FUNDAÇÃO DE ENSINO "EURÍPIDES SOARES DA ROCHA"
CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA - UNIVEM
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

ANDERSON VEIGA RODRIGUES

Framework para Gerência de Projetos em Ambientes Ágeis

MARÍLIA-SP
2012

ANDERSON VEIGA RODRIGUES

FRAMEWORK PARA GERÊNCIA DE PROJETOS EM AMBIENTES ÁGEIS

Trabalho de Curso apresentado ao Curso de Bacharelado em Sistemas de Informação da Fundação de Ensino "Eurípides Soares da Rocha" mantenedora do Centro Universitário Eurípedes de Marília - UNIVEM, como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Orientador:
Prof. Ms. Fábio Lucio Meira

MARÍLIA-SP
2012

RODRIGUES, Anderson Veiga

Framework para Gerência de Projetos em Ambientes Ágeis /Anderson Veiga Rodrigues; orientador: Fabio Lucio Meira. Marília, SP: [s.n.], 2012.

51f.

Trabalho de Curso (Graduação em Sistemas de Informação) – Curso de Sistemas de Informação, Fundação de Ensino "Eurípides Soares da Rocha", mantenedora do Centro Universitário Eurípides de Marília –UNIVEM, Marília, 2012.

1. Controle 2. Processos 3. Software 4. Gerência de projetos 5. Métodos Ágeis 6. Framework

CDD:

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO – AVALIAÇÃO FINAL

Anderson Veiga Rodrigues

Ferramenta CASE para a Implementação e Implantação de Processos de Software

Banca examinadora da monografia apresentada ao Curso de Bacharelado em Sistemas de Informação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Sistemas de Informação.

Nota: 6,0 (seis – AC

Orientador: Fabio Lucio Meira

1º. Examinador: Giulianna Marega Marques

2º. Examinador: Ricardo José Sabatine

Marília, 10 de dezembro de 2012.

AGRADECIMENTOS

Primeiramente, gostaria de agradecer a Deus e aos meus Pais, por sempre me darem força e sustentação nos momentos mais difíceis de minha vida, e por fazerem parte deste sonho que estou realizando de estar me formando em uma Universidade.

Agradeço também a minha irmã, que apesar de ser mais nova, procuro sempre me espelhar em certos detalhes por ser sempre determinada e possuir garra para conseguir o que deseja, e por ter me ajudado muito em conselhos e informações que contribuíram em minha formação acadêmica neste momento.

A minha namorada, que sempre foi companheira e sempre me ajudou nos momentos que precisei, com conselhos, trabalhos e compreensão. Aos amigos e professores, em especial ao professor Fabio Lucio Meira que me propôs e me deu respaldo com total ajuda no projeto do TCC, e a todos que estiveram comigo nesta longa caminhada, onde sei que muitos deles estarão comigo em minha próxima jornada, após a formação acadêmica.

Muito Obrigado.

“A imaginação é mais importante que a ciência, porque a ciência é limitada, ao passo que a imaginação abrange o mundo inteiro.”
Albert Einstein

RODRIGUES, Anderson Veiga. **Framework para Gerência de Projetos em Ambientes Ágeis.** 2012. Trabalho de curso (Bacharelado em Sistemas de Informação) – Centro Universitário Eurípedes de Marília, Fundação de Ensino "Eurípides Soares da Rocha", Marília, 2012.

RESUMO

O processo de software inclui todas as atividades envolvidas no desenvolvimento de software, é um conjunto de sequência de atividades desempenhadas durante o desenvolvimento que geram produtos de trabalho como resultado. Existem muitos tipos de diferentes sistemas, onde cada um requer ferramentas de softwares adequadas e técnicas de engenharia para o seu desenvolvimento. Por ser ágil e multidisciplinar, facilita muito o entendimento e acompanhamento do projeto entre as equipes. Propõe-se neste projeto a proposta de um Framework de gerência de projetos em ambientes ágeis para facilitar e agilizar esse controle nas empresas que utilizam esta metodologia, proposta esta, de desenvolver algo diferenciado atendendo a qualquer demanda de uma forma ampla e inteligente, pois atualmente os usuários precisam se adequar aos sistemas já existentes e este modelo propõe-se que o sistema se adeque aos usuários conforme regras criadas pelos responsáveis pelo projeto.

Palavras – chave: Controle, Processos, Software, Gerência de projetos, Métodos Ágeis, Framework.

RODRIGUES, Anderson Veiga. **Framework for Project Management in Agile Environments**. 2012. Course work (Bachelor of Information Systems) - University Center Marília Euripides, Education Foundation "Euripides Soares da Rocha," Marilia, 2012.

ABSTRACT

The software process includes all activities involved in software development, is a set sequence of activities performed during development that generate work products as a result. There are many different types of systems, where each one requires tools and appropriate software engineering techniques for its development. For being agile, multidisciplinary greatly facilitates the understanding and monitoring of the project between the teams. It is proposed this project a proposed Framework project management in agile environments to facilitate and streamline this control in companies that use this methodology, this proposal, to develop something different serving any demand for a broad and intelligent, as currently users need to adapt existing systems and this model proposes that the system fits according to rules created by the users responsible for the project.

Keywords: Control, Process, Software, Project Management, Agile Methods, Framework.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

PMBOK: PROJECT MANAGEMENT BODY OF KNOWLEDGE

PMI: PROJECT MANAGEMENT INSTITUTE

COBIT: CONTROL OBJECTIVES FOR INFORMATION AND RELATED TECHNOLOGY

PO: PLANEJAR E ORGANIZAR

AI: ADQUIRAR E IMPLEMENTAR

DS: ENTREGAR E SUPORTAR

ME: ENTREGAR E AVALIAR

TI: TECNOLOGIA DA INFORMAÇÃO

SM: SCRUM MASTER

OPENUP: OPEN UNIFIELD PROCESS

XP: EXTREME PROGRAMMING

UML: UNIFIELD MODELING LANGUAGE

ISO: INTERNATIONAL ORGANIZATION FOR STANDARDIZATION

OGC: OFIICE OF GOVERNMENT COMMERCE

ISACF: INFORMATION SYSTEMS AUDIT AND CONTROL FUNDATION

BPEL: BUSINESS PROCESS EXECUTION LANGUAGE

YAM: AIN`T MARKUP LANGUAGE

DSL: DOMAIN SPECIFIC LANGUAGE

EAV: ENTIDADE ATRIBUTO VALOR

SUMÁRIO

CAPÍTULO 1 – INTRODUÇÃO

1.1 Contexto

Processo é uma receita seguida no desenvolvimento de um projeto, onde o projeto concretiza uma abstração, que é o processo. Processos de software são as diversas fases necessárias para produzir e manter um produto de software. Necessitam de organização lógica de diversas atividades técnicas e gerenciais, envolvendo agentes, métodos, ferramentas, artefatos e restrições que possibilitam disciplinar, sistematizar e organizar o desenvolvimento e manutenção de produtos de software.

O processo de software executado pela empresa deve ser descrito e representado em um modelo descritivo (manual ou guia do processo de software). Os modelos de processo de software formam a base para o entendimento e análise do processo de software, possibilitam a melhoria do processo existente, permitem re-projetar ou complementar os processos existentes formando uma base para mudanças ou disseminação do conhecimento sobre o processo de software.

No entanto, poucas empresas têm adotado essa abordagem como uma estratégia para compreender, disciplinar e melhorar seus processos de software. Os problemas enfrentados por uma das empresas que tive a vivência de presenciar que não estava preocupada com determinados métodos foram:

Má distribuição de tarefas entre as equipes;
Atrasos nos desenvolvimentos dos projetos;
Falta de qualidade nos artefatos entregues, devido à falta de planejamento;
Falta de controle de entrega de artefatos (Etapas do processo);
Falta de confiança por parte dos clientes (Atrasos e má qualidade);
Descontrole do projeto quanto ao seu escopo;
Descontentamento dos clientes.

Além de aumentar a qualidade do software, o uso de um processo de desenvolvimento de software, aumenta a produtividade das equipes de desenvolvimento (PRESSMAN, 2006).

É importante também considerar as características da própria empresa, a fim de que o processo seja adequado às suas necessidades e atenda as suas expectativas.

As relações de todas as tarefas requeridas como, métodos utilizados, qualificação, ferramentas, capacitação e motivação das pessoas envolvidas no projeto, devem ser consideradas em um processo de software. Com isso, os custos de um processo de software bem definido caem drasticamente, aumentam sua produtividade e possuem uma melhor qualidade de software (ROCHA, WEBER e MALDONADO, 2001). Atualmente no mercado tornou-se uma necessidade das empresas para sua sobrevivência investir em melhorias e métodos ágeis para aumentar sua competitividade.

Como sabemos, em um processo de software ocorrem frequentes erros. Com isso, algumas atividades como a Garantia de Qualidade de Software devem ser inclusas no ciclo de vida do projeto. Pois consiste na qualidade final do software desenvolvido.

Pressman (2006) ressalta ainda, que informações qualitativas e quantitativas sobre processo devem ser avaliadas e divulgadas para obtenção de um padrão de qualidade de software. Com isso, um processo em desenvolvimento sob controle estático podem obter melhores custos e melhor qualidade em seu cronograma de desenvolvimento.

1.2 Objetivo Geral

O objetivo geral deste trabalho é conceituar a importância da gerência de projetos e as metodologias ágeis aplicadas.

Demonstrar uma proposta de Framework para Gerência de Projetos, seu fluxo, melhorias, habilidades e agilidade no gerenciamento de múltiplos projetos com a estratégia entre as equipes para que o projeto tenha um processo eficaz de qualidade e ágil.

1.3 Objetivo Especifico

- Buscar conceituar o processo de software e identificar a definição de suas diferentes aplicações;
- Demonstrar o processo de software em sistemas de informação a partir do detalhamento de um processo de softwares;
- Conceituar a gerência de projetos em ambientes ágeis, seu papel, sua importância e suas habilidades para que o processo seja seguido conforme o cronograma, tendo grande agilidade, facilitando a comunicação entre os envolvidos no projeto, produzindo um software com qualidade diminuindo os riscos e em um curto espaço de tempo.

1.4 Motivação

Atualmente destaca-se pela sua importância nos dias atuais, onde, tornou-se uma grande necessidade para a própria sobrevivência das empresas no mercado a demanda de produtos de software, dentro de seus limites esperados de cronograma, custo, agilidade e qualidade. Ou seja, garantindo a qualidade em seu software, menos custo de desenvolvimento (retrabalho), gerando contentamento do cliente, prazo de entrega entre outros. Fazendo com que o aumento de sua competitividade junto às demais empresas no mercado seja reconhecido.

1.5 Estrutura da Monografia

Neste capitulo, foi demonstrada uma visão geral destacando o contexto, motivação e justificativa como uma proposta para desenvolvimento de um Framework para gerência de projetos em ambientes ágeis. Os demais capítulos correspondem as seguintes estruturações:

No capítulo 2 são abordados os processos de desenvolvimento de software como um todo. Serão apresentados alguns modelos de processos de software como (Cascata, Espiral, Iterativo e Incremental e de Prototipagem), seus ciclos de vidas e suas vantagens e desvantagens.

No capítulo 3 são abordados alguns tipos de guias de gerência de projetos (PMBOK, ITIL, COBIT e SCRUM), suas definições e organizações para desenvolvimento e gerenciamento em TI.

No capítulo 4 são abordados as Gerência de projetos em ambientes ágeis como (OPENUP, XP), suas práticas e conceitos como um processo de desenvolvimento.

No capítulo 5 são abordados uma Proposta de um Framework de gerência de projetos em ambientes ágeis, seus princípios, funcionamentos, estrutura e finalidades no auxilio ao gerenciamento de projetos.

No capítulo 6 são abordados as conclusões finais desta dissertação, focando na proposta de desenvolvimento de um Framework de gerência de projetos em ambientes ágeis, para futuras pesquisas e/ou desenvolvimento.

CAPÍTULO 2 – PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE

2.1 Considerações Iniciais

Neste capítulo serão abordados conceitos fundamentais para a compreensão e o desenvolvimento do trabalho proposto. Estes embasamentos teóricos apresentados se referem aos principais pesquisadores e estudiosos da área.

