

**PERAF - UM PROCESSO DE ENGENHARIA DE
REQUISITOS ADAPTÁVEL E FLEXIVEL**

ANDRÉ FONSECA AMÂNCIO
ORIENTADOR: RODOLFO SÉRGIO FERREIRA DE RESENDE
CO-ORIENTADOR: HEITOR AUGUSTUS XAVIER COSTA

PERAF - UM PROCESSO DE ENGENHARIA DE
REQUISITOS ADAPTÁVEL E FLEXIVEL

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

BELO HORIZONTE / MG
JULHO DE 2011

© 2011, ANDRÉ FONSECA AMÂNCIO
Todos os direitos reservados.

Amâncio, André Fonseca.

A484p PERAF - um processo de engenharia de requisitos adaptável e flexível / André Fonseca Amâncio. — Belo Horizonte, 2011.
xviii, 45f. : il. ; 29cm

Dissertação (mestrado) — Universidade Federal de Minas Gerais. Departamento de Ciência da Computação.

Orientador: Rodolfo Sérgio Ferreira Resende.
Coorientador: Heitor Augustus Xavier Costa.

1. Computação - Teses. 2. Engenharia de Software - Teses. 3. Engenharia de Requisitos – Teses I. Orientador. II. Coorientador III. Título.

CDU 519.6*32 (043)



UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

PERAF - Um processo de engenharia de requisitos adaptável e flexível

ANDRE FONSECA AMANCIO

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. RODOLFO SÉRGIO FERREIRA DE RESENDE - Orientador
Departamento de Ciência da Computação - UFMG

PROF. HEITOR AUGUSTUS XAVIER COSTA - Co-orientador
Departamento de Ciência da Computação – UFLA

PROF. CLARINDO ISAÍAS P. DA SILVA E PÁDUA
Departamento de Ciência da Computação - UFMG

PROF. ANTONIO MARIA PEREIRA DE RESENDE
Departamento de Ciência da Computação – UFLA

Belo Horizonte, 29 de julho de 2011.

Dedico este trabalho à minha mãe, ao meu pai, ao Tom, à Luci, ao meu orientador Rodolfo, ao meu Co-orientador Heitor, ao programa MINTER, à UFMG e à UFLA.

Agradeço a realização desse trabalho:
aos familiares, pela paciência e apoio;
aos amigos pelo apoio e pelas ajudas nas horas de necessidade;
em especial, ao Tom e à Luci, amigos que estiveram sempre junto nessa etapa.
Agradeço também, à paciência e compreensão dos professores Rodolfo e Heitor,
a oportunidade criada pelo programa MINTER,
à professores e todos envolvidos no mestrado pelo seu excelente trabalho que me
encantaram e deram força para sempre seguir em frente.
Desejo a todos, meus sinceros agradecimentos.
Obrigado!

*“A apatia só pode ser superada com entusiasmo,
e o entusiasmo pode ser despertado por duas coisa:
um ideal que acende a imaginação
e um plano claro e definido para levar esse ideal à prática.”*
[Toybee, J.A. (2004)]

Resumo

Quanto mais as organizações procuram ganhar vantagem competitiva com a rápida implementação de serviços e produtos que atendam ou preferencialmente excedam as necessidades e expectativas dos clientes, mais os desenvolvedores se encontram sob crescente pressão para criar novas funcionalidades ou melhorar as já existentes. Assim, muitas empresas que usavam o desenvolvimento tradicional viram nos métodos de desenvolvimento ágil uma alternativa para superar essa crescente pressão. Esses métodos de desenvolvimento ágil apregoam a habilidade de acomodar mudanças, de aproximar o relacionamento com o cliente e entregar software funcional utilizando pequenos períodos de desenvolvimento. Muitas vezes a forma como esses métodos lidam com os requisitos do projeto é diferenciada dos métodos de desenvolvimentos tradicionais. Assim, é necessário utilizar métodos que permitam gerenciar as atividades relacionadas aos requisitos. O objetivo deste trabalho é fazer um levantamento dos requisitos, características, metas entre outros fatores mais frequentemente mencionados na literatura referentes a processos de Engenharia de Requisitos (ER) de software em metodologias ágeis e tradicionais, bem como elaborar um processo de ER direcionado a minimizar o impacto na cultura organizacional de empresas que estão em processo de migração de uma metodologia tradicional para uma metodologia ágil. Foi realizado um estudo sobre os processos de ER em métodos ágeis e dos métodos tradicionais de desenvolvimento de software. Neste trabalho, foi esboçado um Processo de Engenharia de Requisitos Adaptável e Flexível (PERAF) com o intuito de minimizar o impacto na cultura organizacional de empresas que estão em processo de migração de um método tradicional para um método ágil. O uso do processo foi avaliado em dois estudos de casos; nestes, constatou-se que o PERAF possibilitou a realização da migração de um método tradicional para uma método ágil, oferecendo também suporte às necessidades de ER de uma organização que ainda não tinha um processo de ER definido e formalizado.

Palavras-chave: Engenharia de Requisitos, Métodos de Desenvolvimento de Software, Engenharia de Software.

