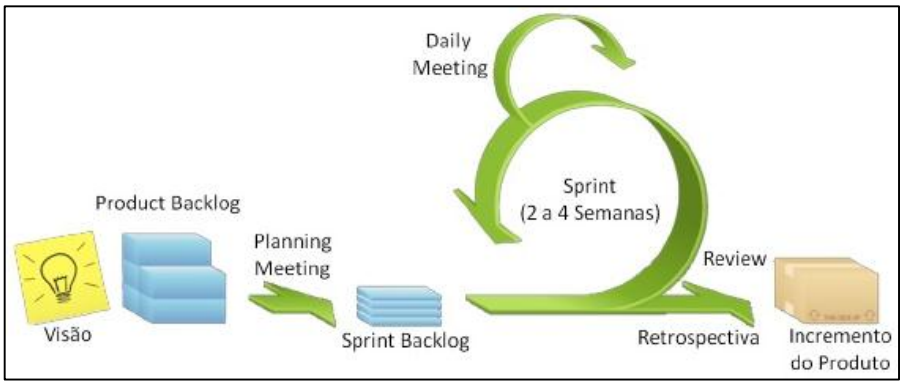

Figura 5 - Fluxo *Scrum*
Fonte: (DALONSO,2010)

– ***Sprint*:** tem duração máxima (recomendada) de quatro semanas e é composto pelo planejamento (Planning Metting), Reuniões Diárias (*Daily Meeting*), o trabalho de construção em si, a revisão (*Review*), uma Retrospectiva da *Sprint* e o Produto pronto (Incremento do produto).

A *Sprint* não sofre mudança de data de entrega. Se ao final do prazo estabelecido o produto não está pronto, serão levantados na Revisão e Retrospectiva os motivos, e o que está pronto é entregue do jeito que está, desde que agregue algum benefício ao cliente. A duração de até quatro semanas é justamente para diminuir o risco de ocorrerem mudanças nas prioridades. (SCHWABER; SUTHERLAND, 2011).

– **Cancelamento da *Sprint***: apenas o *Product Owner* tem autoridade para cancelar uma *Sprint*. Quando isso acontece, o produto é revisto e é devolvido ao *Product Backlog* ou

caso tenha algo que agregue valor, pode ser aceito como entrega pelo *Product Owner*. (SCHWABER; SUTHERLAND, 2011).

– **Reunião de Estimativa:** etapa de preparação para o *Planning meeting*, onde é estimado no tamanho (e não no tempo) cada item do *Product Backlog*. (MACAÚBAS; PEREIRA, 2012?)

– ***Planning meeting*** (Planejamento da *Sprint*): o que e como será entregue. Nesta etapa toda a equipe participa para definir quais itens do *Product Backlog* farão parte do *Sprint*.

No momento de definição de "o que será entregue", o time *Scrum* faz a estimativa, utilizando-se de técnicas como o *Planning Poker*, para definir a complexidade e prazo de cada item do *Product Backlog* e, juntamente com o *Product Owner* e orientação e supervisão do *Scrum Master*, é definido o que cabe e o que será feito na *Sprint*.

Na fase "como será entregue", o time *Scrum* alinha as estratégias e tem como produto deste trabalho o *Sprint Backlog*. (SCHWABER; SUTHERLAND, 2011).

– **Daily meeting** (Reunião diária): o próprio nome já indica que deve acontecer todos os dias. É voltada para o time *Scrum* e o *Scrum Master*. O principal objetivo é responder algumas perguntas antes que comece o trabalho do dia para que todos os envolvidos estejam alinhados com o andamento e saibam o que será produzido naquele dia. As perguntas são: 1) O que foi concluído desde a última reunião? 2) O que será feito até a próxima reunião? E 3) Quais os obstáculos que estão prejudicando o caminho atualmente? Tem duração de apenas 15 minutos e normalmente acontece com os membros da equipe em pé, para garantir a objetividade e foco da reunião. (SCHWABER; SUTHERLAND, 2011).

– ***Review*** (Revisão da *Sprint*): acontece ao final da *Sprint* e tem por objetivo validar o produto concluído e dar baixa no *Product Backlog*, ou então registrar a entrega parcial e adaptar o item remanescente no *Product Backlog*. É comum o debate já antecipar os itens que serão tratados na próxima *Sprint*, através da troca de experiências do que aconteceu na *Sprint* que acabou de encerrar-se. (SCHWABER; SUTHERLAND, 2011).

– **Retrospectiva da *Sprint***: funciona como reunião de *feedback* entre a equipe, onde é debatido o que houve de bom e funcionou e também o que precisa ser corrigido e/ou melhorado para as próximas *Sprints*. (SCHWABER; SUTHERLAND, 2011).

Quanto mais tempo e maior a experiência da equipe, mais assertiva se tornam as estimativas. O entrosamento também reflete na qualidade geral da equipe em termos de eficiência e capacidade de auto gerenciamento.

3.2.2.5. Artefatos

– ***Product Backlog***: é uma lista priorizada com tudo que um determinado negócio, seja um produto já existente ou necessidade de um novo, precisa em termos de incremento e/ou melhoria para lidar com as necessidades diárias requisitadas por seus clientes. Não é fixa e está em frequente atualização pelo *Product Owner*. (SCHWABER; SUTHERLAND, 2011).

– ***Sprint Backlog***: são os itens selecionados pelo time *Scrum* no *Planning Meeting* que farão parte do produto a ser incrementado na *Sprint* em questão. (SCHWABER; SUTHERLAND, 2011).

– **Monitorando o Progresso**: existem diversas formas de controlar e monitorar o andamento das atividades, as mais comuns são o *Burndown Chart* e o Kan-Ban.

– ***Burndown Chart***: é uma forma gráfica de representar a quantidade de trabalho (atividades) restante *versus* a quantidade de tempo (dias) transcorrido de uma *Sprint*. (LIMA, 2012).

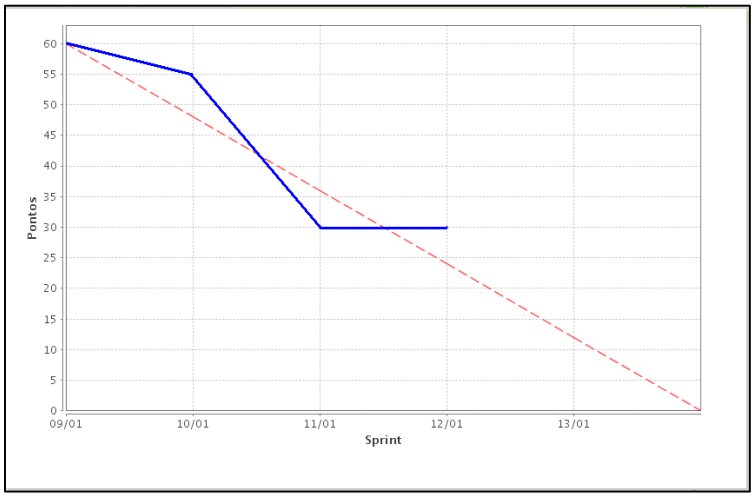

Figura 6 - *Burndown Chart* no decorrer do projeto
Fonte: (LIMA, 2012)

A Figura 6, por exemplo, demonstra a evolução de uma *sprint* durante uma semana. Esta demonstração se passa em um ano qualquer, entre os dias 09/01 e 14/01; podemos notar que no dia 09/01 (eixo X) existem 60 itens/atividades (eixo Y) do *Sprint Backlog* por fazer, ou seja, pendentes. Para os dias seguintes, a linha azul demonstra a evolução de itens por fazer remanescentes. A linha vermelha tracejada demonstra o número médio de atividades que deveriam restar a cada dia para que ao final da *Sprint* todos os itens/atividades planejados tenham sido devidamente concluídos. De uma forma bastante direta e rápida, qualquer envolvido no projeto olhando para o gráfico deverá entender que caso a linha azul esteja acima da vermelha, o projeto está atrasado, estando abaixo, o projeto estará adiantado e, estando exatamente em cima, o projeto estará no prazo.