Na seção 2.2 apresenta a funcionalidade de um processo de desenvolvimento de software e qual sua organização lógica para o andamento do processo. Na seção 2.3 são abordados os componentes de um processo de software para o andamento do projeto onde definem a responsabilidade de cada pessoa envolvida no projeto. Na seção 2.4 são abordados as fases dos processos de softwares, seu ciclo de vida e as atividades que constituem para se obter um produto de software. Na seção 2.5 são apresentados alguns dos modelos de ciclo de vida de processos de software. Na subseção 2.5.1 é abordado as funcionalidades do modelo cascata e suas desvantagens. Na subseção 2.5.2 é abordado as funcionalidades do modelo espiral, como suas vantagens e desvantagens. Na subseção 2.5.3 é abordado as funcionalidades do modelo iterativo e incremental com suas vantagens de utilização. Na subseção 2.5.4 é abordado o modelo de prototipagem, sua funcionalidade, seus benefícios segundo (Pressman) e algumas considerações de utilização deste modelo.

2.2 Processos de Software

Processo é um conjunto de atividades ordenadas, cujo objetivo é atingir uma meta: Entregar um software eficiente de qualidade e que atinja sua necessidade de negócios.

Howard Baetjer Jr., em seu livro "*Software as Capital"*, faz o seguinte comentário sobre processo de software:

> "… Desde que o software, como todo capital, é conhecimento incorporado, e como esse conhecimento está inicialmente disperso, tácito, latente e incompleto na sua totalidade, o desenvolvimento de software é um processo de aprendizado social. O processo é um dialogo no qual o conhecimento, que deve se transformar em software é reunido e incorporado ao software. O processo fornece interação entre usuários e projetistas, entre usuários e ferramentas em desenvolvimento e entre projetistas e ferramentas em desenvolvimento (tecnologia). É um processo iterativo no qual a própria ferramenta

> serve como meio de comunicação, com cada nova rodada de dialogo explicitando mais conhecimento útil do pessoal envolvido…”.
> (Baetjer 1998, pág. 85)

Ou seja, processo é uma receita seguida no desenvolvimento de um projeto, onde o projeto concretiza uma abstração, que é o processo. Processos de software são as diversas fases necessárias para produzir e manter um produto de software. Necessitam de organização lógica de diversas atividades técnicas e gerenciais, envolvendo agentes, métodos, ferramentas, artefatos e restrições que possibilitam disciplinar, sistematizar e organizar o desenvolvimento e manutenção de produtos de software.

2.3 Componentes de Processos de Software

Todo e qualquer Processo é composto por alguns componentes para o andamento do projeto. Estes componentes formam o resultado de uma atividade, uma tarefa ou conjunto de tarefas executadas, auxiliam na execução de atividades e definem a responsabilidade de cada pessoa envolvida no projeto, denominados como:

Artefato: Resultado de uma atividade desenvolvida (Módulo implementado, cujo foi testado e aceito). Quando entregue ao cliente se torna um produto.

Insumo: Elemento necessário para desenvolvimento de uma tarefa e/ou atividade do projeto. Pode ser um elemento de outras tarefas elaboradas ou elemento de saída de outras atividades.

Tarefas: São determinadas ações desempenhadas por uma pessoa junto ao projeto em desenvolvimento, monitorando e/ou realizando essas ações. Além disso, consome recursos, como equipamentos e tempo da pessoal envolvida.

Atividades: É um conjunto de tarefas com objetivo de levar a artefatos. Representam evidências no desenvolvimento, permitindo controle e qualidade do resultado.
São mini projetos onde possuem inicio e fins bem definidos produzindo artefatos e critérios estabelecidos.

Ferramenta e equipamentos: Ajudam a auxiliar na execução de atividades e/ou tarefas podendo automatizar algumas partes da execução minimizando erros e agilizando os processos.

Papel: São as responsabilidades que as pessoas envolvidas nos processos têm e como elas se comportam junto ao mesmo. É importante também ressaltar que papéis não são pessoas, pessoas executam papeis.

2.4 Fases do Processo de Software (Ciclo de Vida)

Em cada fase de um processo de software são executadas as atividades básicas para que sejam atingidos os objetivos propostos. Segundo Schwartz , Pressman e Sommerville estas atividades constituem um conjunto mínimo para se obter um produto de software.

Especificação de Requisitos: Estabelecer uma solução geral para o problema, envolvendo questões extra-software. Levantamento das necessidades do software a ser implementado. Produzir uma especificação de requisitos, que convencionalmente é um documento. Incluir um plano de testes para verificar adequação.

Projeto de Sistema: Desenvolver um modelo conceitual para o sistema, composto de módulos mais ou menos independentes, onde cada módulo tem sua interface de comunicação estudada e definida, os módulos em si são definidos, e possivelmente traduzidos para pseudo-código.

Programação (Codificação): Desenvolvimento do sistema que controla e realiza a computação e lógica. Implementação em si do sistema em uma linguagem de computador.

Verificação e Integração: Realização de testes para verificar a presença de erros e comportamento adequado a nível das funções e módulos básicos do sistema. Reunião dos diferentes módulos em um produto de software homogêneo, e a verificação da interação entre estes quando operando em conjunto.

Nesta fase, o software em geral entra em um ciclo iterativo que abrange todas as fases anteriores (Manutenção e Evolução).

Não existe uma sequencia obrigatória dessas fases, pois diversos autores defendem que o processo é muito mais iterativo e cíclico do que a ideia de fases simples pode sugerir.

Todo e qualquer software é desenvolvido, independente de plataforma e tecnologia , seguindo uma sequencia básicas de grandes atividades, conforme (Figura 2.1).

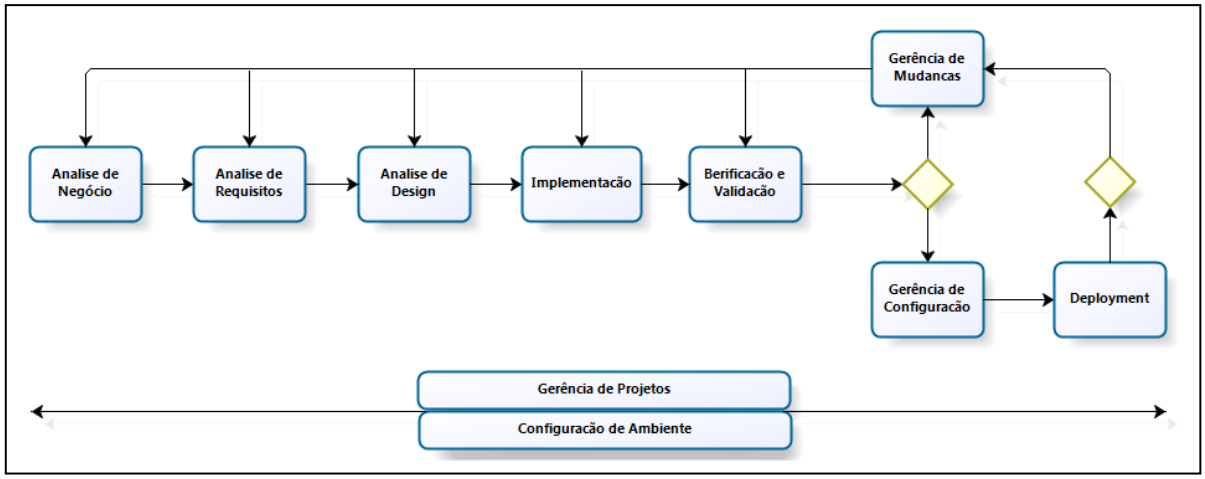

Figura 1 – Ciclo de vida de um Projeto de desenvolvimento de software (Fonte própria)

2.5 Modelos de Ciclo de Vida

Modelo de processos de softwares é uma representação abstrata de um processo. Devem ser aplicados e personalizados conforme necessidade especifica. Alguns desses modelos são os modelos:

- Cascata
- Espiral
- Iterativo e Incremental
- Prototipagem

2.5.1 Modelo Cascata

Criado por (Royce em 1970) seus principais subprocessos são executados em sequência, seus pontos de controles são bem definidos, tornando este processo confiável podendo ser utilizado em qualquer escala de projetos.

Porém este processo se torna rígido e burocrático, pois não são permitidos erros. Por este motivo suas atividades e análise de requisitos devem ser bem elaboradas. Tornando o modelo em cascata em um modelo de baixa visibilidade ao cliente, pois o mesmo somente perceve o resultado final do projeto.

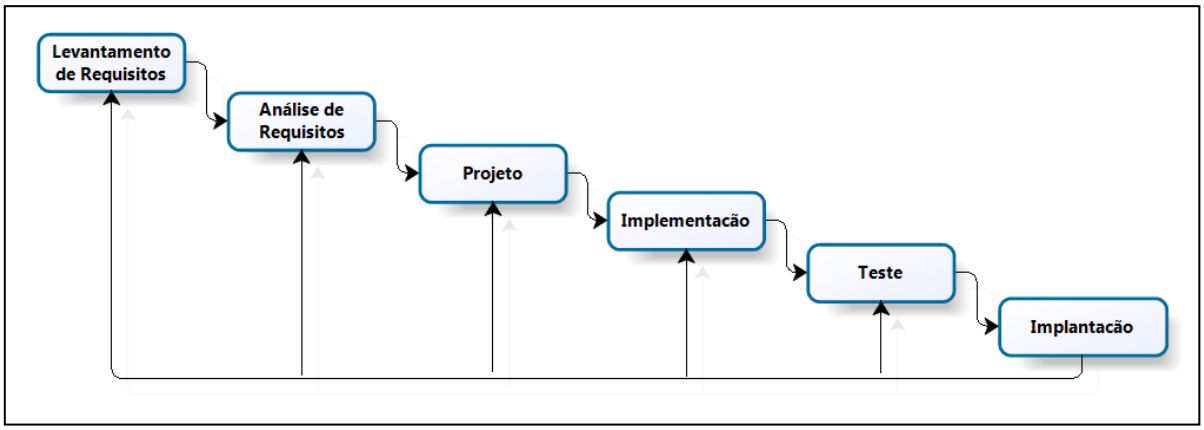

Figura 2 – Modelo de ciclo de vida Cascata (Fonte própria)

Levantamento de Requisitos: Entender o que o cliente deseja para que o desenvolvedor do projeto tenha junto ao mesmo a compreensão do que necessita ser realizado.

Análise de Requisitos: Refinamento das informações do projeto tratado pelo analista junto ao cliente, saber o que é pedido e se o que foi solicitado é viável ou não seu desenvolvimento.

Projeto: Adaptar o modelo de análise realizada como embasamento para desenvolvimento do projeto proposto.

Implementação: Por em prática (execução) o projeto desenvolvido.

Teste: Verificação do software para identificar divergências, irregularidades ou aprovação do mesmo.

Implantação: Fase em que o sistema é implantado junto ao cliente, entrando em execução.

Desvantagens no modelo Cascata:

- Grande dificuldade na realização de alterações quando o processo já encontra-se em execução;
- Tornam-se inflexíveis as partições por estágios de projetos distintos;
- Grande dificuldade de alterações junto aos requisitos estabelecidos;
- Para os modelos sequenciais e lineares os reais projetos raramente se adaptam;
- Requisitos capturados de uma só vez são bem difíceis de serem realizados;

- Somente o resultado do projeto final é apresentado ao cliente;
- Ocorrem atrasos frequentes sem a necessidade junto aos programadores;
- Custo elevado para correções em fases de Testes e Implantação;

2.5.2 Modelo Espiral

Criado em 1988 por (Boehm) para integrar os diversos modelos existentes à época. Tem como objetivo entregar um produto (Software) mais próximo à necessidade e/ou realidade do esperado pelo cliente. Isso, porque é muito semelhante ao PDCA, ferramenta gerencial que auxilia a tomada de decisões, garantindo alcances de metas e objetivos.

Trabalha em cima de versões para o desenvolvimento do projeto em pequenos ciclos. Sua maior vantagem é possuir um maior controle sobre os riscos que o projeto possa sofrer. Tornando assim, o processo mais seguro e dinâmico.