Abstract

As more organizations seek to gain competitive advantage with the rapid deployment of services and products that meet or preferably exceed the needs and expectations of customers, more developers are under increasing pressure to create new features or improve existing ones. Therefore many companies that use traditional development see in agile development methods an alternative to overcome this growing pressure. These agile development methods proclaim the ability to accommodate changes, to get closer relationships with customers and produce functional software in short periods of development. The way that agile methods deal with the requirements of the project is distinguished from traditional development methods. Thus, it is necessary to use methods to manage the activities related to requirements. The objective of this work is to survey the requirements, characteristics, goals and other factors most frequently mentioned in the literature concerning the processes of Requirements Engineering (RE) in the context of agile and traditional methodologies, as well as developing a RE process aimed at minimizing the impact on organizational culture of companies that are in the process of migration from a traditional methodology to an agile methodology. A study was conducted focused on the processes of RE of agile methods and traditional methods of software development. In this work, we outline a RE process adaptable and flexible (PERAF) in order to minimize the impact on organizational culture of companies that are in the process of migration from a traditional method to an agile method. The use of the process was evaluated in two case studies; PERAF made possible the development of a migration from traditional to an agile method, and it also offered support to ER needs of an organization that originally did not have an ER process defined and formalized.

Keywords: Requirements Engineering, Software Development Methods, Software Engineering.

Sumário

CAPÍTULO 1 INTRODUÇÃO 1

1.1. CONSIDERAÇÕES INICIAIS.....	1
1.2. MOTIVAÇÃO	2
1.3. OBJETIVOS.....	2
1.4. ORGANIZAÇÃO DA DISSERTAÇÃO.....	3

CAPÍTULO 2 REFERENCIAL TEÓRICO..... 5

2.1. CONSIDERAÇÕES INICIAIS	5
2.2. PROCESSO DE ENGENHARIA DE SOFTWARE	5
2.2.1. MODELO E METAMODELO	7
2.2.2. SPEM.....	7
2.3. ENGENHARIA DE REQUISITOS.....	10
2.3.1. PROCESSOS DE ER	12
2.4. METODOLOGIAS DE DESENVOLVIMENTO DE SOFTWARE (MDS)	22
2.4.1. ERMT	23
2.4.2. ERMA	24
2.4.3. COMPARAÇÃO ENTRE AS ER DOS MT'S E DOS MA'S	29
2.5. TRABALHOS RELACIONADOS.....	33
2.6. CONSIDERAÇÕES FINAIS.....	37

CAPÍTULO 3 PROCESSO DE ENGENHARIA DE REQUISITOS ADAPTÁVEL E FLEXÍVEL (PERAF) 39

3.1. CONSIDERAÇÕES INICIAIS	39
3.2. ELABORAÇÃO DO PERAF	40
3.3. DESCRIÇÃO DO PERAF.....	43
3.3.1. DISCIPLINA DESENVOLVIMENTO DE REQUISITOS	44
3.3.2. DISCIPLINA GERENCIAMENTO DE REQUISITOS	52
3.4. CONSIDERAÇÕES FINAIS	61

CAPÍTULO 4 AVALIAÇÃO DO PERAF..... 63

4.1. CONSIDERAÇÕES INICIAIS.....	63
4.2. ESTUDO DE CASO 1	64
4.2.1. CAPACITAÇÃO A DISTÂNCIA NO SISTEMA DE ZONEAMENTO ECONÔMICO E ECOLÓGICO DO ESTADO DE MINAS GERAIS	64
4.2.2. CARACTERIZAÇÃO DO CENÁRIO	64
4.2.3. ANÁLISE DO QUESTIONÁRIO DE DISCREPÂNCIAS	66
4.2.4. ANÁLISE DOS QUESTIONÁRIOS DE PRÉ E PÓS EXPERIMENTO	68
4.3. ESTUDO DE CASO 2	70
4.3.1. DES - SISTEMA DE RESÍDUOS QUÍMICOS	70
4.3.2. CARACTERIZAÇÃO DO CENÁRIO	71
4.3.3. ANÁLISE DO QUESTIONÁRIO DE DISCREPÂNCIAS	73
4.3.4. ANÁLISE DOS QUESTIONÁRIOS DE PRÉ E PÓS EXPERIMENTO	74

4.4. CICLO DE MELHORIA	76
4.5. CONSIDERAÇÕES FINAIS.....	82
<u>CAPÍTULO 5 CONCLUSÃO</u>	<u>85</u>
5.1. CONSIDERAÇÕES INICIAIS	85
5.2. RESULTADOS	86
5.3. CONTRIBUIÇÕES.....	86
5.4. PROBLEMAS E LIMITAÇÕES DO TRABALHO.....	87
5.5. TRABALHOS FUTUROS	87
<u>REFERÊNCIAS BIBLIOGRÁFICAS</u>	<u>89</u>