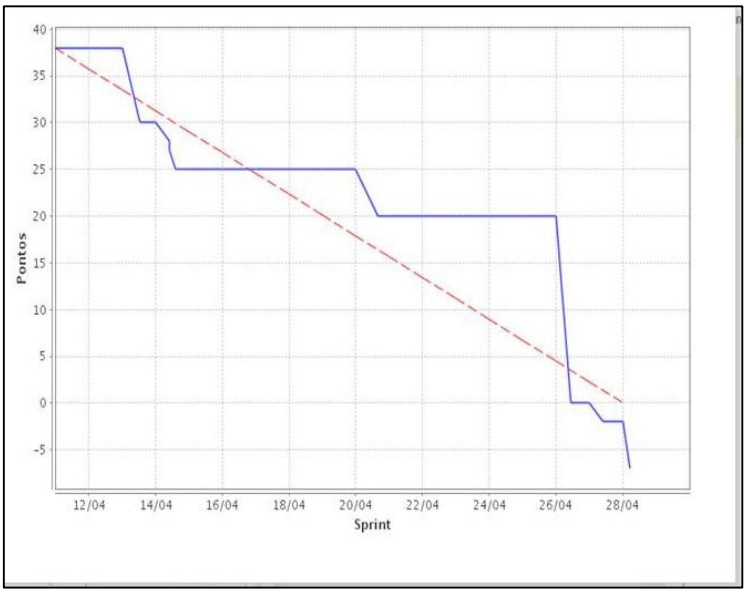

Figura 7 – *Burndown Chart* no final do projeto
Fonte: (LIMA , 2012)

A Figura 7 mostra o gráfico *Burndown Chart* no final do projeto. A partir dele é possível concluir que a evolução não foi regular, alternando períodos em que os itens/atividades estavam atrasados com períodos em que estavam adiantados, considerando uma velocidade regular para conclusão de tarefas. Além disso, os itens/atividades foram concluídos antes do término, inclusive com acréscimo de itens/atividades que foram entregues apesar de não terem sido previstos inicialmente.

A evolução irregular não significa necessariamente um problema; a percepção exata só poderá ser esclarecida pela própria equipe já que a unidade de medida são atividades independentes e que podem possuir complexidade e necessidade de esforço diferentes.

– ***Kanban***: é uma forma de organizar graficamente o fluxo de trabalho. Consiste em dividir o trabalho em tarefas e escreve-las em cartões; posteriormente desenha-se um quadro onde as colunas representam a fase do fluxo e por último os cartões são movimentados entre as colunas de acordo com a etapa em que se encontram. (KNIBERG; SKARIN, 2009).

O quadro *Kanban* deve ser atualizado durante o *Daily Meeting*.

O objetivo principal é que todos os envolvidos, principalmente o time *Scrum,* saibam o que cada integrante está fazendo e o que está concluído do todo. Pode ser feito de forma artesanal, colando folhas pela parede e desenhando as colunas, ou até mesmo utilizando o auxílio de *softwares* próprios para isso.

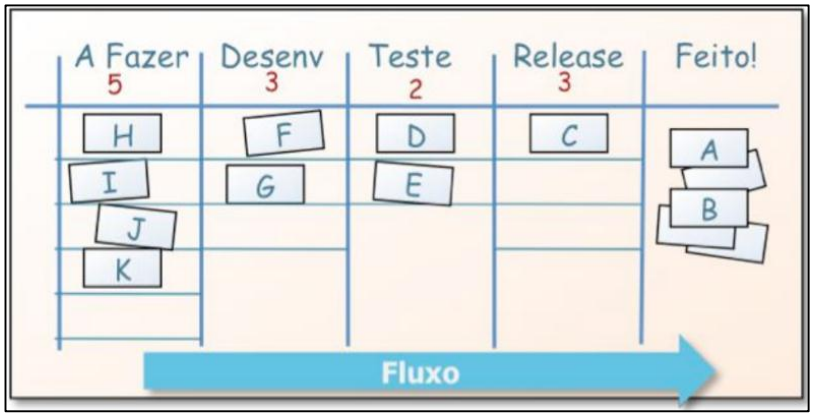

Figura 8 - Kanban
Fonte: (KNIBERG; SKARIN, 2009)

A Figura 8 ilustra um quadro *Kanban*, dividido em cinco colunas: 1) A Fazer; 2) Em Desenvolvimento; 3) Em Teste; 4) Release e 5) Feito. No cabeçalho de cada uma delas há o número de itens contidos na coluna naquele momento. Nos cartões de cada atividade costuma-se identificar quem está responsável por ela. Não há uma restrição quanto ao número, nome e quantidade de itens de cada coluna, ou seja, podem ser adaptados conforme a necessidade.

– **User Stories**: é a ferramenta utilizada para criação dos itens que farão parte do *Product Backlog* pelo *Product Owner* e é uma forma de estruturar as necessidades do cliente. Consiste em um "Cartão de história" onde são respondidas para cada necessidade levantada as seguintes questões: 1) Quem pede? 2) O que pede? e 3) Para que pede?

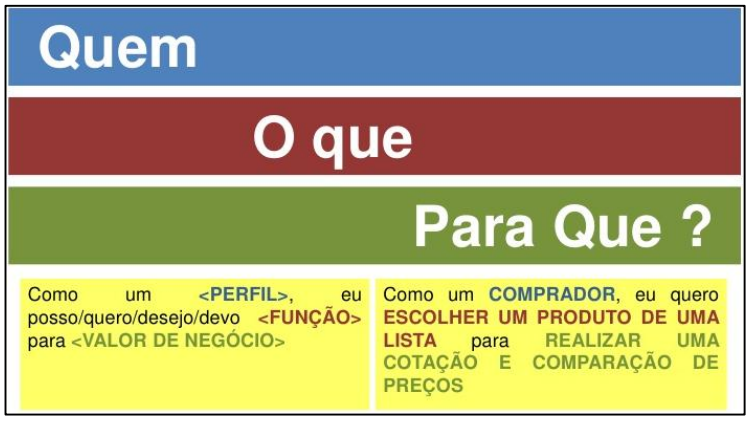

Figura 9 - User Stories
Fonte: (DALONSO, 2010)

A Figura 9 ilustra um exemplo de um "Cartão de História" utilizado para definir o escopo de cada necessidade identificada pelo *Product Owner*.

– **Planning Poker**: o time *Scrum* é o responsável por executar as tarefas durante a *Sprint* em questão, sendo, desta forma, o responsável por estimar a complexidade e prazo de cada atividade. É uma técnica que estimula o diálogo entre o time *Scrum* antes que se chegue a um consenso de complexidade e prazo médio para se entregar cada item.

Para cada item debatido, cada integrante seleciona uma carta com um número que corresponda, na sua opinião, para complexidade e prazo estimados. Após todos selecionarem em segredo, as cartas são mostradas e a partir daí são debatidos os argumentos de cada um para tal nota até que se chegue a um consenso, sendo possível mais de uma rodada de seleção de cartas para que haja esse consenso de estimativa e de complexidade e prazo para cada item. (DALONSO, 2010).