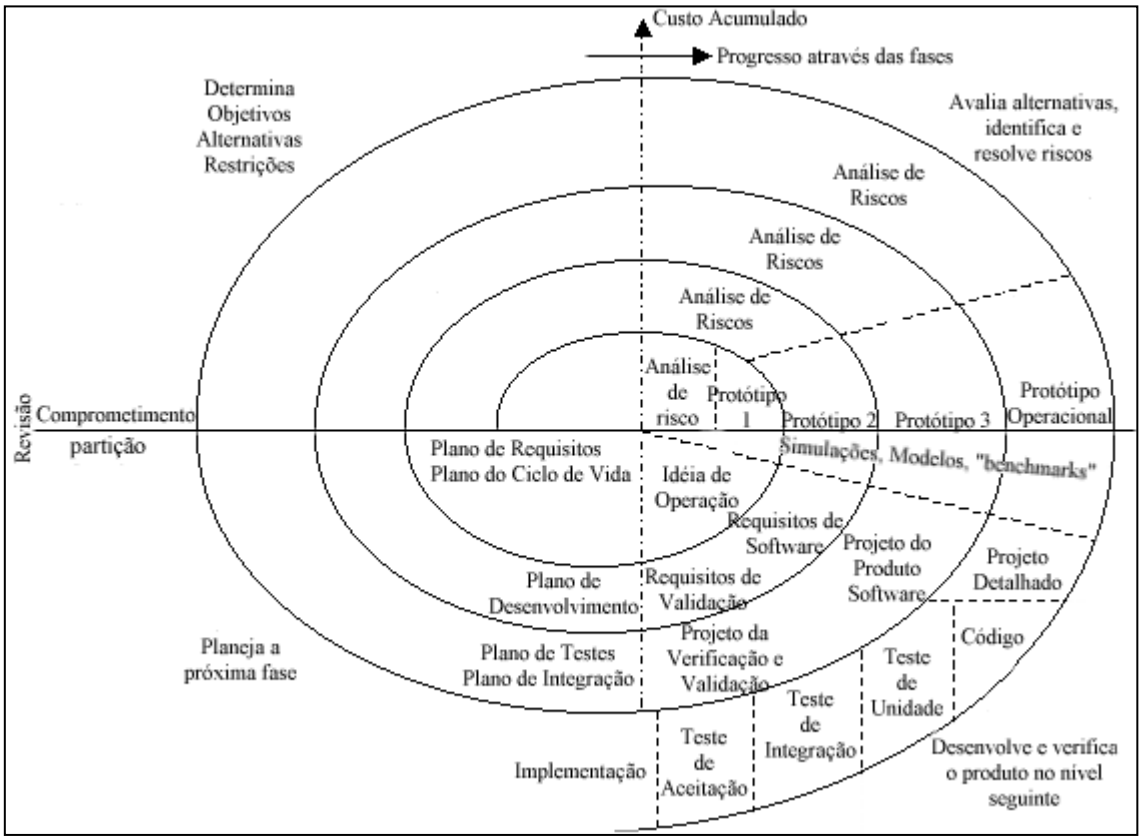

Figura 3 – Modelo de ciclo de vida espiral (Fonte Pressman (2010))

Primeiramente é definido detalhadamente os requisitos do projeto. Realizando entrevistas com usuários (Externos e/ou Internos) do projeto.

Após a definição do sistema, é feita uma análise de todas as alternativas para o desenvolvimento, como estratégias para implementação das alternativas e seu custo-benefício.

Caso ocorram eventuais riscos no projeto, para encontrar uma solução eficaz de lidar com alterações nos requisitos já estabelecidos, é utilizada a prototipagem.

Após esta análise, o primeiro protótipo é desenvolvido utilizando o desenho preliminar. Porém, ainda não se trata da versão completa e sim de uma versão reduzida, cujo representa uma aproximação do produto final.

Após a analise do primeiro protótipo, realizadas em pontos fortes, fracos e riscos que o projeto pode conter, o desenvolvimento do segundo protótipo pode ser realizado.

O processo continua sendo realizado em espiral, até a finalização de todo o sistema em desenvolvimento. Os bugs dos protótipos anteriores são anexados aos próximos protótipos para suas devidas correções.

Vantagens no modelo Espiral:

- A cada nova iterações são realizadas versões mais completas do sistema em desenvolvimento. Reduzindo os riscos do projeto.
- Permite um refinamento mais completo do processo de desenvolvimento seguido pelo modelo cascata.

Desvantagens no modelo Espiral:

- Baseia-se na experiência de avaliação de riscos do projeto em desenvolvimento para o sucesso, pois se um risco grave do projeto não for encontrado, poderá trazer vários problemas futuros.
- O protótipo pode não cumprir com todos os requisitos de desempenho, ficando incompleto ou refletir em determinados aspectos. Ou seja, podem ocorrer diferenças entre o protótipo e o projeto final.
- São necessários usos de técnicas para realizar a sincronização dos cronogramas para processos e determinar indicadores de custo, pois várias partes do projeto podem estar sendo desenvolvidas ao mesmo tempo, sendo abordados de modo diferenciados.

2.5.3 Modelo iterativo e incremental

Se refere a uma extensão do modelo espiral, mais formal e rigoroso. Basicamente funciona com o desenvolvimento repetidos chamados de ciclos (Iterativo), porém executado um de cada vez em pequenas partes (Incremental). Aumentando iterativamente suas versões fazendo com que o sistema se evolua até sua implementação.

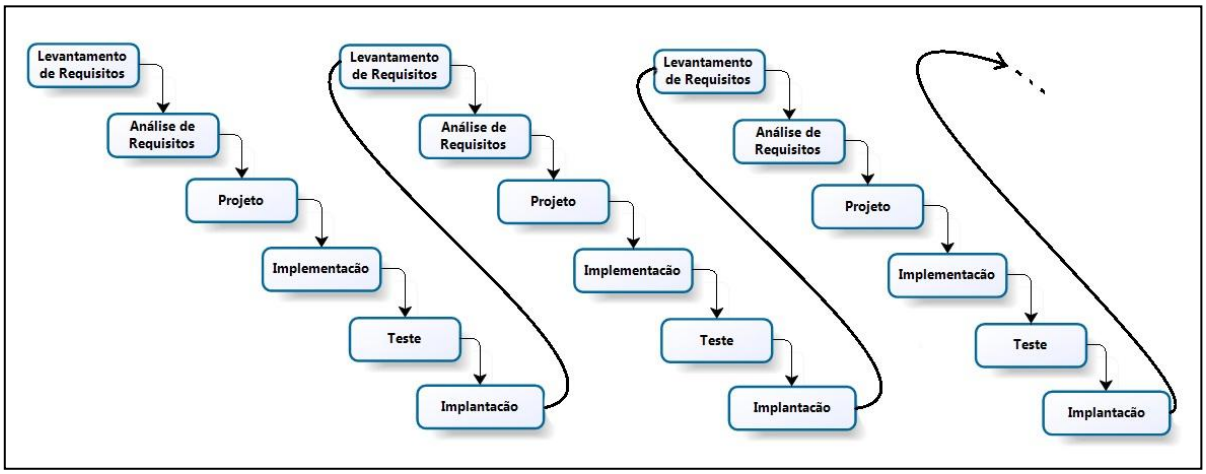

Figura 4 – Modelo Iterativo e Incremental (Fonte própria)

O modelo Iterativo e Incremental pode ser agrupado nas seguintes fases:

Fase de Iniciação: Em níveis mais elevados se dispõem os riscos do escopo do projeto em desenvolvimento.

Fase de Elaboração: Satisfação dos requisitos não funcionais e identificação dos riscos na arquitetura de trabalho.

Fase de Construção: Para o desenvolvimento, são preenchidos os componentes incremental produzidos através das análises já realizadas do projeto, testes dos requisitos até a implementação.

Fase de Transição: Corresponde a entrega do projeto final para produção (Implementação).

Vantagens no modelo Iterativo e Incremental:

- Possibilita identificar medidas eliminando e/ou encontrando riscos do projeto.
- Obtém redução de riscos com relação a custos, onde é perdido somente iterações e não valor do produto.

- Melhor conjunto de regras, controlando solicitações de alterações futuras junto ao projeto em desenvolvimento.

2.5.4 Modelo Prototipagem

O resultado é obtido rapidamente através deste método de desenvolvimento, onde se obtém um feedback já no inicio do projeto. Com isso, é possível a detecção de problemas pode ser realizada com grande rapidez fazendo com que a resolução dos riscos possam ser realizadas antes da finalização do projeto em desenvolvimento.

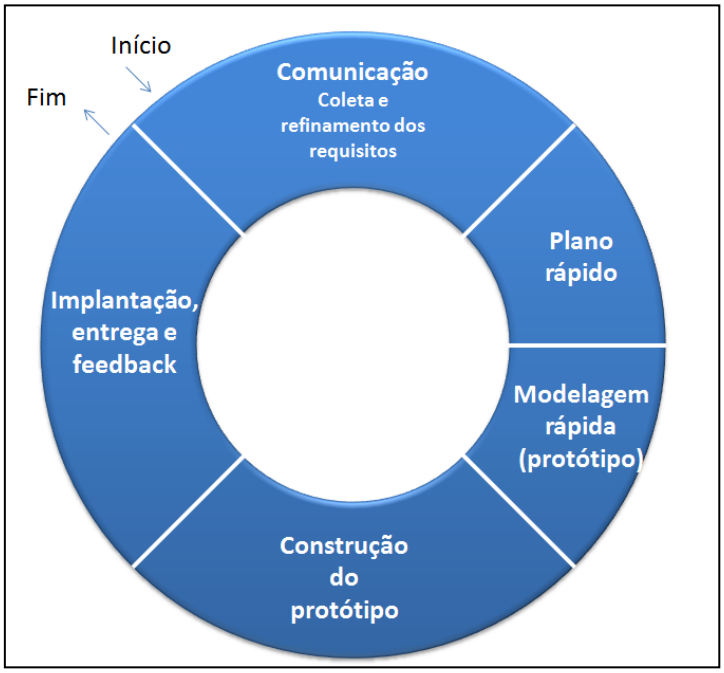

Figura 5 – Modelo Prototipagem (Fonte Pressman (2010))

Segundo Pressman este modelo pode trazer os seguintes benefícios:

- O modelo é muito interessante para projetos de grandes portes onde representam dificuldades nos requisitos.
- Possibilita obter uma verão de como será o projeto que será desenvolvido com um pequeno investimento inicial.
- Experiências obtidas no processo de produção do protótipo desenvolvido, podem reduzir custos das próximas fases do projeto.

Questões a serem consideradas quanto à utilização do modelo:

- A prototipação pode ser utilizada somente para usuários que participaram do projeto, caso contrário não é adequado à utilização do mesmo.
- Durante todo o processo de prototipação, deve-se ter uma boa análise e não se descuidar da mesma.
- Deve-se esclarecer ao usuário que nem sempre o desempenho do protótipo será o mesmo do sistema final.

CAPÍTULO 3 – GUIA DE GERÊNCIA DE PROJETOS

3.1 Considerações Iniciais

Neste capítulo serão abordados conceitos fundamentais de alguns guias de gerência de projetos.

Na seção 3.2 apresenta a funcionalidade do PMBOK, seus grupos integrados e suas funcionalidades. Na subseção 3.2.1 são abordados a divisão dos projetos do PMBOK divididos em 5 grupos e seu diagrama relacionado graficamente e sinteticamente em 42 processos de gerenciamento. Na seção 3.3 apresenta as funcionalidades do ITIL, seus módulos e conceitos para gerenciamento de TI. Na seção 3.4 apresenta as funcionalidades do COBIT, suas áreas de focos e suas práticas para o controle de governança em TI. Na subseção 3.4.1 são apresentados os princípios básicos do COBIT. Na subseção 3.4.2 são apresentados os domínios do COBIT e sua cobertura sobre os demais processos. Na seção 3.5 apresenta as funcionalidades do SCRUM, seus processos e seu ciclo de vida. Na subseção 3.5.1 é apresentado a utilização do SCRUM junto ao KanBan tornando o processo extremamente ágil.

3.2 PMBOK

O PMBOK (Project Management Body of Knowledge Guide) é um padrão de projetos americano desenvolvido pela PMI (Project Management Institute). Apesar de ser confundido por muitos como uma metodologia, o PMBOK não pode ser considerado uma, pois não diz o que fazer, é apenas uma referência básica.