Lista de Ilustrações

Figura 1 - Modelo Conceitual SPEM [Adaptado de OMG, 2005].	8
Figura 2 – Elementos da UML utilizados na modelagem do PERAF.	10
Figura 3 - Processo de ER [Sommerville, 2007].	16
Figura 4 - Processo de obtenção e análise de requisitos [Sommerville, 2007].	17
Figura 5 - Gerenciamento de mudanças de requisitos [Sommerville, 2007].	20
Figura 6 - Comparação entre os processos de ER propostos por Wiegers, Pressman e Sommerville.	21
Figura 7 - Mapa Conceitual da Elaboração do PERAF.	40
Figura 8 - Diagrama de Fluxo de Trabalho do PERAF.	44
Figura 9 - Diagrama de Fluxo de Trabalho da Disciplina <i>Desenvolvimento de Requisitos</i> .	45
Figura 10 - Diagrama de Fluxo de Trabalho da Atividade <i>Elicitação</i> .	46
Figura 11 - <i>Diagrama</i> de detalhamento de atividade da Atividade <i>Elicitação</i> .	48
Figura 12 - Diagrama de Fluxo de Trabalho da Atividade <i>Análise</i> .	48
Figura 13 - <i>Diagrama</i> de detalhamento de atividade da Atividade <i>Análise</i> .	49
Figura 14 - Diagrama de Fluxo de Trabalho da Atividade <i>Especificação</i> .	50
Figura 15 - <i>Diagrama</i> de detalhamento de atividade da Atividade <i>Especificação</i> .	51
Figura 16 - Diagrama de Fluxo de Trabalho da Atividade <i>Validação</i> .	51
Figura 17 - <i>Diagrama</i> de detalhamento de atividade da Atividade <i>Validação</i> .	52
Figura 18 - Diagrama de Fluxo de Trabalho da Disciplina <i>Gerenciamento de Requisitos</i> .	53
Figura 19 - Diagrama de Fluxo de Trabalho da Atividade <i>Controle de Alteração</i> .	54
Figura 20 - <i>Diagrama</i> de detalhamento de atividade da Atividade <i>Controle de Alteração</i> .	55
Figura 21 - Diagrama de Fluxo de Trabalho da Atividade <i>Rastreabilidade e Manutenção do Estado dos Requisitos</i> .	56
Figura 22 - <i>Diagrama</i> de detalhamento de atividade da Atividade <i>Rastreabilidade e Manutenção do Estado dos Requisitos</i> .	57
Figura 23 - Diagrama de Fluxo de Trabalho da Atividade <i>Controle de Versão</i> .	58
Figura 24 - <i>Diagrama</i> de detalhamento de atividade da Atividade <i>Controle de Versão</i> .	59
Figura 25 - Diagrama de Fluxo de Trabalho da Atividade <i>Rastreabilidade dos Requisitos</i> .	60
Figura 26 - <i>Diagrama</i> de detalhamento de atividade da Atividade <i>Rastreabilidade dos Requisitos</i> .	61
Figura 27 - Adoção do PERAF no Projeto EAD/ZEE-MG (Linha do Tempo).	65
Figura 28 - Descrição do Processo de Desenvolvimento do Projeto EAD/ZEE-MG.	67
Figura 29 - Linha do tempo para a adoção do PERAF no projeto DES.	72
Figura 30 - Descrição do Processo de Desenvolvimento do Projeto DES.	73
Figura 31 - Parte do Diagrama de Fluxo de Trabalho da Atividade <i>Elicitação</i> que contém a Tarefa <i>Planejar a Elicitação</i> .	77
Figura 32 - Parte do <i>Diagrama</i> de detalhamento de atividade da Atividade <i>Elicitação</i> que contém a Tarefa <i>Planejar a Elicitação</i> .	78
Figura 33 - Partes dos Diagramas de Fluxo de Trabalho das Atividades <i>Rastreabilidade e Manutenção do Estado dos Requisitos</i> , <i>Controle de Versão</i> e <i>Rastreabilidade dos Requisitos</i> .	78
Figura 34 - Parte do <i>Diagrama</i> de detalhamento de atividade da Atividade <i>Rastreabilidade e Manutenção do Estado dos Requisitos</i> .	78
Figura 35 - Parte do <i>Diagrama</i> de detalhamento de atividade da Atividade <i>Controle de Versão</i> .	79
Figura 36 - Parte do <i>Diagrama</i> de detalhamento de atividade da Atividade <i>Rastreabilidade dos Requisitos</i> .	79
Figura 37 – Diagrama de Causa e Efeito – Disciplina <i>Desenvolvimento de Requisitos</i> .	80
Figura 38 – Diagrama de Causa e Efeito – Disciplina <i>Gerenciamento de Requisitos</i> .	80

Lista de Tabelas

Tabela 1 - Elementos utilizados para a modelagem do processo.	9
Tabela 2 – Relações entre elementos utilizados na modelagem do PERAF.	10
Tabela 3 - Abordagem comparativa da ERMT e da ERMA [Ramesh <i>et al.</i> , 2010].	31
Tabela 4 – Detalhamento do comparativo entre ERMT e ERMA.	32
Tabela 5 – Detalhamento do comparativo entre ERMT e ERMA (Continuação).	33
Tabela 6 – Trabalhos Relacionados.	34
Tabela 7 - Tarefas Modificadas.	42
Tabela 8 - Descrição das Tarefas da Atividade <i>Elicitação</i> .	47
Tabela 9 - Descrição das Tarefas da Atividade <i>Análise</i> .	49
Tabela 10 - Descrição das Tarefas da Atividade <i>Especificação</i> .	50
Tabela 11 - Descrição das Tarefas da Atividade <i>Validação</i> .	52
Tabela 12 - Descrição das Tarefas da Atividade <i>Controle de Alteração</i> .	55
Tabela 13 - Descrição das Tarefas da Atividade <i>Rastreabilidade e Manutenção do Estado dos Requisitos</i> .	57
Tabela 14 - Descrição das Tarefas da Atividade <i>Controle de Versão</i> .	58
Tabela 15 - Descrição das Tarefas da Atividade <i>Rastreabilidade dos Requisitos</i> .	60
Tabela 16 – Requisitos do Projeto EAD/ZEE-MG.	65
Tabela 17 – Análise de questionários EAD/ZEE.	69
Tabela 18 – Requisitos do Projeto DES - Sistema de resíduos químicos.	71
Tabela 19 – Análise de questionários DES.	75
Tabela 20 – Definição do problema Prioritário – Desenvolvimento de Requisitos.	76
Tabela 21 – Definição do problema Prioritário – Gerenciamento de Requisitos.	77