3.2.2.6. Quem usa o Scrum?

Inúmeras empresas globais e nacionais utilizam o *Scrum*. Como exemplos tem-se: Microsoft, Yahoo, Google, Electronic Arts, Philips, Siemens, Nokia, BBC, Itaú, Globo entre outras (COHN, 2011?)

3.2.3. *Extreme Programming* (XP)

A especificação da metodologia *Extreme Programming* (XP) foi feita pelo engenheiro de *software* americano Kent Beck, um dos dezessete autores do Manifesto Ágil. O marco

oficial da metodologia foi a aplicação dos conceitos elaborados e descritos em meio acadêmico por Kent Beck entre os anos de 1987 e 1996, em um projeto de folha de pagamento, no ano de 1997, na empresa Chrysler que já havia tido insucesso utilizando-se da metodologia tradicional de desenvolvimento de *software*. (TELES, 2005).

A metodologia se sustenta em quatro valores básicos: comunicação, simplicidade, *feedback* e coragem. (BECK, 2004). Suas características são descritas por doze práticas: Cliente Presente, Jogo do Planejamento, *Stand up meeting*, Programação em Par, Código Coletivo, Código Padronizado, Design Simples, Desenvolvimento Orientado a Testes, Refatoração, Integração Contínua, *Releases* Curtos e Metáfora. (TELES, 2005).

3.2.4. *Dynamic Systems Development Method* (DSDM)

É uma Metodologia proprietária, desenvolvida por um consórcio de consultores, especialistas e vendedores da área de Sistemas de Informação. Na Inglaterra, na década de 1990, foi aplicada pela primeira vez em 1995. (TEIXEIRA).

Tem como característica principal o fato de ser baseada em ferramenta de Desenvolvimento Rápido de Aplicação (RAD). É iterativa, incremental e prevê o envolvimento constante do usuário. (Wikipedia, s.d. A).

3.2.5. *Adaptive Software Development* (ASD)

O *Adaptive Software Development (ASD)* foi idealizado por Jim Highsmith, um dos integrantes do grupo responsável pelo Manifesto Ágil, e é voltado para desenvolvimento de sistemas e *softwares* complexos. A ênfase deste método é uma equipe auto gerenciável, além de se preocupar mais com os conceitos e cultura organizacional do que com práticas de *software* (2002, *apud* DA SILVA, 2010).

3.2.6. Crystal

O método Crystal foi idealizada por Alistair Cockburn, em 1991. Sua particular característica é o fato de utilizar-se de cores para representar complexidade, tamanho e importâncias dos projetos. É um método que dá ênfase para comunicação e cooperação entre pessoas (2002, *apud* DA SILVA, 2010).

3.2.7. *Feature-Driven Development* (FDD)

O *Feature-Driven Development* (FDD), surgiu em 1997, em Singapura, e em português significa Desenvolvimento Guiado por Funcionalidades. Sua formalização foi feita por Jeff De Luca e Peter Coad (GOYAL, 2007). É situado numa posição intermediária entre os métodos tradicionais e ágeis, combinando princípios de ambos os grupos (Wikipedia, s.d. B).

O FDD é dividido em duas fases: Concepção & Planejamento e Construção; estas duas fases abrangem cinco processos divididos conforme se segue (HEPTAGON, s.d. A; s.d. B):

A) Concepção & Planejamento:
   - Desenvolver um Modelo Abrangente: Análise Orientada por Objetos;
   - Construir a Lista de Funcionalidades: Decomposição Funcional;
   - Planejar por Funcionalidade: Planejamento Incremental.

B) Construção:
   - Detalhar por Funcionalidade: Desenho (Projeto) Orientado por Objetos;
   - Construir por Funcionalidade: Programação Orientada por Objetos.

3.2.8. *Lean*

Surgiu na indústria automobilística, no Japão, mais precisamente na Toyota em seu conhecido sistema Toyota de produção, também conhecido por *Lean Manufacturing* (2010 *apud* PEREIRA, 2012?).

Trazido para o mundo de Tecnologia da Informação, segundo Steffen (2011), *Lean* é um Método Ágil que tem como foco principal eliminar desperdícios no processo de desenvolvimento de *software*. Há inúmeras possibilidades de desperdício no processo de desenvolvimento de *software*, como por exemplo: projetos cancelados na metade por falta e/ou mudança de prioridade; códigos mal documentados que exigem uma reconstrução em momento de manutenção; perda de profissionais no decorrer do projeto fazendo com que o novo integrante tenha que passar pela curva de aprendizado e normalmente atrasando os prazos acordados.

A metodologia *Lean* é fundamentada em 7 princípios (2011, STEFFEN):

- Elimine Desperdícios;
- Inclua a Qualidade no Processo;
- Crie Conhecimento;
- Adie Decisões e Comprometimentos;
- Entregue o quanto antes;
- Respeite as Pessoas e Potencialize a equipe;
- Otimize o Todo.

4. PROJETOS DE DESENVOLVIMENTO DE *SOFTWARE* EM NÚMEROS

4.1. Estatísticas e Relatórios

Partindo da problemática de alto índice de insucesso em projetos, especificamente na área de Tecnologia da Informação (TI) e Desenvolvimento de *Software*, este capítulo pretende organizar relatórios e resultados já publicados pelas fontes mais confiáveis para este assunto, como o PMI®, já citado anteriormente, e o *The Standish Group*, uma empresa americana, sediada em Boston, focada em avaliar risco, custo e retorno financeiro em projetos de TI com base em pesquisas de projetos reais e, a partir disso, prestar serviços de consultoria voltadas para melhoria desses aspectos para novos projetos de seus clientes.

4.1.1. *The Standish Group - Chaos Report*

O *The Standish Group* é responsável pela publicação do *Chaos Report*, um conceituado relatório elaborado após criteriosa pesquisa em projetos de TI do mundo inteiro.

O último relatório *Chaos Report* de 2011 (2011 *apud* D'ÁVILA, 2011), com dados até 2010, mostra o índice de sucessos e falhas em projetos de TI:

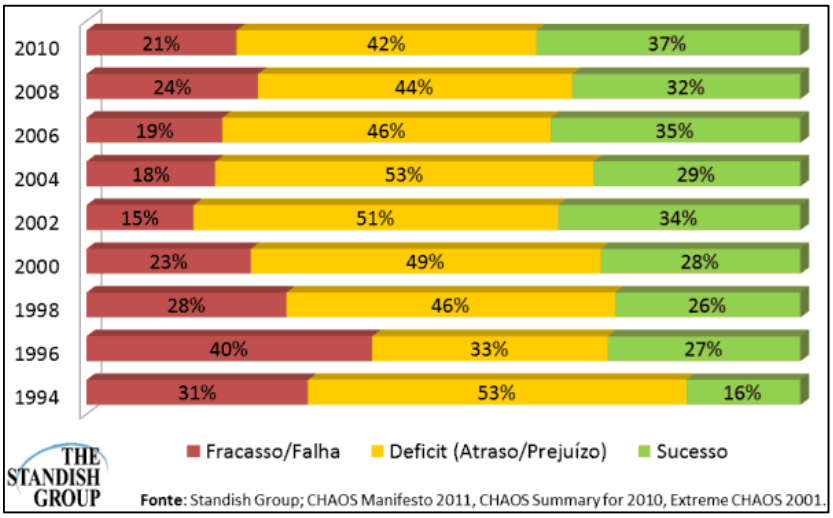

Figura 10- Índice de Sucessos e Falhas em Projetos de TI
Fonte: (2011 *apud* D'ÁVILA, 2011)

Segundo o *Chaos Report* (2011 *apud* D'ÁVILA, 2011), as definições para a classificação utilizada na Figura 10 são:

A) **Sucesso**: projeto bem sucedido frente a tripla restrição (escopo, tempo e custo), ou seja, entregou parte considerável das funcionalidades requisitadas, no tempo prometido e com gasto do orçamento planejado.