Possui uma ótima definição de vocabulário comum entre os profissionais que trabalham com gestão de projetos.

Vargas (2003), afirma que as áreas de gerenciamento de projetos descrevem seus componentes e seus gerenciamentos de processos, onde os processos podem ser organizados em nove grupos integrados.

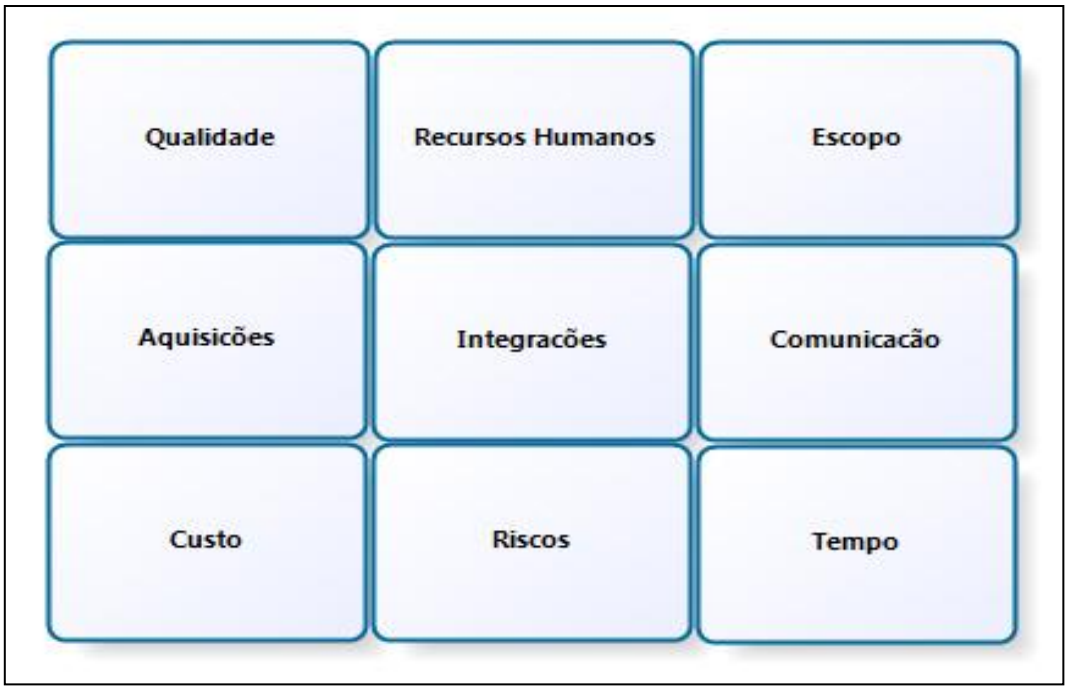

Figura 6 – Nove áreas de conhecimento do PMBOK (Fonte própria)

Escopo: Garantir que todo o trabalho necessário, somente o necessário seja realizado.

Riscos: Minimizar as ameaças do projeto em desenvolvimento e aumentar as oportunidades.

Tempo: Estimativa de duração do projeto e sequência das atividades envolvidas (Cronograma).

Recursos Humanos: São os papeis, responsabilidade, formação e desenvolvimento da equipe contida no projeto.

Comunicações: Identificar e manter as partes interessadas no projeto em desenvolvimento.

Custos: Definir e controlar o orçamento junto ao projeto, identificar recursos dentre outros.

Qualidade: Garantir o total atendimento das necessidades obtidas para as quais o projeto em desenvolvimento foram criados.

Aquisições: Gerenciar a contratação de Bens e Serviços de Terceiros.

Integração: Coordenar todas as áreas abrangentes no projeto, garantindo as entregas.

3.2.1 Grupos PMBOK e sua divisão de projetos

O guia PMBOK propõe uma divisão dos projetos em 5 grupos de processos.

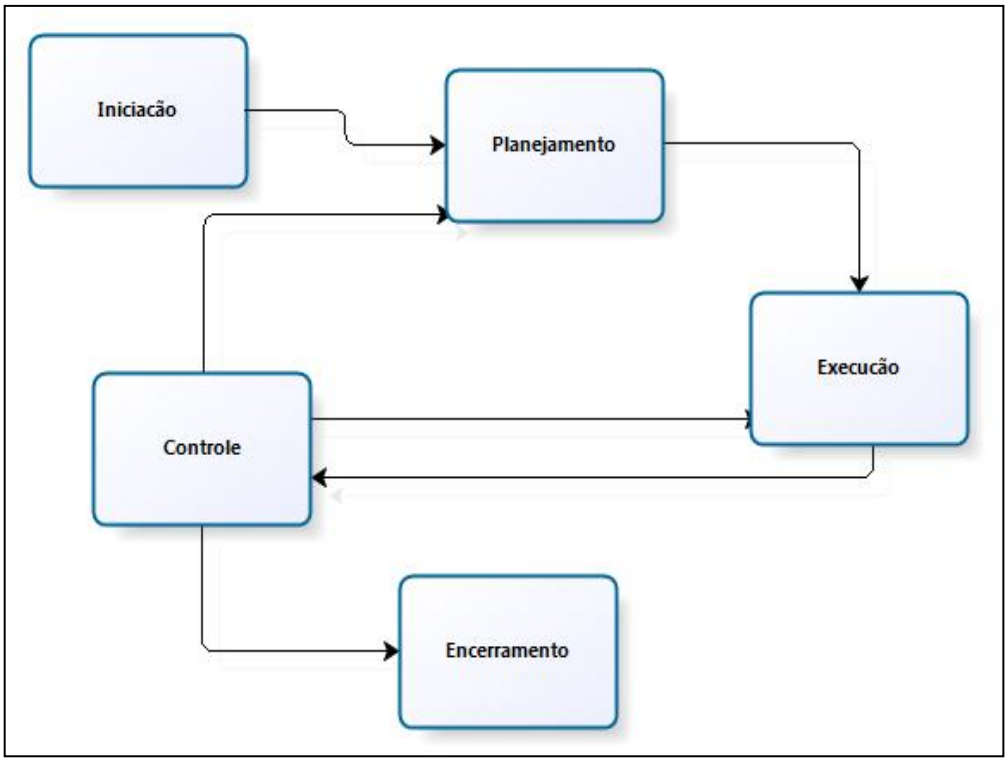

Figura 7 – Grupos de Processos do PMBOK (Fonte própria)

- **Iniciação –** Autorização do projeto e das fases contidas junto ao mesmo.
- **Planejamento –** Definir, refinar e selecionar as alternativas do projeto em desenvolvimento.
- **Execução –** Coordenar pessoas e recursos do projeto.
- **Controle –** Monitorar, corrigir e garantir resultados do projeto.
- **Encerramento –** Documentar o histórico e realizar a formalização da aceitação do projeto e fases do mesmo.

Mauro Sotille (2009) propõe um diagrama relacionado graficamente e sinteticamente 42 processos de gerenciamento descrito no PMBOK 4ª Edição, indicando cinco grupos dos quais são distribuídos os processos com suas respectivas áreas de conhecimento associadas a cada um.

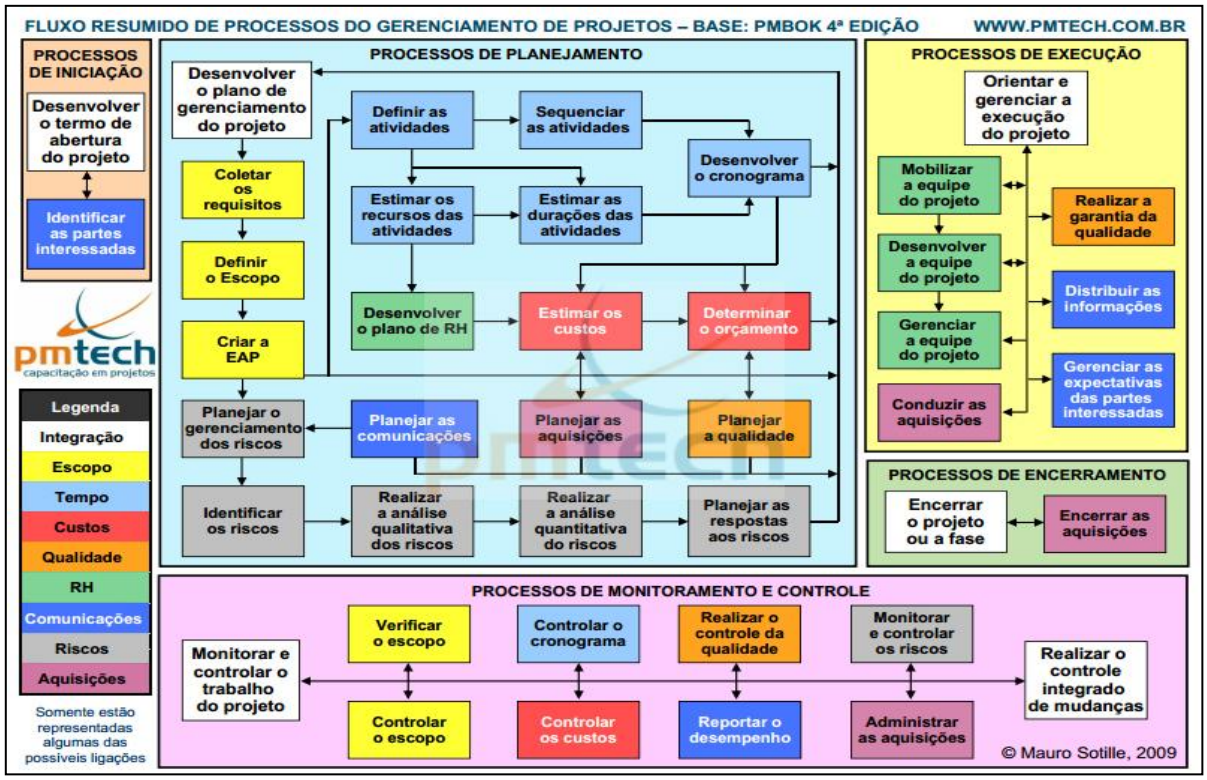

Figura 8 – Fluxo de processos do gerenciamento de projetos do PMBOK (Fonte PMBOK 4ª Edição2)

Quadro 1 – Grupo de Processos do PMBOK (Fonte própria)

Grupo	Número de Processos
1- Iniciação	2
2- Planejamento	20
3- Execução	8
4- Controle	10
5- Encerramento	2
Total	**42**

3.3 COBIT

Criado pela ISACF (Information Systems Audit and Control Fundation) o framework possui modelos através de práticas e padrões mais maduros (ITIL, COSO e ISSO/IEC 17799). Não é um padrão definitivo, apenas serve como apoio para a implementação de controles na Governança de TI.

Com quase duas décadas de existência, o framework está em constante evolução para se aderir as melhores práticas que vem surgindo ao mercado mundial.

Segundo COBIT 4.1 (2010), as sustentações da governança de TI são:

Figura 9 – Área foco do COBIT (Fonte COBIT 4.1 (2010))

Alinhamento Estratégico – Seu foco é a governança de TI, visando alinhar os planos estratégicos, deixando a TI aderente aos objetivos de negócio. Ou seja, deve estar preparada para que qualquer requisito não seja um problema ou impasse para que ocorra a evolução do negócio, agregando valor junto aos produtos e serviços da empresa, ajudando no uso de recursos e auxiliando no posicionamento competitivo da empresa.

Entrega de Valor – Seu foco é garantir a entrega dos serviços que atendem os requisitos. Ou seja, realizar a entrega com qualidade dentro do prazo previsto, do custo estimado, são princípios básicos desta área.

Gestão de Recursos – Pode se entender como otimizar os custos e maximizar a utilização dos ativos de TI. Assegurar a capacidade suficiente de recursos para atender e suportar os negócios.

Gestão de Risco – Seu foco é realizar o entendimento dos riscos, probabilidades e seus impactos que são de grande importância. Obter um plano de ação auxilia o gerenciamento dos riscos controladamente deixando a empresa tranquila e mais segura.

Mensuração de Desempenho – Seu foco é monitorar e acompanhar a implementação da estratégia e dos projetos em andamento entre outros, realizando medições e extrações de indicadores de desempenho.