Capítulo 1

Introdução

1.1. Considerações Iniciais

Quanto mais organizações procuram ganhar vantagem competitiva com a implementação rápida dos serviços e dos produtos que atendam e excedam as necessidades e as expectativas dos clientes, mais os desenvolvedores se encontram sob crescente pressão para desenvolver novas funções ou melhorar as existentes rapidamente [Turk *et al.*, 2005]. Assim, as organizações encontraram nos Métodos Ágeis (MA's) uma forma de atender a demanda constante de criação e de alteração das funções exigidas pelos clientes.

No entanto, os MA's diferem dos Métodos Tradicionais (MT's), em especial na Engenharia de Requisitos (ER). Os MA's adotam uma abordagem iterativa para lidar com os requisitos enquanto os MT's seguem um procedimento formal para tentar produzir uma especificação mais completa. A Engenharia de Requisitos em Métodos Tradicionais (ERMT) visa descrever os requisitos de uma forma mais completa, sendo centrada em descobrir os requisitos antes de iniciar o desenvolvimento. Por outro lado, a Engenharia de Requisitos em Métodos Ágeis (ERMA) é mais dinâmica, adaptável e está distribuída ao longo do desenvolvimento do software [Paetsch *et al.*, 2003 e Cao e Ramesh, 2008].

Combinar os benefícios dos MA's com os benefícios dos MT's em uma abordagem híbrida pode ser uma solução efetiva. Essa combinação agruparia a capacidade de acomodar mudanças, teria um foco maior nas pessoas do que nos processos e agregaria os benefícios da boa documentação voltados para manutenção futura, suporte e treinamento [Cao e Ramesh, 2008].

1.2. Motivação

Com base nesse contexto, a motivação para a realização deste trabalho considera os seguintes itens:

- Existe uma vasta gama de métodos de desenvolvimento de software que podem ser organizados em MT's e MA's.
- Existe a necessidade das empresas adotarem novas metodologias para acompanhar as necessidades de negócio (implementação rápida dos serviços e dos produtos que atendam e excedam as necessidades e as expectativas dos clientes).
- Organizações vem encontrando nos MA's de desenvolvimento uma forma de atender a demanda constante de criação e de alteração das funções exigidas pelos clientes.
- São úteis as abordagens para deixar a migração de MT's para MA's menos impactantes, considerando que estudos apontam para uma tendência de organizações migrarem para os MA's.

1.3. Objetivos

O objetivo geral deste trabalho foi realizar um levantamento dos requisitos, características, metas entre outros fatores mais frequentemente mencionados na literatura referentes a processos de ER de software em MA's e MT's, bem como definir um processo de ER direcionado a não causar grande impacto na cultura organizacional de empresas que estão em processo de migração de um método tradicional para um método ágil.

O projeto correspondente a esta dissertação atendeu aos seguintes objetivos específicos:

- Verificação do estado da arte dos temas relacionados aos MT's, MA's com foco na ER correspondente a cada uma dessas áreas;
- Elaboração de um processo de gerência de requisitos semi-ágil;
- Implantação do processo em duas organizações;

- Avaliação do processo proposto;

No entanto, conforme é discutido na conclusão, não existem fortes evidências de que os objetivos foram atingidos de forma plena. O nome inicial do processo era Processo de Engenharia de Requisitos Semi-Ágil (PERSÁ), contudo, após discussões sobre sua agilidade ou semi-agilidade, melhor detalhadas na conclusão, o nome foi alterado para Processo de Engenharia de Requisitos Adaptável e Flexível (PERAF). Além disso, foram observadas dificuldades na definição do processo, assim, o PERAF pode ser melhor visto como um conjunto de diretrizes para a instanciamento de um processo ou ainda um esboço de processo do que um processo propriamente dito.

1.4. Organização da Dissertação

Este trabalho está organizado em cinco capítulos, sendo que este apresenta o contexto no qual a proposta de mestrado está inserida, bem como os objetivos e a motivação para o seu desenvolvimento.

No Capítulo 2 é apresentado o Referencial Teórico considerando os assuntos relevantes para esse trabalho. São feitas considerações sobre os processos de engenharia de software, processos de ER, MA's e MT's de desenvolvimento de software e por fim são apresentados os trabalhos relacionados à dissertação.

No Capítulo 3 é apresentada a elaboração do PERAF, e ainda uma descrição de suas disciplinas, atividades, tarefas e principais fluxos usados para sua descrição.

No Capítulo 4, o resultado da avaliação do PERAF é apresentado por meio de dois estudos de caso (projetos reais de desenvolvimento de software) a partir de equipes com características e contextos organizacionais distintos.

Por fim, no Capítulo 5 o trabalho é concluído por meio da discussão dos resultados obtidos, das contribuições, das limitações deste trabalho e da apresentação de sugestões de pesquisa para trabalhos futuros.

Por fim são apresentadas as referências bibliográficas utilizadas no trabalho.