B) Deficit (Atraso/Prejuízo) - **Sucesso parcial**: projeto entregue, porém, sem cumprir um item da tripla restrição.

C) Fracasso/Falha - **Fracasso**: projetos não entregues, cancelados ou não utilizados.

O relatório *Chaos Report* de 2009 (2009 *apud* SOUZA, 2010), que analisa os dados referentes a 2008, ilustrados na Figura 10, aponta os principais fatores para as classificações de resultado de projetos já definidos anteriormente - Sucesso, Sucesso Parcial e Fracasso:

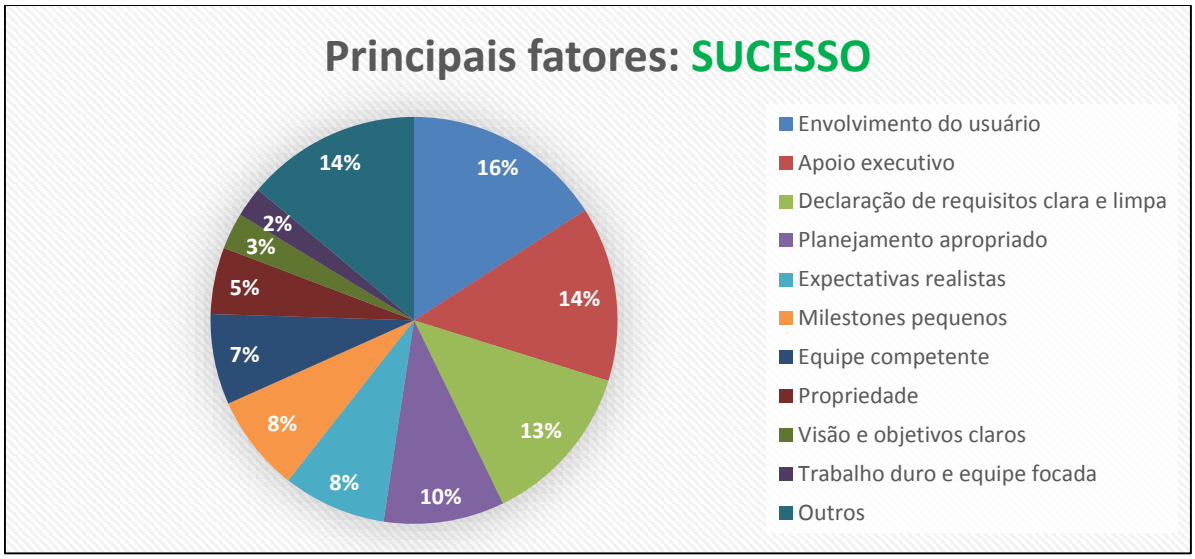

Figura 11 - Principais fatores para "Sucesso" de um Projeto
Fonte: (2011 *apud* D'ÁVILA, 2011)

A partir da análise da Figura 11, obtemos que o "Sucesso" de um projeto de Desenvolvimento de *Software* é determinado por (em percentual de ocorrência): Envolvimento do usuário (16%); Apoio executivo (14%); Declaração de requisitos clara e limpa (13%); Planejamento apropriado (10%); Expectativas realistas (8%); *Milestones* pequenos (8%); Equipe competente (7%); Propriedade (5%); Visão e objetivos claros (3%); Trabalho duro e equipe focada (2%) e Outros (14%).

Figura 12 - Principais fatores para "Sucesso Parcial" de um Projeto
Fonte: (2011 *apud* D'ÁVILA, 2011)

A partir da análise da Figura 12, obtemos que o "Sucesso Parcial" de um projeto de Desenvolvimento de *Software* é determinado por (em percentual de ocorrência): Falta de insumos do usuário (13%); Requisitos & Especificações incompletas (12%); Mudanças nos requisitos & especificações (12%); Falta de apoio executivo (9%); Ambiente tecnológico incompleto (7%); Falta de recursos (6%); Expectativas irrealistas (6%); Objetivos nebulosos (5%); Ciclos (tempo) irrealistas (4%); Novas tecnologias (4%); e Outros (23%).

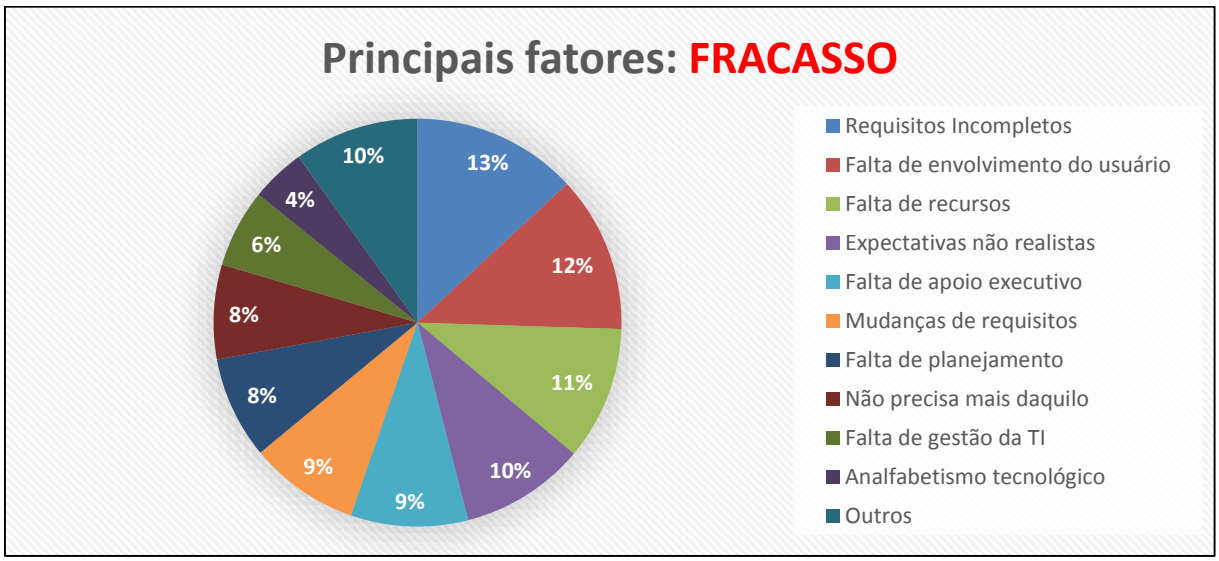

Figura 13 - Principais fatores para "Fracasso" de um Projeto
Fonte: (2011 *apud* D'ÁVILA, 2011)

A partir da análise da Figura 13, obtemos que o "Fracasso" de um projeto de Desenvolvimento de *Software* é determinado por (em percentual de ocorrência): Requisitos

Incompletos (13%); Falta de envolvimento do usuário (12%); Falta de recursos (11%); Expectativas não realistas (10%); Falta de apoio executivo (9%); Mudança de requisitos (9%); Falta de planejamento (8%); Não precisa mais daquilo (8%); Falta de gestão da TI (6%); Analfabetismo tecnológico (4%); e Outros (10%).