3.3.1 Princípios básicos do COBIT

Visando um maior alinhamento entre as áreas de negócios existentes e a tecnologia, para um melhor entendimento dos Stakeholders. O COBIT, fornece indicadores de desempenho para verificação se o caminho está sendo conforme o planejado. Os princípios do modelo que é totalmente voltado ao negócio são:

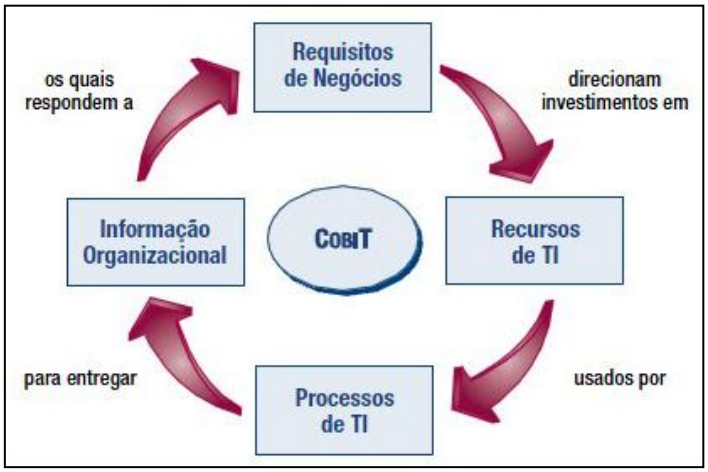

Figura 10 – Princípios básicos do COBIT (Fonte COBIT 4.1 (2010))

As informações existentes dentro da empresa tem que possuir critérios de aceite, proposto pelo modelo do COBIT. Pois através disso, os gestores podem tomar determinadas decisões e escolher rumos diferentes para sua organização caso necessário.

3.3.2 Domínios do COBIT

COBIT 4.1 (2010) realiza a divisão do framework COBIT em 4 domínios distribuídos e possui 34 processos (Figura 3.6).

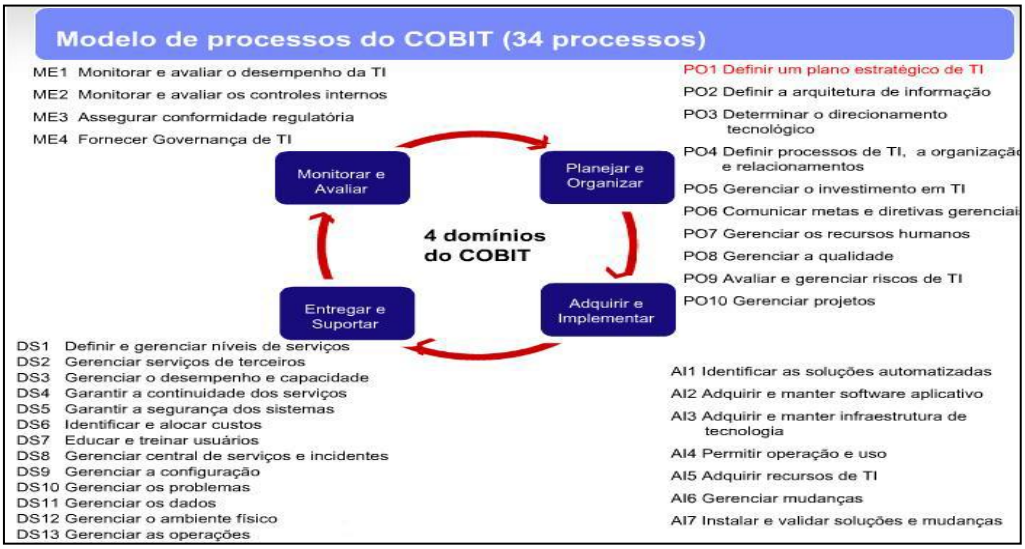

Figura 11 – Adaptado do ITGI, domínios e processos do COBIT ((Fonte COBIT 4.1 (2010))

Planejar e Organizar (PO): Auxiliar no alinhamento da estratégia de TI junto a de negócios, analisar se a empresa está obtendo um uso eficiente de seus recursos e deixar claro seus objetivos.

Adquirir e Implementar (AI): Garantir a continuidade dos ciclos de vida do projeto, pois mudanças e manutenções existem. Estabelecer a implementação e integração junto aos processos.

Entregar e Suportar (DS): Inclui gerenciamento de segurança, suporte aos serviços junto aos usuários, infra-estrutura operacional e gestão de dados.

Monitorar e Avaliar (ME): Assegurar a qualidade dos processos de TI, controlar os objetivos, através de mecanismos regulares com avaliações internas e externas.

Com o COBIT não há preocupação em como será implantado e sim em o que será implantado. Pois nele, entram vários outros frameworks como, PMI, ITIL, ISO27001 entre outros. Ou seja, o COBIT é um conjunto de processos dos mais variados frameworks, onde realiza a integração desses processos. Chamado como guarda chuva, devido cobrir vários outros processos dos mais variados, conforme demonstração da (Figura 3.7):

Figura 12 – Cobertura do COBIT em outros Frameworks (Fonte COBIT 4.1)

3.4 SCRUM

Criado em 1996 por Jeff Suttherland e Ken Schwabe, auxilia equipes no processo de desenvolvimento de um projeto de software de forma flexível em ambientes que sofrem constantes mudanças. Formado por uma equipe pequena e com uma comunicação entre todos os integrantes da equipe, apresenta uma abordagem flexível e produtiva.

Sua metodologia é semelhante as do XP, ou seja, com equipes pequenas, requisitos pouco estáveis ou desconhecidos e com iterações curtas entre os envolvidos no projeto, para obtenção de uma visibilidade no desenvolvimento.

É um processo ágil e leve que utiliza práticas iterativas e incrementais no controle e gerenciamento de desenvolvimento de software.

O Scrum possui um ciclo de vida baseado em três fases principais de acordo com (Abrahamsson, 2002).

Pré-planejamento: Cria-se uma lista de funcionalidades (Requisitos) do projeto, definição da equipe de desenvolvimento, ferramentas que serão utilizadas, possíveis riscos do projeto e necessidades de treinamentos. Realizando atualizações constantes na lista, caso ocorram novas alterações no projeto.

Desenvolvimento: Identificam-se variáveis onde serão consideradas em todo o projeto realizando flexibilidade e acompanhamento de mudanças. O projeto (Software) é desenvolvido em Sprints, com novas funcionalidades que são adicionadas e a cada Sprint é realizado um planejamento que não pode ultrapassar um mês.

Pós-Planejamento: São verificados se todas as funcionalidades estão conforme solicitações do cliente e se as funcionalidades foram implementadas corretamente. Realiza-se a liberação do produto ao cliente e são feitas etapas de integração, testes finais junto a ferramenta e documentação.

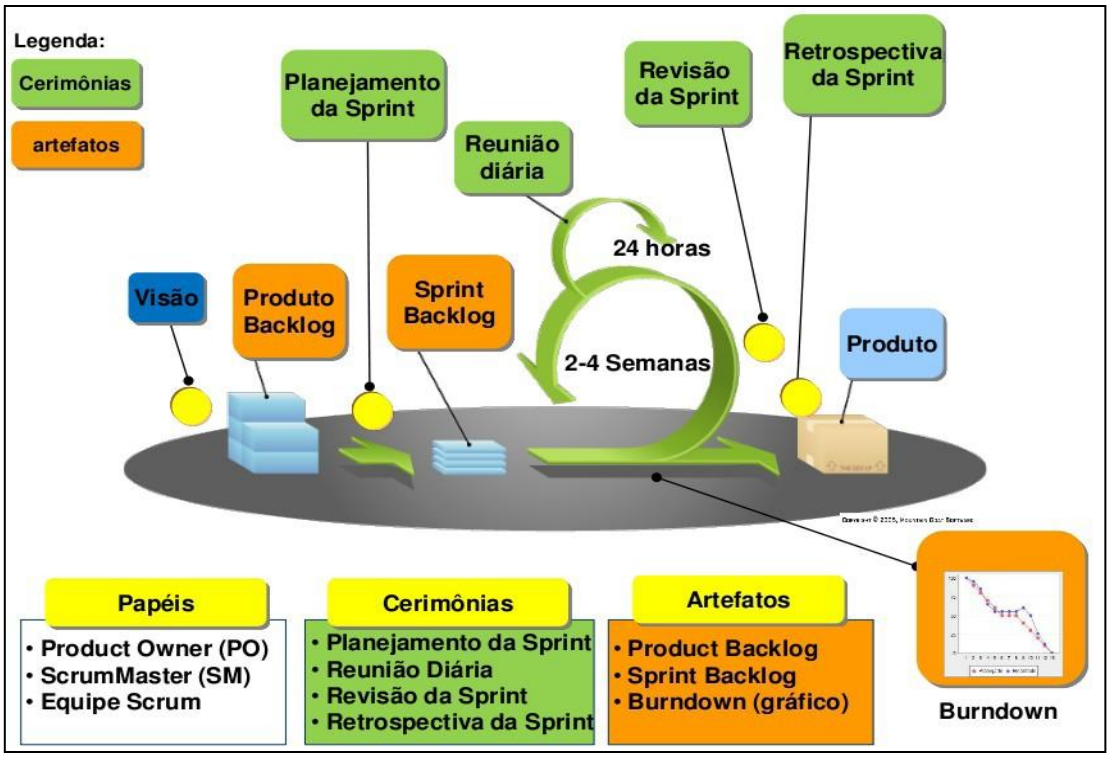

Figura 13 – Processo da metodologia Scrum (Mallmann, 2010)

O Scrum se baseia nos seguintes elementos:

Time e Papéis: Scrum Master; Product Owner e Time.

Artefatos: Backlog do Produto, Backlog da Sprint, Burndown e Incremento.

Eventos (Time Boxes) e Regras: Planejamento da Release, Planejamento da Sprint, Reunião Diária e Revisão da Sprint.

Times – Os times são compostos pelo Product Owner (Dono do projeto), equipe de desenvolvimento e Scrum Master (Lider do time). Escolhem a melhor forma de realizar e finalizar o projeto. Devem obter, criatividade, flexibilidade e produtividade. Ou seja, todos devem contribuir com o projeto, exigindo aprender novas habilidades ou lembrar-se de habilidades antigas.

Scrum Master – O Scrum máster é responsável pelo time, mas não gerencia e sim auto organiza o time. Pode ser um membro do time, desenvolvendo uma determinada tarefa na Sprint, mas nunca sendo o Product Owner . Seu papel é ajudar, proteger o time das interferências externas, remover barreias entre o desenvolvimento e garantir que o processo está sendo respeitado.

Product Owner – O Product Owner pode ser o financiador do projeto ou alguém interessado no mesmo. Seu papel é priorizar o product backlog, alterar prioridades fora da Sprint, aceitar ou não os resultados do projeto e definir as funcionalidades do mesmo.

Daily Scrum Meeting – São reuniões diárias que permite o acompanhamento e andamento do projeto em desenvolvimento, além disso, ajuda a identificar os obstáculos e impedimentos ocorridos no projeto. As reuniões são rápidas e duram em média quinze minutos.

Product Backlog – São as tarefas do produto, é o núcleo do scrum onde contem uma lista de prioridades dos trabalhos a serem realizados no projeto, descrito conforme solicitação do cliente.

Sprint – São intervalos pré definidos de aproximadamente trinta dias (um mês), mas também são comuns os sprints de 2 a 3 semanas. Nos sprints são executados todo o trabalho especificado no Product Backlog.

Sprint Backlog – São tarefas do Sprint, extraídas do Product Backlog e direcionadas ao próximo Sprint a ser realizado.

Sprint Planning – São planejamento do Sprint, ou seja, planeja todos os objetos e suas funcionalidades, priorizando os itens que serão desenvolvidos em cada sprint.

Sprint Review – É a revisão do Sprint, são realizadas reuniões no final de cada Sprint executada, afim de demonstrar os resultados. Além disso, são decididos as novas atividades que podem integrar o Product Backlog.

Burndown – São gráficos que registram os esforços restantes do Backlog ao longo do tempo.