Capítulo 2

Referencial Teórico

2.1. Considerações iniciais

Este capítulo apresenta os principais conceitos do texto da dissertação e uma discussão sobre o estado da arte dos temas relacionados à Engenharia de Requisitos (ER) dos Métodos Tradicionais (MT's) e dos Métodos Ágeis (MA's).

Para evitar problemas de interpretação dos diversos trabalhos, foram adotadas terminologias gerais para alguns termos apresentados com diferentes nomenclaturas pelos autores citados neste trabalho, destacando os termos “método” e “metodologia de desenvolvimento de software” em detrimento de termos como “ciclo de vida” e ou “modelo de desenvolvimento de software”, respectivamente. O termo “método” é utilizado como sendo um ou mais componentes de um processo. O termo “metodologia” é utilizado para representar uma coleção de métodos similares ou o estudo de uma coleção de métodos. Assim temos o “método XP” e o método Scrum” dentre outros e a expressão “métodos ágeis” ou “metodologia ágil” representa dois ou mais métodos ágeis.

Este capítulo está organizado da seguinte forma: na Seção 2.2 são apresentados os conceitos relacionados a processos de engenharia de software; na Seção 2.3 são abordados processos de ER; na Seção 2.4 é referenciado a ER das principais metodologias de desenvolvimento de software; na Seção 2.5 são apresentados os trabalhos relacionados à dissertação e, por fim, na Seção 2.6 estão as considerações finais.

2.2. Processo de Engenharia de Software

Um processo pode ser visto como uma série de passos parcialmente ordenados envolvendo atividades, restrições e recursos que produzem uma saída desejada. Um processo normalmente envolve um conjunto de ferramentas e técnicas executadas para gerenciar,

desenvolver e manter softwares. No desenvolvimento de software é importante seguir um processo de desenvolvimento, a fim de compreender, controlar e melhorar o que acontece durante a produção deste [Pfleeger, 2001].

Compartilhando essa ideia, o *Institute of Electrical and Electronic Engineers* (IEEE) define processo como uma sequência de passos realizados para um determinado propósito e trata Engenharia de Software como a aplicação de uma abordagem sistemática, disciplinada e quantificável, para o desenvolvimento, a operação e a manutenção de software [IEEE, 1990].

Sommerville [Sommerville, 2007] une os termos *processo* e *engenharia de software* em um conceito único, definindo *processo de engenharia de software* como um conjunto de atividades que leva à produção de um software, ressaltando que, embora existam muitos processos de software diferentes, algumas atividades fundamentais são comuns a eles:

- *Especificação de Software*: a funcionalidade do software e as restrições sobre sua operação devem ser definidas;
- *Desenho¹ (design) e implementação de software*: o software que atenda a especificação deve ser produzido;
- *Validação de software*: o software deve ser validado para garantir que ele faça o que o cliente deseja, e;
- *Evolução de software*: o software deve evoluir para atender às necessidades mutáveis do cliente.

Associado a essas atividades o processo é composto pelos elementos: i) etapas, que configura um termo genérico para uma divisão temporal de um processo; ii) papéis, entendido como o conjunto correlato de proficiências, competências e responsabilidades, desempenhado por uma ou mais pessoas; iii) insumos, que se refere a aquilo que é consumido em processo, e; iv) resultados, referentes às saídas dos processos [Paula Filho, 2009].

A estrutura do processo guia as ações permitindo avaliar, entender, controlar e aprimorar as atividades que compõem o processo, garantindo que as experiências sejam capturadas e possam ser transmitidas para suas próximas versões. Além disso, os processos podem ser descritos de diversas maneiras diferentes, seja usando figuras, textos ou uma

¹ A tradução utilizada para o termo “projeto” neste trabalho é “desenho”.

combinação de ambos para permitir a seus usuários desenvolverem softwares utilizando suas metodologias, técnicas e ferramentas preferidas [Pfleeger, 2001].

2.2.1. Modelo e Metamodelo

Um modelo de processo de software é um exemplo usado para descrever o estado atual da modelagem de processos de software. Esses modelos possibilitam que os envolvidos no processo tenham a mesma visão sobre o processo, possibilitando o monitoramento e coordenação de sua execução [Curtis *et al.*, 1992].

Em decorrência do surgimento de várias linguagens de modelagem de processo de software, diversas empresas, como, por exemplo, Alcatel, IBM e Unisys, se uniram para elaborar um metamodelo que pudesse atender aos requisitos definidos pelo *Object Management Group* (OMG). Um metamodelo é um modelo que define a linguagem para expressar um modelo [Rational Software Corporation, 2002]. Em 2002, o metamodelo SPEM (*Software Process Engineering Metamodel*), baseado na UML (*Unified Modeling Language*), foi oficializado pela OMG [OMG, 2005] e é apresentado de forma resumida na próxima subseção.

Como se verá adiante, a importância de se descrever mais sucintamente o SPEM reside no fato de que PERAF foi modelado com base neste. Em complemento a isso, os diagramas do PERAF foram preparados com o uso da ferramenta MagicDraw [MAGICDRAW, 2011], aliada ao componente (*plug-in*) de suporte ao SPEM. A descrição do PERAF é apresentada no Capítulo 3.