No relatório *Chaos Report* de 2011 (2011 *apud* D'ÁVILA, 2011) (que analisa os dados referente até 2010, em que se pode notar um salto no índice de projetos com "Sucesso" para o indicador dos anos de 2008 e 2010 de 32% para 37% conforme detalhado na Figura 10) o *The Standish Group* aponta quatro fatores principais para essa melhora:

A) Projetos Ágeis: aumento da utilização de processos ágeis no desenvolvimento de *software* pelas empresas;

B) Modernização: projetos de modernização (conversão de código/banco de dados, por exemplo) que apesar de ter crescimento menor que projetos ágeis, possui um índice de sucesso maior;

C) Pacotes Empresariais: diminuição de projetos com essa característica (implantação de ERP e CRM, por exemplo) que possui alto índice de insucesso;

D) Processos em cascata: tratam-se dos métodos tradicionais, como crescem em menor velocidade, sua participação relativa ao total de projetos diminui, logo contribuem para a melhoria do índice já que possuem taxa de insucesso maior que projetos ágeis.

4.1.2. PMI® - Estudo de *Benchmarking* em Gerenciamento de Projetos

Quando uma empresa opta por adotar um padrão, um processo ou uma forma de fazer estruturada, organizada, sobretudo diferente da que já utiliza, visa tirar algum benefício, uma vantagem competitiva frente aos seus concorrentes e apresentar à seus clientes um diferencial de qualidade, agilidade, valor, retorno do investimento em comparação com a concorrência. Tão claro quanto o "fim", que é o resultado desejado, é o "início" e o "meio" do processo; por "início" podem ser citados o investimento financeiro, a adequação estrutural, as mudanças de cultura de seus funcionários, os fornecedores e gestores; e por "meio", que seria o conjunto das ações necessárias para que o "fim" seja alcançado. Daí surgem diversas perguntas para quaisquer que sejam os "fins" desejados, e a adoção do conjunto de melhores práticas para gestão de projetos como a defendida pelo PMI® não é diferente, ao passo que visa o quanto se vai melhorar, em quanto tempo e qual o retorno financeiro esperado.

"Estudos do PMI tem demonstrado que a utilização de Métodos Ágeis praticamente triplicou entre dezembro de 2008 e maio de 2011." (PMI®-SP, 2012 B).

O artigo titulado “Problemas Persistentes em Projetos”, publicado no Capítulo São Paulo do PMI®, pelo autor Airton Molena (MOLENA, 2011), apresenta diversos números de pesquisas publicadas pelo PMI® entre 2003 e 2009, sendo que alguns desses números respondem às perguntas acerca das práticas difundida pelo PMI® para gestão de projetos, as quais serão apresentados a seguir.

Primeiramente, o que é interessante saber no caso de uma metodologia já conhecida e utilizada é o quanto ela melhora para quem já utilizou e/ou adotou:

Utilização de metodologias de Gerenciamento de Projeto	Insucesso	Sucesso
Empresas que não utilizam metodologia de GP	51%	49%
Empresas que utilizam metodologia de GP	24%	76%

Figura 14 - Qual a relação entre a utilização de uma metodologia de Gestão de Projetos e o Sucesso em Projetos?
Fonte: (2008 *apud* MOLENA, 2011)

A Figura 14 demonstra que ao adotar uma Metodologia para Gestão de Projetos as empresas, em média, veem o índice de Sucesso em projetos saltar de 49% para 76%.

Em termos financeiros e quantidade de recursos humanos gastos desnecessariamente, na Figura 14 é citada a dimensão do prejuízo:

Utilização de metodologias de Gerenciamento de Projeto	Insucesso	Prejuízo em US$	Pessoas envolvidas*
Empresas que não utilizam metodologia de GP	51%	7,4 trilhões	8,4 milhões
Empresas que utilizam metodologia de GP	24%	3,5 trilhões	4,0 milhões

Figura 15 - Prejuízo por insucesso nos projetos
Fonte: (2008 *apud* MOLENA, 2011)

A Figura 15 explicita que entre as empresas que não utilizam uma metodologia para Gestão de Projetos e que possuem um índice de insucesso em projetos de 51%, são gastos 7,4 trilhões (US$) com o envolvimento desnecessário de 8,4 milhões de pessoas. Já para as empresas que utilizam alguma metodologia para Gestão de Projetos, cujo índice de insucesso em projetos é de 24%, são gastos 3,5 trilhões (US$) com o envolvimento desnecessário de 4,0 milhões de pessoas. O que nos permite concluir que, apesar de ainda ter-se valores muito altos, a redução do prejuízo é significativa tanto em orçamento quanto em recursos humanos,

com uma "economia" de pouco mais de 50% em cada uma destas variáveis analisadas, somente pelo fato de ser utilizada uma metodologia para Gestão de Projetos.

Em suas pesquisas, o PMI® divide os problemas em três grupos (MOLENA, 2011):

1) Restrição tripla (escopo, prazo e custo);

2) Habilidades interpessoais;

3) Gerência/Diretoria

Molena apresenta os principais problemas encontrados nos projetos dividido entre esses três grupos:

Problema	Grupo	2009 % (posição)	2008 % (posição)	2007 %	2005 %	2004 %
Comunicação	2	**76 (1º)**	58 (3º)	64	71	61
Não cumprimento de prazos	1	71 (2º)	62 (1º)	66	72	66
Mudanças constantes de escopo	1	70 (3º)	59 (2º)	62	69	64
Escopo não definido adequadamente	1	61 (4º)	53 (4º)	60		
Concorrência entre o dia-a-dia e o projeto na utilização dos recursos	3	52 (5º)	43 (7º)	-		41
Estimativas incorretas ou sem fundamento	1	52 (6º)	38 (10º)	32		
Riscos não avaliados corretamente	3	50 (7º)	47 (5º)	48		59
Não cumprimento do orçamento	1	50 (8º)	42 (8º)	42		53
Problemas com fornecedores	3	37 (9º)	30 (11º)	27		41
Retrabalho em função da falta de qualidade do produto	3	28 (10º)	19 (13º)	26		
Recursos humanos insuficientes	3	16 (15º)	45 (6º)	51	62	60
Mudanças de prioridade constantes ou falta de prioridade	3	14 (17º)	35 (9º)	37		57

Figura 16 - Problemas que ocorrem com mais frequência na Gestão de Projetos
Fonte: (2009 *apud* MOLENA, 2011); (2008 *apud* MOLENA, 2011); (2007 *apud* MOLENA, 2011); (2005 *apud* MOLENA, 2011); (2004 *apud* MOLENA, 2011)

Diante desses problemas e números é importante não responder indevidamente, pois nem sempre há uma resposta ou caminho únicos a serem seguidos. No entanto, é necessário refletir sobre algumas questões referentes a cada um dos grupos de problema:

A) **Grupo 1**: trata da restrição tripla, podemos refletir sobre o por que independente da experiência dos profissionais utilizados e/ou equipes formadas, é bastante comum não atingir pelo menos uma das três restrições que parecem ser tão exatas como escopo, prazo e custo. A complexidade assumida para projetos de *software* são condizentes com a realidade? A natureza e dinamicidade do negócio envolvendo sistemas de informação são tão previsíveis e imutáveis no decorrer de projetos de médio (6 meses a 2 anos) e longo (maior que 2 anos) prazo?

B) **Grupo 2**: o item Comunicação também é visto como de grande importância para o sucesso de um projeto também em outras pesquisas, como a do Professor e Mestre Hélio Yasuki Seki, autor da pesquisa de mestrado intitulada "Um estudo interdisciplinar da Maturidade Corporativa para adoção de projetos tecnologicamente viáveis com a linha de pesquisa: Inteligência Coletiva e Ambientes Interativos". Essa pesquisa constata a importância da interdisciplinaridade entre os temas para se obter melhores resultados, tanto na aplicação como na solução de problemas. (MOLENA, 2011). Não basta ser puramente técnico, é muito importante aprimorar diversas habilidades em diversas áreas do conhecimento, desenvolver as chamadas Inteligências Múltiplas (GARDNER, 1995). A comunicação não deve ser negligenciada, tendo em vista que é em muitos casos a causa de insucessos, e o alinhamento entre as expectativas deve ser sempre revisto no decorrer de qualquer projeto, afim de que ajustes eventuais possam ser feitos, direcionando o projeto ao sucesso.