3.4.1 Scrum em conjunto com o KanBan

O Scrum, utilizado junto ao KanBan se torna muito eficiente e extremamente ágil. Pois dividi-se o trabalho em partes, escrevendo cada item em papel e colando-os na parede, divididos em colunas nomeadas e demonstrando onde cada item encontra-se em seu fluxo de trabalho.

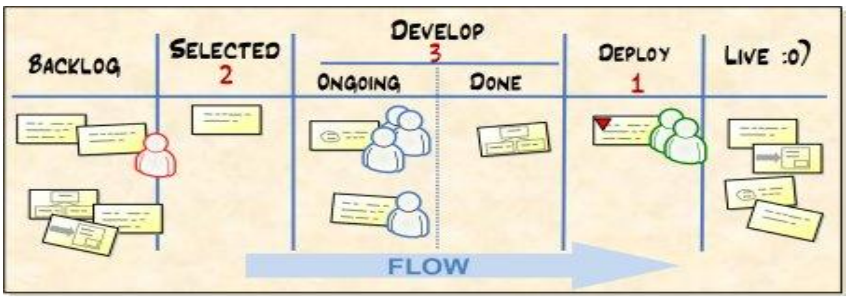

Figura 14 – Fluxo de trabalho Scrum x KanBan (Fonte Livro: Kanban e Scrum obtendo o melhor de ambos)

CAPÍTULO 4 – GERÊNCIA DE PROJETOS EM AMBIENTES ÁGEIS

4.1 Considerações Iniciais

Neste capitulo serão abordados alguns conceitos de gerência de projetos em ambientes ágeis.

Na seção 4.2 apresenta a fundamentação do Open UP, seus princípios básicos e seus ciclos de vidas. Na seção 4.3 apresenta as funcionalidades dos XP (Extreme Programming) e algumas práticas segundo (BECK, 2000).

4.2 OPEN UP

O Open UP (Open Unifield Process) foi criado pela IBM, e aborda as principais características do processo unificado, com práticas e métodos ágeis. Construído para apoiar o desenvolvimento de software orientado a objetos, para absorver os principais benefícios da modelagem unificada (UML).

Fundamentado por quatro princípios básicos, cujos mesmos seguem abaixo. Determinam que as organizações adotem os mesmos, caso desejam utilizar o Open UP como um processo de desenvolvimento.

Equilibrar as prioridades concorrentes e aumentar os valores dos Stakeholders: O desenvolvimento do software deve ser realizado em harmonia com as diferentes perspectivas que muitas vezes são concorrentes. Aumentar os benefícios aos Stackholders e que seja compatível com todas as restrições do projeto.

Colaborar para alinhar os interesses e compartilhar os conhecimentos: Auxiliar na colaboração e compreensão do projeto em desenvolvimento, possibilitar pessoas com interesse e habilidades para produzirem softwares eficientes.

Focar na evidenciação da arquitetura o mais cedo possível para minimizar os riscos e organizar o desenvolvimento: Alguns softwares são difíceis de serem evoluídos, com isso, é importante que os desenvolvedores alinhem seus interesses e conhecimentos reduzindo riscos e diminuindo o retrabalho.

Promover melhorias e fazer com que os Stakeholders tenham feedback do desenvolvimento: Em modo geral é impossível entender todas as necessidades do cliente, compreender todas as tecnologias e identificar todos os riscos do projeto. Com isso, ocorrem

muitas mudanças ao longo do ciclo de vida do projeto em desenvolvimento. E para obter um feedback continuo do projeto, divide-se o projeto em pequenas iterações com um tempo pré-determinado.

O ciclo de vida do Open Up também é dividido em quatro fases, conforme (Figura 4.1).

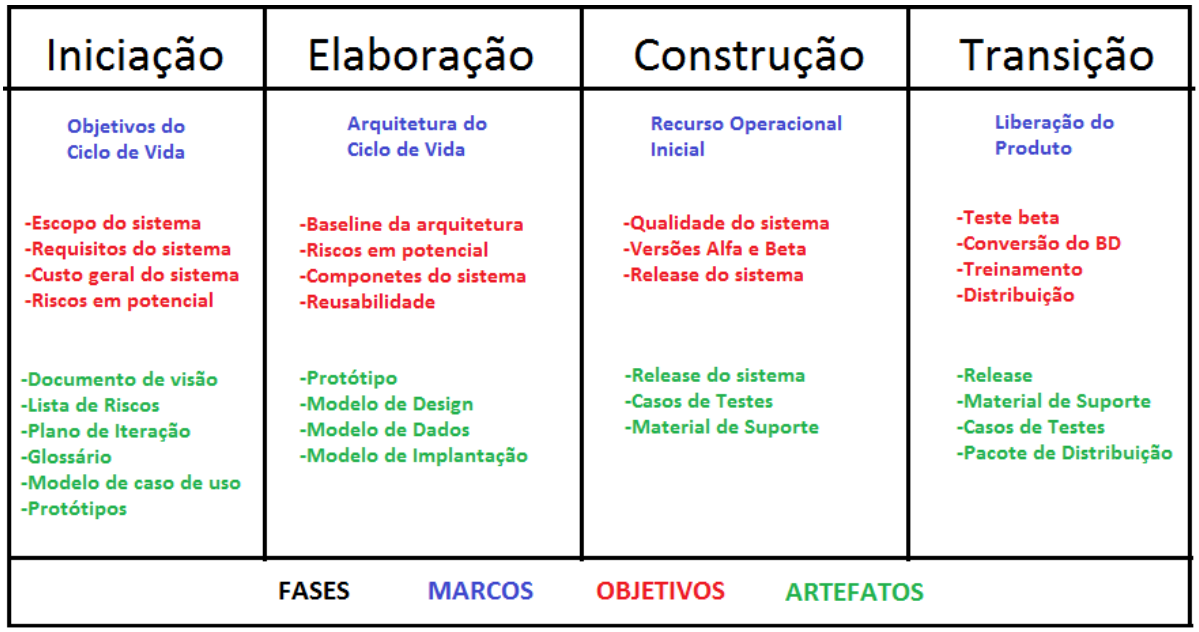

Figura 15 – Ciclo de Vida do Open UP (Fonte própria)

Iniciação (Concepção) – Foca e compreende o escopo e objetivos do projeto, viabiliza o projeto determinando o que deve ou não ser realizado.

Elaboração – Foca nos riscos de arquitetura do sistema. Onde os principais devem ser identificados criando uma arquitetura solida e utilizada como núcleo do sistema,

Construção – Foca no projeto, testes de funcionalidades do sistema, implementação e realiza continuamente o desenvolvimento do sistema.

Transição – Foca na segurança das funcionalidades do sistema, verificando se as mesmas foram implementadas e se as especificações foram atendidas.

O Open UP recomenda um mês para cada iteração, pois durante este tempo é permitido que novas funcionalidades sejam realizadas e entregues em um tempo razoável, havendo feedbacks de stakeholders frequentes. Essas iterações pode ser especificadas pelo

gerente de projeto quantas vezes forem necessárias a cada fase do projeto em desenvolvimento. Pois depende de diversos aspectos como (Tamanho do projeto, tecnologia que será utilizada, complexidade etc..).

4.3 XP (Extreme Programming)

Criado por Kent Beck e Ward Cunningham, o Extreme Programming conhecido como (XP) é o mais popular entre os métodos ágeis. Desenvolvido para atender projetos de software onde os requisitos sofrem mudanças frequentes,

As principais diferenças do XP estão na (Comunicação, Simplicidade, Feedback e Coragem) e doze práticas simples que são utilizadas pelas equipes de um projeto.

Comunicação: Seu fator chave é a comunicação entre o desenvolvedor e o cliente, pois com a participação direta do cliente junto ao projeto e com liberações de versões frequentes, os erros no projetos são menores.

Simplicidade: Desenvolvimento de um projeto de maneira mais simples com códigos claros sem funções desnecessárias. Ou seja, procurar desenvolver e implementar apenas requisitos atuais evitando incluir funcionalidades que podem ser importantes no futuro. Segundo (SOARES, 2004), é melhor fazer algo simples pensando hoje, e pagar um pouco mais amanhã para fazer modificações necessárias do que desenvolver algo complicado hoje e que não venha ser utilizado amanhã.

Feedback: Mantes sempre informado e atualizados o cliente, sobre o andamento do projeto, assim o cliente pode seguir novas características, evitando obstáculos entre o gerente do projeto, cliente e desenvolvedor.

Coragem: Permite suporte a simplicidade quando é percebido pela equipe do projeto que é possível simplificar o software, permitindo a construção de projetos mais simples. Necessita de coragem para implementar as etapas (Comunicação Simplicidade e Feedback) pois não são todas as pessoas envolvidas no projeto que possuem facilidade de comunicação e com relacionamento.

Segundo (BECK, 2000), o XP possui algumas práticas, cujas mesmas serão apresentadas a seguir:

- **Pequenas Versões:** Realiza atualização do sistema frequentemente em um ciclo pequeno de desenvolvimento.
- **Metáfora:** Evita termos técnicos facilitando o desenvolvimento e comunicação.
- **Planejamento (Jogo):** Utiliza estimativas de custos pelos desenvolvedores para verificação do que precisa ser desenvolvido e o que pode ser adiado junto ao projeto em desenvolvimento. Desta forma permite que o cliente verifica o que é mais importante para ser desenvolvido na versão atual e o que pode aguardar para a próxima versão do sistema.
- **Teste:** Nesta etapa são validados os projetos através de testes de unidade, testes funcionais conhecidos como (Caixa preta) onde não se vê o código fonte realizando verificações com as funções do sistema e não com a implementação do mesmo. Testes de aceitação também devem ser realizados pelo cliente assumindo a responsabilidade do mesmo.
- **Projeto Simples:** Enfatiza a simplicidade no desenvolvimento do software, incluindo apenas os requisitos atuais e não os requisitos futuros.
- **Refatoração:** Realizar alterações no sistema em mudar o comportamento do mesmo, removendo as duplicidades de códigos existentes, simplificando-o e melhorando a comunicação entre os desenvolvedores.
- **Diagramação em Pares:** Realização de elaboração de modelos de softwares em conjunto.
- **Programação em Pares:** A programação é realizada em conjunto (2 programadores) desenvolvendo o mesmo código em uma única máquina, enquanto um escreve o outro analisa.
- **Integração Contínua:** Cada vez que uma tarefa é finalizada a integração pode ser realizada várias vezes ao dia.
- **Cliente Presente:** Com a participação do cliente junto ao desenvolvimento do software, as dúvidas que possam ocorrer durante o desenvolvimento podem ser sanadas pelo cliente.
- **Padrões de Codificação:** Padroniza toda a escrita do código fonte para que o mesmo possa ser compartilhado com todos os desenvolvedores do projeto, fazendo com que a equipe trabalhe eficazmente.
- **Horas de Trabalho:** Cada envolvido no projeto tende cumprir uma carga horária de 40 horas semanais, desta forma todos permanecem descansados.

CAPÍTULO 5 – PROPOSTA DE UM FRAMEWORK DE GERÊNCIA DE PROJETOS EM AMBIENTES ÁGEIS

5.1 Considerações Iniciais

Neste capitulo serão abordados o framework proposto e seu diferencial atendendo a qualquer demanda de uma forma ampla e inteligente, sendo dinâmico e não amarrado a nenhuma tecnologia.

Na seção 5.2 apresenta a funcionalidade do framework, seu foco, sua comunicação, seus módulos e um esboço de como funcionará o framework como proposto. Na subseção 5.2.1 serão apresentados todas as funcionalidades, regras e comunicação do framework proposto em modo geral e bem simplificado.

5.2 Framework Proposto

A proposta do Framework de gerência de projetos em ambientes ágeis em pauta neste trabalho. Tem como objetivo facilitar e agilizar o controle de desenvolvimento de software nas empresas, proposta esta, de desenvolver algo diferenciado atendendo a qualquer demanda de uma forma ampla e inteligente, pois atualmente os usuários precisam se adequar aos sistemas já existentes e este modelo propõe-se que o sistema se adeque aos usuários conforme regras criadas pelos responsáveis pelo projeto.

O foco deste Framework não é de tecnologia e sim focar em regras de negócios, seria genérico permitindo que qualquer outro sistema de gerenciamento de projetos rode sobre ele não ficando amarrado ao usuário. Sua função é apenas permitir que uma determinada ação aconteça e esta ação seja executada e realizada dentro de um banco de dados.