2.2.2. SPEM

O SPEM corresponde a uma especificação de Metamodelo de análise e construção de modelos aplicáveis e úteis para definição de processos. Na época dos trabalhos desta dissertação o SPEM se encontrava na versão 2.0. Ele é utilizado para definir processos de software e para o desenvolvimento de softwares e seus componentes. O âmbito do SPEM é limitado aos elementos mínimos necessários para definir modelos de processo de software, sem adição de funções específicas para domínios de desenvolvimento ou disciplinas (por exemplo, gerenciamento de projetos). O objetivo é acomodar uma ampla gama de métodos e processos de desenvolvimento de diferentes estilos, culturas, comunidades, níveis de formalismo métodos [SPEM, 2008].

O SPEM 2.0 não pretende definir uma linguagem de modelagem de processo genérico, nem mesmo fornecer seus conceitos de modelagem. O SPEM 2.0 é útil para capacitar o implementador na escolha da abordagem de modelagem do comportamento genérico que melhor se adapta às suas necessidades, além de fornecer elementos específicos para descrever processos de desenvolvimento. Em outras palavras, o SPEM 2.0 tem foco no fornecimento de estruturas de informação adicionais que são necessárias para processos modelados com UML 2.0, *Business Process Modeling Notation* (BPMN) ou *Business Process Definition Metamodel* (BPDm) com a finalidade de descrever um processo de desenvolvimento real [SPEM, 2008].

Os princípios fundamentais do SPEM são provenientes da ideia que processo é uma colaboração entre entidades abstratas ativas, chamadas de papéis de processo (*process roles*), que realizam operações chamadas de atividades (*activities*) em entidades chamadas de produtos de trabalho (*work products*). Os relacionamentos entre esses elementos são apresentados na Figura 1, por meio de um diagrama de classes UML [SPEM, 2008], na qual estão representados os relacionamentos entre os estereótipos fundamentais de processo [OMG, 2005; SPEM, 2008].



Figura 1 - Modelo Conceitual SPEM [Adaptado de OMG, 2005].

Os estereótipos SPEM utilizados neste trabalho são apresentados na Tabela 1, na qual a primeira coluna contém o nome do estereótipo, a segunda contém a descrição do estereótipo e a terceira contém a notação utilizada para representar o estereótipo. As relações entre os elementos do SPEM 2.0 utilizados na modelagem do processo são apresentadas na Tabela 2.

Tabela 1 - Elementos utilizados para a modelagem do processo.

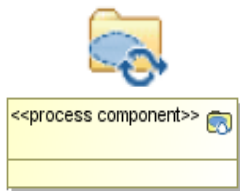
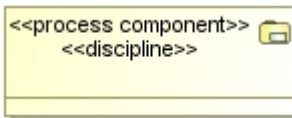
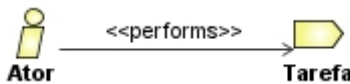
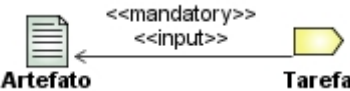
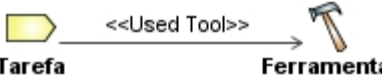
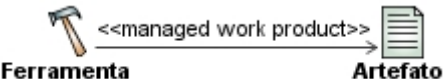
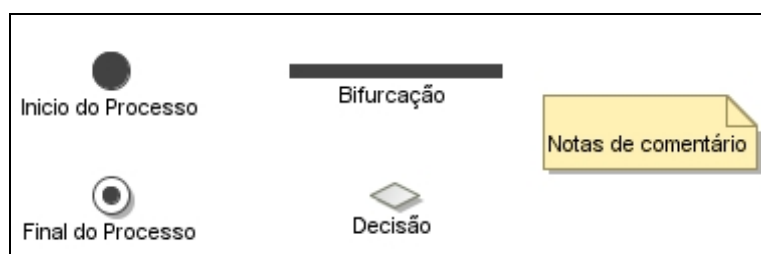
Estereótipos	Descrição	Notação
Componente do processo (<i>Process component</i>)	Um componente do processo contém exatamente um processo representado por uma atividade e define um conjunto de artefatos de software de entradas e saídas de um componente do processo.	
Processo (<i>Process</i>)	É a descrição completa de um processo, em termos de papéis, artefatos, conjuntos de trabalho, atividades e guias.	
Diagrama de fluxo de trabalho do SPEM (<i>SPEM Workflow diagram</i>)	É uma representação do diagrama de fluxo de atividades utilizado para descrever a interação entre as disciplinas, as atividades e as tarefas do processo.	
Diagrama de detalhamento de atividade do SPEM (<i>SPEM activity detail diagram</i>)	É uma representação do diagrama de fluxo de atividades utilizado para descrever a interação entre os atores, as tarefas, as ferramentas e os artefatos.	
Disciplina (<i>Disciplines</i>)	Representa um padrão de recurso e é usada para definir os fluxos de trabalho em termos de Atividades e Tarefas.	
Atividade (<i>Activity</i>)	É um elemento que define as unidades básicas de trabalho dentro de um processo, bem como um processo em si. Em outras palavras, toda atividade representa um processo no SPEM 2.0.	
Definição de tarefa (<i>Task definition</i>)	É um elemento que define um trabalho que deve ser executado. Uma tarefa é associada à entrada e à saída que podem ser de caráter obrigatório ou opcional.	
Definição de papel (<i>Role definition</i>)	É um elemento de conteúdo que define quem realiza e é responsável por um conjunto de produtos de trabalho. As definições de papéis são usadas por tarefa.	
Definição de ferramenta (<i>Tool definition</i>)	É um método de elementos especiais que podem ser utilizados para especificar a participação de uma ferramenta em uma tarefa.	
Definição de produto de trabalho (<i>Work product definition</i>)	É qualquer elemento de conteúdo utilizado, modificado e produzido pelas tarefas. Uma definição de produto de trabalho pode estar relacionada com outras definições de produto de trabalho.	