Curiosamente, nas pesquisas de "Habilidades mais valorizadas pelas organizações no gerenciamento de projetos" feita pelo PMI® entre 2006 e 2009, a habilidade de Comunicação fica em 2º lugar, atrás apenas da Liderança. Em outra pesquisa sobre "Habilidades que as organizações consideram mais deficientes", também realizada pelo PMI® no mesmo período, Comunicação ficou em 1º lugar nesses 4 anos. (MOLENA, 2011).

C) **Grupo 3:** para este grupo as perguntas para reflexão são mais diretas, como por que um gestor alocou menos recursos para um determinado projeto? Por que há super alocação de alguns recursos enquanto alguns ficam ociosos? Todos os riscos descobertos no decorrer do projetos eram passíveis de previsão, prevenção ou mitigação antes do início do projeto? Se os fornecedores são criteriosamente contratados, por que apresentam problemas? A formação de profissionais é adequada? Assumindo que sim, por que há tanto retrabalho? O planejamento foi feito de forma a garantir a qualidade e atendimento das expectativas assumidas? Por que prioridades mudam tanto em Sistemas de Informação? (MOLENA, 2011).

4.2 Metodologias Ágeis como solução

As pesquisas tanto do *The Standish Group* quanto do PMI® não são voltadas para análise da eficiência nos resultados, de sucesso ou não em um projeto de Desenvolvimento de *Software*, entre Metodologias Transacionais ou Metodologias Ágeis. Porém, ambas as pesquisas indicam resultados que possibilitam perceber diversos pontos favoráveis à adoção de Metodologias Ágeis como caminho para o sucesso.

No *Chaos Report* (do *The Standish Group*), por exemplo, dentre os quatro fatores principais para a melhora do índice de sucesso entre os números apresentados nos relatórios de 2009 e 2011, dois estão relacionados às Metodologias: 1) aumento da utilização de processos ágeis no desenvolvimento de *software* pelas empresas; 2) diminuição da participação (relativa ao total do número de projetos) dos projetos que utilizam o processo em cascata, já que o número de projetos que utilizam as Metodologias Tradicionais crescem em menor número e possuem taxa de sucesso menor do que os projetos que utilizam as Metodologias Ágeis.

Já o estudo do PMI®, afirma que o índice de sucesso melhora ao se utilizar alguma metodologia de Gestão de Projetos; no entanto, não especifica qual. Neste mesmo conjunto de pesquisas é apontado três grupos principais de problemas encontrados em projetos: 1) Restrição tripla (escopo, prazo e custo); 2) Habilidades interpessoais; e 3) Gerência/Diretoria. Para esses três grupos de problemas, podemos encontrar nas Metodologias Ágeis respostas satisfatórias em termos de soluções, como as citadas nos exemplos a seguir, utilizando o *Scrum* como referência.

Quanto a restrição tripla, para o escopo, o *Scrum* busca eliminar as dúvidas e ambiguidades entre as partes (fornecedor e cliente) na definição do escopo através da utilização do artefato conhecido como *User Stories*; já o prazo é tratado de forma menos flexível, já que a duração da *Sprint* é pré-definida e acordada entre as partes; e o custo também é menos impactado, uma vez que caso algo não saia conforme o esperado, o prejuízo será de somente a duração de uma *Sprint*, ao contrário de um projeto na Metodologia Tradicional que somente será conferido ao seu término.

Sobre as Habilidades interpessoais, o principal problema identificado é o fator Comunicação, sendo que esse é um dos pontos fortes das Metodologias Ágeis, sobretudo do *Scrum*. Já foi citado o quanto o uso do artefato *User Stories* pode ser facilitador na comunicação do desejo do cliente e no entendimento do fornecedor*;* fora isso, artefatos como *Burndown Chart*, atualizado diariamente, posiciona os envolvidos sobre o andamento do

projeto e ajuda a equilibrar expectativas sobre o andamento das atividades. Entre a equipe de desenvolvimento o quadro *Kanban* contribui para que todos saibam com quem está cada atividade e facilita o processo de sinergia entre a equipe, para que atividades dependentes sejam construídas com o real envolvimento dos principais conhecedores de cada assunto, aproveitando dessa forma as melhores características e conhecimentos de cada integrante. A reunião diária permite que não avance um processo sobre o qual haja dúvida e permite um melhor gerenciamento da equipe com uma percepção real do andamento das atividades previstas e realizadas. As reuniões de revisão, *Review* e Retrospectiva da *Sprint*, ajudam a alinhar o que não deu certo e criar um conhecimento coletivo, aumentando gradativamente a eficiência da equipe.

Por último, os itens decorrentes do terceiro grupo de problema (Gerência/Diretoria) também podem ser auxiliados e facilitados pelas Metodologias Ágeis: a duração relativamente curta das *Sprints* permite um melhor gerenciamento dos recursos entre os projetos, além de manter a motivação, concentração e foco na atividade. O retrabalho também pode acontecer ao utilizar-se as Metodologias Ágeis, no entanto, como o ponto de verificação acontece em um intervalo de tempo menor, consequentemente o prejuízo/desperdício é menor e também o direcionamento para o rumo certo é feito em menor tempo. Mudanças de prioridades acontecem sempre, independente da Metodologia utilizada. Entretanto, como a *Sprint* é curta e sempre deve-se focar na entrega e no agregar valor dentro de cada uma caso uma prioridade seja alterada, o prejuízo se limitará à *Sprint* em questão apenas.

5. CONCLUSÃO

Foram apresentados ao decorrer da pesquisa dados estatísticos que corroboram a problemática, que foi o ponto de partida deste trabalho: o alto índice de insucesso em projetos de desenvolvimento de *software*.

Baseando-se nos dados apresentados, bem como nos conceitos e definições dos diversos autores e pesquisadores utilizados como fontes, é possível chegar a algumas conclusões:

As Metodologias Tradicionais não foram concebidas para Desenvolvimento de *Software*, mas sim para outras Engenharias, como, por exemplo, a Engenharia Civil. O principal motivo para essa afirmação é a questão da complexidade envolvida nesses processos, onde para se construir um prédio, por exemplo, parte-se do requisito para que seja possível definir o que se quer, como se quer, como será construído, com que materiais etc, sendo que o *know-how* necessário é totalmente dominado. Em contrapartida, quando se trata de um requisito de *software* é muito comum e inerente ao processo que hajam mudanças no meio do caminho, como, por exemplo, que o cliente simplesmente mude de ideia no meio do processo, queira que seja utilizada uma tecnologia diferente (que eventualmente possa estar fora do *know-how* da equipe), entre outras tantas possíveis alterações no escopo que podem vir a ocorrer.

No entanto, não se pode afirmar que a utilização de Metodologias Tradicionais não seja uma alternativa para Desenvolvimento de *Software*, uma vez que há tempos vem sendo utilizadas e com relativo sucesso. Mesmo não sendo o método mais eficiente e apropriado, Metodologias Tradicionais funcionam para Desenvolvimento de *Software*.