O usuário poderá usar o sistema de maneira ampla não ficando preso na plataforma que o sistema rodar, seja ela WEB, Desktop, Mobile etc... pois o processo depende muito de como o responsável pelo projeto (Desenvolvedor) irá desenvolver o sistema que trabalhará em conjunto com o Framework. Ou seja, o Framework não fica preso a uma tecnologia, mesmo tendo dependências tecnológicas.

Toda a comunicação entre o sistema desenvolvido pelo responsável pelo projeto e o Framework proposto, deverá ser realizada através de API, onde entrando com determinados dados a API realiza a consulta e retorna os resultados solicitados. O framework também possuirá uma API com ligação aos módulos prontos de negócios, esses módulos serão os principais módulos do framework, pois neles estarão os seguintes itens:

- Projetos
- Atividades
- Usuários, equipes e papeis
- Workflow
- Relatórios
- Custos, tempo e orçamento

Além dos módulos de negócios, existirão os módulos internos como:

- Customização de banco de dados
- Registros e Históricos de ações (Logs)
- Comunicação e alertas (E-mails, SMS, etc...)
- Internacionalização

Todos esses módulos estarão interligados uns aos outros, e a customização existindo alguma regra própria elaborada pelo responsável do projeto (desenvolvedor do sistema) as regras padrões do framework serão alteradas. Estes módulos internos, poderão ser ligados a outros frameworks, claro que não deixará total acesso aos mesmos, mas como exemplo: se utilizar a linguagem PHP e baixo nível o CAKE PHP onde já possuem vários métodos prontos para serem utilizados, preparando os dados modificando as consultas e conversando com o banco de dados, cria-se um meio de fazer softwares com gerenciamento de projetos não ficando amarrados a nenhuma tecnologia, possibilitando que qualquer metodologia funcione junto ao mesmo. Permitindo assim, que não fique preso ao que foi idealizado pelos desenvolvedores. Os desenvolvedores precisam apenas conhecer as regras do framework proposto, desta forma conseguirá juntar este framework de gerenciamento de projetos com qualquer outro sistema existente.

Por exemplo: Caso ele já possua um sistema dentro de sua empresa, seja ele de atendimento, contas apagar etc... como o framework não é fixo, pode fazer com que ao invés de ter uma aplicação separada, utilize uma já existente apenas implementando os métodos para comunicação com o framework.

Na figura 5.1 é demonstrada a forma simplificada de como deve funcionar o Framework proposto junto a um sistema e demais frameworks interligados.

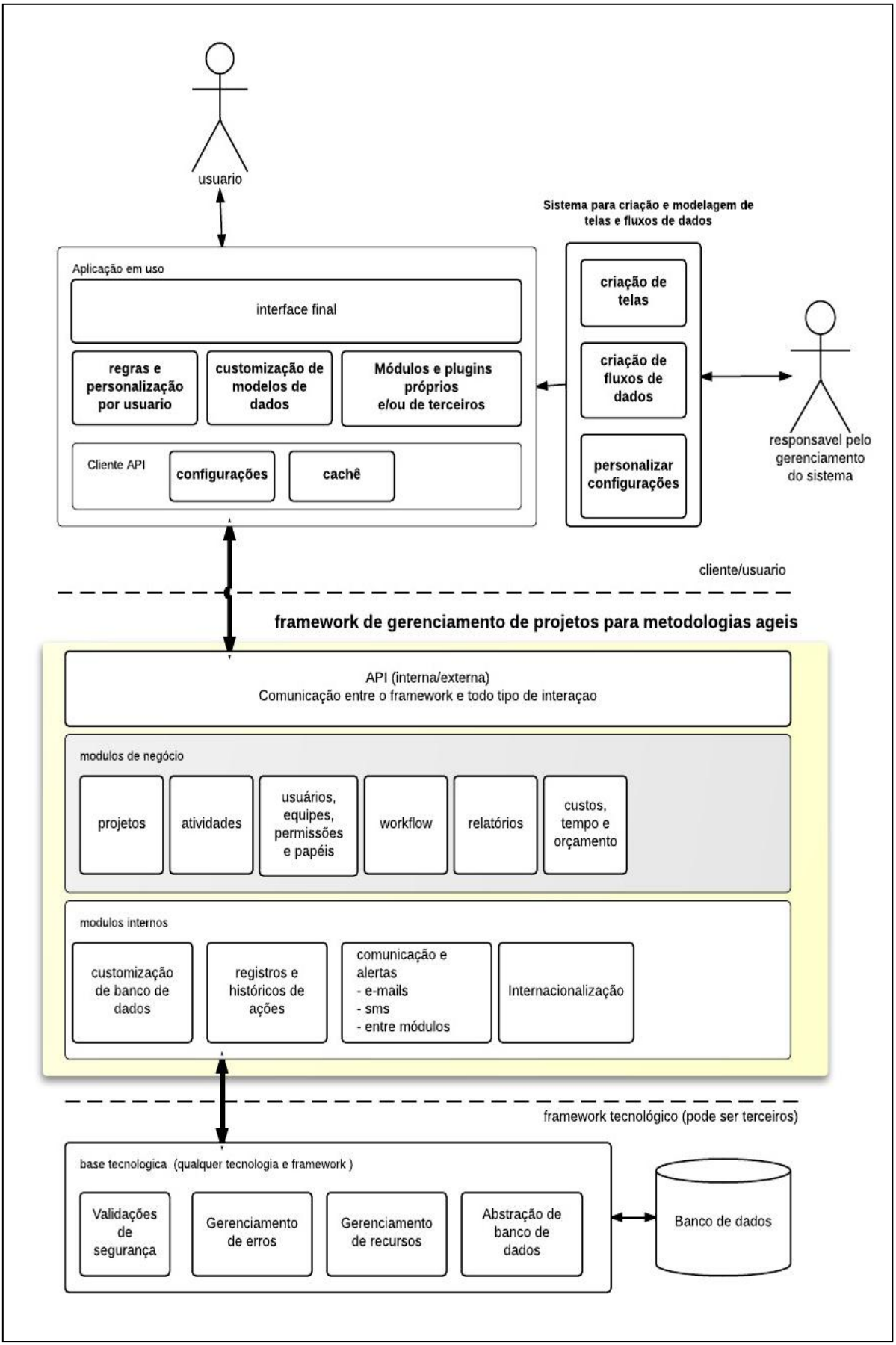

Figura 16 - Proposta de Framework de gerencia de projetos em ambientes ágeis (Fonte própria)

5.2.1 Entendendo melhor o Framework Proposto

Conforme demonstrado na (figura 5.1), página 44, o responsável pelo gerenciamento, pressupõe que ele já tenha conhecimento em tecnologia e tenha habilidade, diferente do usuário que apenas se preocupa com a atividade que está empenhado a fazer. Este responsável deve conhecer todo o ambiente do framework através de documentações e elementos para preparar o ambiente o mais correto possível, criando telas (Interfaces) aos usuários, ou utilizando IDE para criação de telas conversando com a API onde a mesma fará o restante da comunicação com o framework.

Como informado anteriormente, a parte tecnológica não necessita reinventar a roda, ou seja, não há necessidade para criar um framework de negócio fazer um novo sistema para lidar com o banco de dados etc... pois existem outros frameworks com esta finalidade e com isso tiraria o foco do framework proposto que facilita deixando o projeto ágil, ficando mais maleável e controlando as tarefas programadas.

A abstração do banco de dados, será apenas um facilitador para o framework na comunicação com o banco de dados, seja ele tradicional ou não, o que importa é que seja bem compreendido pelo framework.

A API manda uma ação e retorna um resultado onde as chamadas podem ser internas ou externas. Os módulos de negocio são onde encontra-se o diferencial do framework, são sistemas genéricos onde tem o mínimo necessários para funcionar, porém, todos respeitam o que o responsável customizou. Este responsável conhece a relação entre cada um deles, pode não saber como funciona mas sabe que pode mesclar (N) informações e cada um tem um papel dentro do modulo de negocio.

- **Projetos**: seu papel é servir de numerador de todas as outras ações.
- **Atividades**: são literalmente as ações que estão ocorrendo podendo estar ligadas ao projeto e também as outras partes.
- **Usuários, permissões e papeis:** é um modulo que gerencia os demais. Ou seja, quando é realizado um login, por exemplo, este modulo é chamado e irá reagir conforme as ações que foram programadas.
- **Workflow**: conforme as regras que são recebidas repercute para outras ações, ou seja, conforme as regras do workflow ele replica as demais, por exemplo, cria-se uma regra de atividades de tempo limite de 2 dias, com isso, quando estiver expirando este tempo

avisa os demais usuários do projeto onde o responsável pode ampliar o prazo ou fechar o projeto caso desejado. Permite também criar um novo projeto trazendo as tarefas que não foram concluídas no projeto anterior.

- **Relatórios**: são um compilador de informações conforme o que foi programado no sistema pelo responsável, trará um relatório já pronto e customizado.

- **Custo, tempo e orçamento:** estão ligadas aos demais módulos, onde faz todo o processo de calculo, analise e registra essas informações. Ou seja, demonstrará quanto custou cada atividade desenvolvida ou em desenvolvimento, quanto tempo o usuário trabalhou ou está trabalhando no projeto, quanto de acerto do projeto (O quanto foi pretendido e o quanto foi feito) este calculo é chamado de acurácia.

Como o framework trabalhará sobre outra tecnologia, a parte interna demonstrada na (figura 5.1) página 44, fará com que o framework precise de tratamentos para cada comunicação que será realizada. Ou seja, transformará as ordens que estão sendo realizadas pelos usuários mais as ordens do modelo de negócio em uma consulta personalizada. Por padrão será registrado (O que o usuário fez, que horas acessou, do que ele participou, etc...).

O modulo de comunicação possibilitará que todas as outras partes se comuniquem e possam avisar os usuários que estão no sistema. Exemplo: ao logar no sistema, se alguém enviou um aviso anteriormente a pessoa que acaba de logar, ele obtém essas informações ou avisos por e-mail, SMS etc...

Por se tratar de um framework de projetos ligados a negócios, e negócios não é limitados ao País de origem, ele não é amarrado a um idioma especifico, com isso, poderá obter personalizações de linguagens conforme as regras personalizadas que o usuário pretender processando e devolvendo conforme solicitado.

A ideia geral de um framework de projetos e o foco em metodologias ágeis é que elas são dinâmicas não sendo amarrada, igual ao modelo anterior PMBOK, possuindo (N) possibilidades, sendo uma proposta de desenvolver algo diferenciado atendendo a qualquer demanda de uma forma inteligente.

CAPÍTULO 6 – CONCLUSÃO

6.1 Considerações Iniciais

Neste capitulo serão abordados o diferencial do framework proposto, seus pontos fortes, fracos e uma pequena comparação com algumas ferramentas de mesma finalidade atualmente no mercado.

Na seção 6.2 serão apresentados o diferencial do framework em relação a outros modelos já existentes e alguns pontos importantes do mesmo. Na subseção 6.3 serão apresentado as vantagens e desvantagens do framework proposto, fácil adaptação, customização e dependências. Na seção 6.4 serão abordados os resultados obtidos pelo framework e alguns critérios de avaliação realizado por pessoas da área que atuam no seu dia a dia com outras ferramentas existentes. Na seção 6.5 serão abordados sugestões para trabalhos futuros, como pesquisas e desenvolvimentos, para melhoria e implementação do framework proposto nesta tese.

6.2 Diferencial do Framework Proposto

Devido à demanda de produtos de software, pontos chave como cronograma, qualidade, menos custo de desenvolvimento (retrabalho), satisfação do cliente, prazo de entrega entre outros. Poucas empresas abordam esta metodologia para compreender e disciplinar suas equipes para um desenvolvimento de software ágil e de qualidade.

Muitas ferramentas existentes atualmente fazem com que os usuários precisem se adequar a elas. Com isso, um framework que possibilita se adequar aos usuários por meio de regras criadas pelos responsáveis do projeto (desenvolvedores) e não os usuários se adequarem ao sistema como as demais ferramentas, torna-se um diferencial atendendo a qualquer demanda de uma forma ampla e inteligente, sendo dinâmica e não amarrada a nenhuma tecnologia.