Tabela 2 – Relações entre elementos utilizados na modelagem do PERAF.

Descrição	Notação
Relação na qual um ator realiza uma tarefa.	
Relação na qual um artefato é entrada para uma tarefa.	
Relação na qual uma tarefa é desempenhada com o auxílio de uma ferramenta.	
Relação na qual uma ferramenta produz um artefato como saída da tarefa que a utiliza.	

Além dos estereótipos da Tabela 2 usados no SPEM, também foram utilizados alguns elementos da UML para a modelagem do processo (Figura 2).

**Figura 2 – Elementos da UML utilizados na modelagem do PERAF.**

2.3. Engenharia de Requisitos

Existe uma falta de consenso na utilização do termo “engenharia de requisitos”. A exemplo: Sommerville [Sommerville, 2007] utiliza o termo *engenharia de requisitos*; outros, como Leffingwell e Widrig [Leffingwell e Widrig, 2003], referem-se a ele como *gerência de requisitos*; além disso, existem aqueles que tratam somente como *domínio de requisitos*. Para esse trabalho foi adotado o termo engenharia de requisitos.

O sucesso do desenvolvimento de software está fortemente ligado em quão bem são entendidas as necessidades e o ambiente dos usuários. Os requisitos de software compreendem essas necessidades de entendimento e a ER é o processo que determina esses requisitos. O processo de ER envolve o entendimento das necessidades dos usuários, dos empreendedores e de outros envolvidos no projeto, assim como o entendimento do contexto no qual o software a ser desenvolvido será utilizado. O processo de ER envolve a modelagem, a análise, a negociação e a documentação dos requisitos do software. Ele

também aborda a validação dos documentos de requisitos e o gerenciamento da evolução desses requisitos [Cheng e Atlee, 2007].

O processo de ER atua de modo a fornecer um mecanismo apropriado para entender o que o cliente deseja, analisando as necessidades, avaliando a exequibilidade, negociando uma condição razoável, especificando a solução de modo não ambíguo, validando a especificação e gerindo os requisitos à medida que eles são transformados em um software operacional. O processo de ER precisa ser adaptado às necessidades do processo, do projeto, do produto e do pessoal que realiza o trabalho [Pressman, 2010].

Um dos aspectos mais importantes da ER é o fato dos requisitos mudarem em decorrência de vários fatores, como, por exemplo, erros de análise ou de mudanças de ambiente. Assim, o processo de ER não pode ser considerado como uma atividade com início e fim durante o ciclo de vida do software, mas que ela está espalhada por todo o ciclo de desenvolvimento [SWEBOK, 2004].

Para a descrição do processo são adotadas as terminologias baseadas nas propostas de Paula Filho [Paula Filho, 2009] e de Wiegers [Wiegers, 2003]. Assim, o processo é subdividido nas disciplinas Desenvolvimento de Requisitos e Gerenciamento de Requisitos.

A disciplina Desenvolvimento de Requisitos envolve todas as atividades relacionadas à reunião, avaliação e documentação dos requisitos de software. Para se ter sucesso nessa disciplina é necessário realizar ciclos de refinamento dos requisitos, evoluindo seu detalhamento e confirmando a corretude com o cliente. Ao se analisar as atividades compreendidas na disciplina *Desenvolvimento de Requisitos*, percebe-se que é nesta disciplina que acontece a compreensão do desejo do cliente, suas necessidades e os demais procedimentos que irão dar suporte ao desenvolvimento do projeto [Wiegers, 2003; Pressman, 2010].

A disciplina Gerenciamento de Requisitos é composta por um conjunto de atividades que ajudam a equipe do projeto a identificar, priorizar, controlar, rastrear e manter requisitos e suas modificações durante todo o projeto [Wiegers, 1999; Ameche e Ketabchi, 2007; Pressman, 2010]. De modo geral, a disciplina de gerenciamento de requisitos é focada no controle da informação relacionada aos requisitos, em especial na preservação da integridade dessa informação durante o ciclo de vida do requisito e, ao mesmo tempo, respeitando as mudanças no software e no ambiente [Gotel e Mader, 2009].

2.3.1. Processos de ER

Nessa Subseção, são descritos três processos de ER: proposto por Wiegers [Wiegers, 2003], proposto por Pressman [Pressman, 2010] e proposto por Sommerville [Sommerville, 2007]. Além disso, é apresentada uma comparação entre os três processos.

2.3.1.1. Processo de ER Proposto por Wiegers

Nessa subseção, é descrito o processo de ER proposto por Wiegers [Wiegers, 2003]. Esse processo é dividido nas disciplinas de (i) Desenvolvimento de Requisitos e de (ii) Gerenciamento de Requisitos. A seguir, são descritas as disciplinas, as suas subdisciplinas e as atividades do processo.