As Metodologias Ágeis, por outro lado, são focadas para o Desenvolvimento de *Software* e são claramente voltadas para a realidade e particularidades de desenvolvimento de Sistemas de Informação, apesar de poderem ser aplicadas em outras realidades. Todavia, ainda possuem alguns pontos falhos, como a gestão de custos e gestão de riscos, por exemplo, e por isso a complementariedade com outras Metodologias também é importante. Estes pontos deficientes na Metodologia Ágil podem e devem ser geridos por práticas mais focadas nesses assuntos como, por exemplo, o PMBOK®.

A principal vantagem das Metodologias Ágeis é que, assumindo que o cenário está em constante transformação quando se trata de Desenvolvimento de *Software*, saem na frente das

Abordagens Tradicionais, por serem mais adaptáveis a responder à mudanças, uma vez que o ponto de checagem é muito menor e não se espera que todo o projeto seja concluído para só então verificar se é satisfatório e agregou valor para o cliente, se está dentro do esperado, ou simplesmente será útil ou trará algum benefício ao negócio.

A Metodologia Tradicional e a Metodologia Ágil são complementares e a utilização de uma abordagem pode restringir a utilização de outra. Porém, se combinadas, agregam valor e aumentam o índice de sucesso em projetos.

O item Comunicação é um fator importante e relevante para o sucesso de um projeto. Além da abordagem e da Metodologia utilizada, as pesquisas apresentadas apontam também para um problema antigo e conhecido: Comunicação é um fator crítico.

Por exemplo, o *Chaos Report* aponta que em 2009, 25% das falhas são por requisitos incompletos e falta de envolvimento do usuário. (2009, *apud* SOUZA, 2010). Isso significa que há um ciclo vicioso de falha no planejamento, onde técnicas de gestão simples poderiam produzir resultados interessantes, por exemplo, no caso do produtor (fornecedor) construir sem planejamento adequado e consequentemente fazer com que o produto final não seja o que o cliente necessita. Essas práticas hoje presentes no mundo corporativo não tem dinamismo e desperdiçam tempo, esforço e investimentos financeiros, sendo que o simples ato de planejar com cuidado diminuindo a velocidade, sendo mais assertivo, checando o canal de comunicação utilizado e se a mensagem pretendida chegou ao seu destino da forma que se intencionou em sua origem já proporciona maior qualidade ao Projeto. Com a comunicação mais próxima durante todo o processo, desde o início até o fim, o planejamento se torna menos estressante e as expectativas são atendidas de ambos os lados.

O *Scrum* é a Metodologia Ágil em voga na atualidade. Ao estudá-lo de forma mais aprofundada, percebe-se que na verdade não é uma receita única, mas sim um conjunto de práticas que podem ser aplicadas individualmente ou em conjunto. Em um estágio de "maturidade *Scrum*" avançado, os utilizadores farão uso das suas técnicas e práticas de forma natural sem que se perceba que está fazendo uso de uma metodologia inovadora, porque não mais o será, fará parte da cultura e terá a sensação de que sempre foi feito desta forma.

Quanto a adoção do *Scrum*, a principal dificuldade é a resistência natural à mudança e ao desconhecido. Apesar disso, tem potencial para ser uma metodologia amplamente difundida, primeiramente nos meios acadêmicos e posteriormente pelas empresas que apostam e investem em inovação.

Como sugestão de continuidade de estudos, acredito ser válido o aprofundamento em outras Metodologias Ágeis que foram tratadas superficialmente neste trabalho, na sua totalidade ou algumas de suas técnicas, como, por exemplo, o *Extreme Programming* (XP) e sua Programação em Par e Desenvolvimento Orientado a Testes. Quanto ao estudo do *Scrum*, acredito ser muito relevante investir tempo em trocar ideias com pessoas que implementam, gerenciam ou simplesmente participam de processos *Scrum* afim de ampliar os conhecimento afim de ser capaz de enxergar oportunidades de aplicar tais técnicas no dia-a-dia. Dependendo do nível de interesse no assunto, é possível também investir em formação formal, através de cursos e certificações.

REFERÊNCIAS BIBLIOGRÁFICAS

ALVIM, Paulo. **Scrum Certificado (MPS.BR F): A experiência da Powerlogic.** Recife-PE, 2008. Disponível em: <http://imprensa.cesar.org.br/scrum/SPINRecife_Mai08_PowerLogic.pdf>. Acesso em: 16 fev. 2013.

BECK, Kent. Programação Extrema (XP) Explicada - Acolha as Mudanças. Porto Alegre-RS, Bookman, 2004.

BECK, Kent et al. **Manifesto Ágil**. 2001. Disponível em: <http://manifestoagil.com.br/>. Acesso em: 22 Abr.2012

BLOG DA LOCAWEB, 2009. **Defendendo o modelo Waterfall de desenvolvimento.** Publicado em: 11 mar. 2009. Disponível em: <http://blog.locaweb.com.br/tecnologia/defendendo-o-modelo-waterfall-de-desenvolvimento/>. Acesso em: 20 jan. 2013.

BOEHM, Barry. **A Spiral Model of Software Development and Enhancement,** 1986. Disponível em: <http://csse.usc.edu/csse/TECHRPTS/1988/usccse88-500/usccse88-500.pdf>. Acesso em: 20 jan. 2013.

CAPES. **Tabela de Áreas de Conhecimento**, 2012. Disponível em: <http://www.capes.gov.br/images/stories/download/avaliacao/TabelaAreasConhecimento_072012.pdf>. Acesso em: 06 abr. 2013.

CARDOSO, Anderson. **PMI – ACP A certificação Ágil do PMI**, 2012?. Disponível em: <http://www.slideshare.net/barcellosreis/pmiacp-a-certificao-agile-do-pmi>. Acesso em: 09 jan. 2013.

COHN, Mike. **Uma introdução ao SCRUM.** Disponível em: <http://www.mountaingoatsoftware.com/scrum/a-reusable-scrum-presentation>. Acesso em: 16 fev. 2013.

DALONSO, Fabio A. **Iniciando com Scrum: Uma visão geral do mais badalado framework de Gerenciamento de Projetos do momento.** 2010. Disponível em:

<http://www.slideshare.net/fdalonso/treinamento-scrum-v10#btnNext>. Acesso em: 16 fev. 2013.

DA SILVA, Meire Elen Alves. **Metodologia Ágil de Gerenciamento de Projetos – SCRUM (Monografia)**. Centro Estadual de Educação Tecnológica "Paula Souza". Faculdade de Tecnologia de Taquaritinga. Taquaritinga-SP, 2010. Disponível em: <http://pt.scribd.com/doc/46922421/monografia-meire-scrib>. Acesso em: 02 fev. 2013.

D'ÁVILA, Márcio. **Artigo: Sucesso de projetos atualizado.** Publicado em: 18 jun. 2011. Disponível em: <http://blog.mhavila.com.br/2011/06/18/sucesso-de-projetos-atualizado/>. Acesso em: 12 jan. 2013.

GARDNER, Howard. Inteligências Múltiplas: a Teoria na Prática. Porto Alegre: Artes Médicas, 1995.

GONÇALVES, Encarnação de Lourdes Bassoli Andreo. **Gerenciamento de Risco de Software: Um Modelo de Processo e uma Ferramenta.** Piracicaba-SP, 2006. Disponível em: <https://www.unimep.br/phpg/bibdig/pdfs/2006/BRPEXSPTWXUA.pdf>. Acesso em: 07 jan. 2013.

GOYAL, Sadhna. **Feature Driven Development. Agile Techniques for Project Management and Software Engineering.** Munich, 2007. Disponível em: <http://csis.pace.edu/~marchese/CS616/Agile/FDD/fdd.pdf >. Acesso em: 03 fev. 2013.