Realizando uma breve comparação do Framework proposto com outros modelos já existentes no mercado atualmente, como Redmine e Project, dois dos modelos mais famosos de gerenciamento de processos de software.

Tanto o Project quanto o Redmine trabalham com API e com possibilidades de ampliar a maior parte de seus recursos. Mas o grande diferencial do Framework Proposto em relação aos demais, além de utilizar API, é que ele seria um framework de negócios.

Ou seja, basearia na realização de tarefas conforme necessidade dos usuários, não estaria preso e não possuiria um formato pronto para utilização. Com isso, se adapta melhor a realidade do cotidiano das empresas e a novas metodologias e variações das mesmas.

Outra questão, é que a parte das telas (Visual do sistema), não nasce amarrada como nos demais. O framework teria uma aplicação de telas demonstrativas, mas não significa que ela seria definitiva, pois poderá criar N aplicações com utilização do Framework gerando assim o valor esperado pelo cliente no momento que ele necessitasse.

Outro ponto relevante do Framework seria a integração que poderá ocorrer facilmente com qualquer projeto, ou seja, qualquer sistema já existente. Não obrigatoriamente irá seguir regras bem definidas como os demais seguem, onde para criar um projeto o usuário precisa mandar informações de um jeito ou de outro, pelas regras já existentes que não são maleáveis ou limitadas as possibilidades básicas. Com o Framework proposto, a empresa pode desenhar um modelo de projeto único. Ou seja, ela pode até utilizar um sistema de ordem de serviço, ou até mesmo um ERP que ela possui, para consumir este projeto. Tornando assim, um modelo de negócio livre e dinâmico.

A parte técnica do framework vai ser melhor elaborada em futuras situações, mas a proposta inicial seria esta, de tornar um sistema de negócios fazendo esta parte genérica, que é a comunicação com o banco de dados, a comunicação entre as pessoas, filtrar, ordenar e organizar tudo conforme desejado pelo usuário. Dando total liberdade e personalização ao responsável pelo sistema que com seu conhecimento poderá criar e adaptar o framework a suas necessidades.

O framework não se limita em metodologias como Scrum, XP ou qualquer outro, pois foca na realidade da empresa, estando bem próximo do que a empresa precisa que na realidade não é algo complexo cheio de amarrações e que acaba não dando o resultado esperado como alguns dos modelos já existentes.

6.3 Vantagens e Desvantagens do Framework Proposto

Como citado ao longo do trabalho, a vantagem do modelo proposto é ele se adequar a empresa, foram vistos diversos modelos que buscam solucionar a execução de projetos, mas na realidade cada grupo dentro da empresa, tem necessidades que vão muito além do que soluções oferecem, não é pela quantidade de recursos que teremos a certeza da aplicação correta do projeto ou da boa comunicação necessária para o sucesso, mas pela utilidade e

valor final que é gerado, isso é a vantagem e o objetivo na proposta de um framework de negócios.

A desvantagem deste modelo estará totalmente limitada ao nível de conhecimento e experiência da pessoa ou grupo de pessoas que serão responsáveis pela tradução das necessidades da empresa ao fluxo de negócios que o framework executará.

Este ponto apesar de ser algo muito crítico para o sucesso ou insucesso não pode ser um desestimulo para quem for concretizá-lo, assim é sugerido que além da montagem na do sistema, também seja feita um bom material de apoio, com exemplos e aplicações básicas.

6.4 Resultados obtidos (Critérios de Avaliação)

Após a formatação da proposta desta solução, a mesma foi submetida a avaliação de duas pessoas da área de TI, sendo as mesmas com experiência de mais de 5 anos na execução de projetos e testes, de um empresa de Marília - SP.

Inicialmente foi apresentado os diferenciais do framework proposto, sendo estes já citados durante o decorrer desta dissertação, também foi buscado a opinião sobre a realidade que cada um enfrenta no trabalho cotidiano, utilizando Redmine como ferramenta para o desenvolvimento interno e o Microsoft Project para projetos externos.

O *feedback* foi muito favorável, tendo respostas que confirmaram a necessidade de uma solução maleável e amigável a realidade, pois na empresa que trabalham se usa o Project para o controle macro dos projetos junto ao cliente e o Redmine na execução das tarefas necessárias, mas não existe integração automática entre os mesmos, gerando assim a necessidade de controles paralelos como planilhas, arquivos textos dentre outros controles.

Apesar de cada ferramenta citada resolver bem o que se propõem, fica claro que não são satisfeitos os pontos chave que a empresa busca, como minimização de custos, e um planejamento mais simples de se executar, isso na pratica tem bom resultado devido o mérito dos profissionais e não pelo fluxo em si.

6.5 Sugestões de Trabalhos Futuros

Dando continuidade na proposta do trabalho desenvolvido nesta dissertação, podem-se realizar diversas pesquisas e/ou desenvolvimento para trabalhos futuros, como:

- Colocar em prática essa proposta, assim a mesmo se tornando um produto comercial, ou de código aberto, será muito útil para solucionar na prática as necessidades das empresas perante os problemas comuns dos projetos.

- Adaptar este modelo proposto ao outras ferramentas com outros propósitos, assim contribuir para área de TI, principalmente para que desenvolvimento e tarefas cotidianos não se limitem a soluções ou modelos comuns, mas principalmente a forma como as pessoas melhor produzam, já que cada empresa ou grupo possui desafios e limitações impossíveis de se prever completamente.

Como últimas recomendações para um trabalho futuro, a respeito da execução desta proposta:

Primeiro, buscar uma linguagem comum (ubíqua), algo compreensível para as pessoas do projeto, sendo as mesmas técnicas ou não, responsáveis pelo sistema e o framework, no período quando esta dissertação foi realizada, as tecnologias mais viáveis para a execução desta tarefa é o BPEL (*Business Process Executation Language*), um padrão de linguagem para descrever fluxos de negócios, ou modelos mais simplificados como YAML (*YAML Ain't Markup Language*) ou DSL (*Domain Specific Language* – linguagem específica de domínio), estas sendo padrão simples de expressão as ações que o sistema deve executar feita através de uma linguagem próxima da natural, assim podendo ser revisado e melhorado continuamente.

Ainda na esfera de padrões, um grande desafio é a modelagem dos dados dinamicamente, este ponto é complexo, pois além de inúmeras implementações e alterações dos usuários, deve interferir diretamente no desempenho do sistema, o padrão atual de banco de dados relacionais não é a melhor escolha, mas caso pretenda utilizar o mesmo fica a sugerido se aprofundar os estudos no modelo EAV (Entidade Atributo Valor) amplamente divulgado pela empresa Magento, este permite criar bancos complemente dinâmicos do modelo físico atual, mas saindo do modelo tradicional de bancos, essa solução emergente mais aconselhada é o modelo NoSql, este permite criação, manutenção e manipulação dinâmica de dados com alto desempenho.

No interior do Framework de negócios, além da criação de um modelo de interpretação das regras de negócio, outro ponto chave é o relacionamento interno dos módulos, para isso ocorrer sem que haja amarração dos mesmos é sugerido o uso de um modelo amplamente utilizado do framework web Django, chamado sinais (*signals*), na prática todas as ações que devem ser utilizados por outros módulos não são instanciados, e sim

registram através desses sinais, que poderão ser usados. Assim, dinamicamente um modulo de baixo nível responsável por interpretar os demais módulos facilmente integra as ações. Ex.: sistema projetos precisa enviar um e-mail a cada nova interação, e o mesmo estão descritos nas regras, mas o envio de e-mails é ação exclusiva do módulo de comunicação, neste caso se o mesmo for um sinal registrado, quando solicitado envio de e-mail é feito pelo outro modulo de forma natural no fluxo.

REFERÊNCIAS

ABRAHAMSSON, P.; SALO, O.; RONKAINEN, J.; WARSTA, J. **Agile software development methods, Review and analysis. ESPOO (Technical Research Centre of Finland) 2002. VTT Publications n 248.** Disponível em: http://www.vtt.fi/inf/pdf/publications/2002/P478.pdf

ARTHUR, Jay Lowel. **Improving software quality: an inside`s guide to TQM**. New York: Wiley & Sons, 1993.

BECK, K. **Extreme programming explained:** embrace change. 2.ed Addison-Wesley, 2000. 224p.
BECK, K.; BEEDLE, M.; VAN BENNEKUM, A.; COCKBURN, A.; CUNNINGHAM, W.; FOWLER, M.; GRENNING, J.; HIGHSMITH, J.; HUNT, A.; JEFFRIES, R.; KERN, J.; MARICK, B.; MARTIN, R. C.; MELLOR, S.; SCHWABER, K.; SUTHERLAND, J.; THOMAS, D.; **Manifesto for agile software development.** Disponível em: http://www.agilemanifesto.org

BAETJER, Ir., H., **Software as Capital**, IEEE Computer Society Press, 1998.

BOEHM, B.; HUANG, L. Value-Based Software Engineering: Reinventing "Earned Value"Monitoring and Control. **Acm Sigsoft Software Engineering Notes**, Nova Iorque, 2003

CHALOLA, Luiz. **Gerenciamento de Serviços de TI na Prática: Uma abordagem com base em COBIT e ITIL.** 2007.

COBIT 4.1. Cobit 4.1 – **Modelo, Objetivos de Controle, Diretrizes de Gerenciamento e Modelos de Maturidade.** Disponível em: http://www.isaca.org/ContentManagement/ContentDisplay.cfm?ContentID=55274

FAGUNDES, Eduardo Mayer. COBIT: **Um kit de ferramentas para a excelência na gestão de TI**. 2004 Disponível em: http://www.efagundes.com/artigos/Arquivos_pdf/Cobit.PDF

FERNANDES, A. A., **Gerência de software através de métricas : garantindo a qualidade do projeto, processo e produto**. Atlas S.A. São Paulo - 1995.

FERREIRA, D.; COSTA, F.; ALONSO, F.;
ALVES, P.; NUNES, T. SCRUM - **Um Modelo Ágil para Gestão de Projetos de Software.** Disponível em: http://paginas.fe.up.pt/~aaguiar/es/artigos%20finais/es_final_19.pdf.

MALLMANN, P (2010). **Um modelo abstrato de gerência de software para metodologias ágeis**. UNISINOS, PIPCA.

PMI. **Um Guia do Conhecimento em Gerenciamento de Projetos.** Guia PMBOK®. Quarta Edição – EUA: Project Management Institute, 2008.

PRESSMAN, R. S. **Software engineering: A practitioner's approach**. 6th. ed. McGraw Hill, 2006.

PRESSMAN, R. S. **Software Engineering: A Practitioner's approach.** 7th ed., McGraw Hill, 2010.

ROCHA, A. R. C.; MALDONADO, J. C.; WEBER, K. C. **Qualidade de software: Teoria e Prática**, São Paulo: Prentice Hall, 2001.

ROYCE, W. Managing the Development of Large Software Systems: Concepts and Techniques. **In International Conference on Software Engineering**, Califórnia, 1970.

SCHWABER, Ken e BEEDLE, Mike – **Agile Software Development with Scrum**, Prentice Hall, 1996.

SCHWABER, Ken e BEEDLE, Mike – **Agile Software Development with Scrum**, Prentice Hall, 2002.

SCHWARTZ, J. I. ,**Construction of software. In: Practical Strategies for Developing Large Systems. Menlo Park:** Addison-Wesley, 1st. ed., 1975.

SOMMERVILLE, I. **Software engineering.** 5th. ed. Addison-Wesley, 1995.

SOTILLE, Mauro ett alli. **Gerenciamento de Escopo em Projetos**. Rio de Janeiro: Fundação Getúlio Vargas,2006. (ISBN 8522505799). Fluxo resumido de processos do gerenciamento de projetos. 2009. Disponível em http://www.pmtech.com.br/artigos/Fluxo_PMBOK_4aEd_Mauro_Sotille_A4.pdf

SUTHERLAND, J.; SCHWABER, K. **The Scrum papers: nuts, bolts, and origin of an agile process**, 2007. Disponível em: http://scrumtraininginstitute.com

VARGAS, Ricardo Viana. **Manual Prático do Plano de Projeto Utilizando o PMBOK®** Guide 2000 Edition. Rio de Janeiro: Brasport, 2003.