HAAS, Hugo. **História do Scrum.** 2010. Disponível em: <http://projeto-agil.blogspot.com.br/2010/08/03-historia-do-scrum.html>. Acesso em: 16 fev. 2013.

HEPTAGON. (s.d. A). **FDD – Processos.** Disponível em: <http://www.heptagon.com.br/fdd-estrutura>. Acesso em: 03 fev. 2013.

HEPTAGON. (s.d. B). **O que é FDD?** Disponível em: <http://www.heptagon.com.br/fdd-oque>. Acesso em: 03 fev. 2013.

KNIBERG, Henrik; SKARIN, Mattias. **Kanban e Scrum: obtendo o melhor de ambos.** 2009. Disponível em: <http://www.infoq.com/resource/minibooks/kanban-scrum-minibook/pt/pdf/KanbanEScrumInfoQBrasil.pdf>. Acesso em: 16 fev. 2013.

LIMA, Ester. **Burndown chart – Mede o progresso da sprint e dá indicativos do processo de trabalho da equipe.** 2012. Disponível em: <http://blog.myscrumhalf.com/2012/01/burndown-chart-medindo-o-progresso-de-sua-sprint-e-trazendo-indicativos-do-processo-de-trabalho-da-equipe/>. Acesso em: 16 fev. 2013.

MACAÚBAS, Igor; PEREIRA, Marcos. **SCRUM: Gestão ágil de projetos.** 2012? Disponível em: <http://www.slideshare.net/macaubas/seminario-scrum-presentation>. Acesso em: 16 fev. 2013.

MOLENA, Airton. **Artigo: Problemas persistentes em projeto. e-NEWS edição 1106 de Junho/2011**. Disponível em: <http://www.pmisp.org.br/enews/edicao1106/artigo_01.asp>. Acesso em: 12 jan. 2013.

NEGRAO, Eduardo. **Artigo: Afinal, o que é Engenharia de Software**, 2010. Disponível em: <http://www.portalarquiteto.com.br/afinal-o-que-e-engenharia-de-software/>. Acesso em: 13 jan. 2013.

PEREIRA, Guilherme Vota. **Metodologia Lean de Desenvolvimento de Software: Uma visão geral.** 2012? Disponível em: <http://www.ceavi.udesc.br/arquivos/id_submenu/387/guilherme_metodologia_lean_de_desenvolvimento_de_software___uma_visao_geral.pdf>. Acesso em 03 fev. 2013.

PMBOK®. A Guide to the Project Management Body of Knowledge – PMBOK® Guide – Fourth Edition, Project Management Institute (PMI®), USA. 2008.

PMI®-Brasil. (s.d. A). **Qual a certificação certa para você?** Disponível em: <http://brasil.pmi.org/brazil/CertificationsAndCredentials/WhichCertificationIsRightForYou.aspx>. Acesso em: 08 jan. 2013

PMI®-Brasil. (s.d. B). **Quem são os gerentes de projetos**. Disponível em: <http://brasil.pmi.org/brazil/AboutUS/WhoareProjectManagers.aspx>. Acesso em: 08 jan. 2013.

PMI®-Brasil. (s.d. C). **Valores Fundamentais**. Disponível em: <http://brasil.pmi.org/brazil/AboutUS/CoreValues.aspx>. Acesso em: 08 jan. 2013.

PMI®-SP, 2012 A. **Entrevista: João Gama Neto, PMP, PMI-AC**. **E-NEWS Edição abr. 2012** Disponível em: <http://www.pmisp.org.br/enews/edicao1204/entrevista_01.asp>. Acesso em: 09 jan. 2013.

PMI®-SP, 2012 B. **Entrevista: Ricardo G. Costa, PMP fala sobre como foi a II Jornada Agile**. Disponível em: <http://www.pmisp.org.br/noticias/entrevista-ricardo-g-costa-pmp-fala-sobre-como-foi-ii-jornada-agile>. Acesso em: 09 jan. 2013.

PMI®-SP, (s.d.) **O instituto**. Disponível em: <http://www.pmisp.org.br/institucional/pmi/o-instituto>. Acesso em: 07 jan. 2013.

ROYCE, Winston W. Managing. **The Development of Large Software Systems,** 1970**.** Disponível em: <http://www.cs.umd.edu/class/spring2003/cmsc838p/Process/waterfall.pdf>. Acesso em: 20 jan. 2013.

SCHWABER, Ken. **Agile Project Management With Scrum: Microsoft Press**, 2004

SCHWABER, Ken; SUTHERLAND, Jeff. **Guia do Scrum – Um guia definitivo para o Scrum: As regras do jogo.** 2011. Disponível em: <http://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum%20Guide%20-%20Portuguese%20BR.pdf#zoom=100>. Acesso em: 10 fev. 2013.

SILVA, Tiago de Farias. **Compondo Métodos Ágeis de Desenvolvimento de Software (Dissertação)**. Recife-PE, 2009. Disponível em: <http://www.cin.ufpe.br/~tg/2009-1/tfs.pdf>. Acesso em: 03 fev. 2013

SOARES, Michel dos Santos. Metodologias Ágeis Extreme Programming e Scrum para o Desenvolvimento de Software. Revista Eletrônica de Sistemas de Informação, Conselheiro Lafaiete-MG. 2004. Disponível em: <http://revistas.facecla.com.br/index.php/reinfo/article/view/146>. Acesso em: 10 Mar.2012

SOUZA, Washington. **Artigo: Chaos Report: Como esta a TI no mundo?** Publicado em: 09 ago. 2010. Disponível em: <http://www.blogcmmi.com.br/geral/chaos-report-como-esta-a-ti-no-mundo>. Acesso em: 12 jan. 2013.

SILVA, Edilberto. (s.d) **Visão Geral sobre Engenharia de Software.** Disponível em: <http://www.edilms.eti.br/uploads/file/engenharia/aula01-introducao-engenharia-software.pdf >. Acesso em: 24 abr. 2013.

STEFFEN, Juliana Berossa. **Lean para desenvolvimento de Software.** 2011. Disponível em: <https://www.ibm.com/developerworks/mydeveloperworks/blogs/rationalbrasil/entry/lean_para_desenvolvimento_de_sw_o_que__c3_a9_isso_afinal12?lang=en>. Acesso em: 03 fev. 2013.

TEIXEIRA, Daniel Dinis. PIRES, Fernando Jorge Afonso. SOUSA, José Pedro Gaiolas de. SANTOS, Tiago Alexandre Gonçalves Pereira. **DSDM – Dynamic Systems Development Methodology.** Disponível em: <http://paginas.fe.up.pt/~aaguiar/es/artigos%20finais/es_final_14.pdf >. Acesso em: 02 fev. 2013.

TELES, Vinícius Manhães. **Um estudo de caso da adoção das práticas e valores do Extreme Programming.** Dissertação de Mestrado, Universidade Federal do Rio de Janeiro, 2005. Disponível em <http://www.improveit.com.br/xp/dissertacaoXP.pdf>. Acesso em: 02 fev. 2013.

MACKENZIE, Universidade Presbiteriana. **Apresentação de trabalhos acadêmicos: guia para alunos da Universidade Presbiteriana Mackenzie** / Universidade Presbiteriana Mackenzie. – 4. ed. – São Paulo : Ed. Mackenzie, 2006.

Wikipedia®, (s.d. A). **Dynamic Systems Development Method.** Disponível em <http://pt.wikipedia.org/wiki/Dynamic_Systems_Development_Method>. Acesso em: 02 fev. 2013.

Wikipedia®, (s.d. B). **Feature Driven Development.** Disponível em: <http://pt.wikipedia.org/wiki/Feature_Driven_Development>. Acesso em: 03 fev. 2013.