A disciplina *Desenvolvimento de Requisitos* envolve as atividades relacionadas à reunião, à avaliação e à documentação dos requisitos de um software. As suas subdisciplinas são:

- *Elicitação*: os requisitos são elicitados (requisitos de negócio, de usuário e funcionais). Eles são elicitados de diferentes fontes, em diferentes momentos do projeto e precisam ser documentados de diferentes maneiras. . Essa subdisciplina apresenta as seguintes atividades a serem realizadas: i) definir um processo de desenvolvimento de requisitos; ii) escrever um documento de visão e um documento de escopo; iii) identificar as classes de usuários e suas características; iv) selecionar um representante para cada classe de usuário; v) Estabelecer grupos focais de usuários típicos; vi) trabalhar com usuários representativos para identificar casos de uso; vii) identificar eventos de sistema e respostas; viii) realizar *workshops* de elicitación de requisitos; ix) observar o usuário em seu ambiente de trabalho; x) examinar os relatórios de problemas dos atuais softwares para levantar ideias de requisitos, e; xi) reutilizar requisitos entre projetos;
- *Análise*: os requisitos são refinados para garantir o entendimento dos envolvidos no projeto e detalhados para se encontrar possíveis erros, omissões e outras deficiências. Essa subdisciplina envolve a decomposição dos requisitos em alto nível de modo mais detalhado, criação de protótipos, avaliação de viabilidade e negociação de prioridades. A análise apresenta as seguintes atividades: i) desenhar o diagrama de contexto; ii) criar interfaces de usuários e protótipos técnicos do produto; iii) analisar a viabilidade dos requisitos; iv) priorizar os requisitos; v) Modelar os requisitos; vi) criar um dicionário de dados; vii) alocar os requisitos a

subsistemas, e; viii) aplicar o Desdobramento da Função Qualidade (Quality Function Deployment - QFD) ;

- *Especificação*: a documentação dos requisitos deve ser feita de forma consistente, acessível e revisável, podendo haver registro em forma de documentos de visão e documentos de escopo. Os requisitos de usuários normalmente são representados em forma de casos de uso e em formato de tabelas. Essa subdisciplina apresenta as seguintes atividades: i) adotar um gabarito (template) de documentação de requisitos; ii) identificar a fonte dos requisitos; iii) definir uma identificação única para os requisitos; iv) armazenar as regras de negócio, e; v) especificar atributos de qualidade;
- *Validação*: os requisitos devem ser revisados de modo a garantir que estão corretos, deve-se gerar demonstrativos de qualidade e satisfação das necessidades dos clientes, por fim deve-se escrever casos de teste e garantir que não ocorram ambiguidades. Os requisitos são usados para formulação de testes de aceitação, e de software. Essa subdisciplina apresenta as seguintes atividades: i) inspecionar documentos de requisitos; ii) escrever testes para os requisitos, e; iii) definir critérios de aceitação.

A disciplina *Gerência de Requisitos* reúne as atividades necessárias para manter a integridade e a exatidão dos requisitos ao longo do projeto. As suas subdisciplinas são:

- *Controle de Mudanças*: há tomada de decisão quanto a aceitar ou não uma mudança proposta a partir da sua análise e garantir que as atualizações sejam devidamente documentadas. Essa subdisciplina apresenta as seguintes atividades: i) propor mudanças; ii) analisar impactos; iii) tomar decisões; iv) atualizar o documento de requisitos; v) atualizar planos, e; vi) mensurar volatilidade de requisitos;
- *Controle de Versões*: o documento de requisitos é único para cada versão. Cada membro da equipe tem acesso à versão corrente dos documentos e as mudanças são claras e comunicadas aos membros da equipe. Essa subdisciplina apresenta as seguintes atividades: i) definir um esquema de identificação de versão; ii) identificar a versão dos documentos de requisitos, e; iii) identificar a versão individual dos requisitos;
- *Rastreamento do Status dos Requisitos*: os requisitos são classificados em diversas categorias de *status* de modo a permitir o monitoramento da completude dos requisitos e do projeto. Essa subdisciplina apresenta as seguintes atividades: i)

definir estados dos requisitos; ii) armazenar o estado de cada requisito, e; iii) reportar o estado de distribuição de todos os requisitos;

- *Rastreamento dos Requisitos*: são criadas ligações (*links*) entre os requisitos de modo a rastrear as interferências destes sobre os demais requisitos dos documentos de requisitos e demais elementos do software. Essa subdisciplina apresenta as seguintes atividades: i) definir ligações para outros requisitos, e; ii) definir ligações para outros elementos do software.

2.3.1.2. Processo de ER Proposto por Pressman

Nessa subseção, é descrito o processo de ER proposto por Pressman [Pressman, 2010]. Apesar do autor não definir de forma explícita um processo de ER, são definidas atividades que podemos considerar que seriam componentes de um processo aqui conjecturado. Esse processo é dividido em sete atividades: Concepção, Levantamento, Elaboração, Negociação, Especificação, Validação e Gestão de Requisitos. Cabe ressaltar que algumas destas atividades podem ocorrer em paralelo e que são adaptadas às necessidades do projeto, tentam definir o que o cliente deseja e servem para estabelecer uma fundação sólida para o projeto e a construção do que o cliente obtém. As atividades, as subatividades, as tarefas e pontos relevantes do processo são:

- *Concepção*: os engenheiros de software perguntam uma série de questões livres de contexto ao cliente. A intenção é estabelecer um entendimento básico do problema, saber quem são as pessoas que querem uma solução, conhecer a natureza da solução desejada e ter uma avaliação preliminar da efetividade da comunicação e colaboração entre cliente e desenvolvedor;
- *Levantamento*: os objetivos, as funcionalidades e as necessidades requeridos pelos envolvidos no negócio para o software ou produto são reunidos. A atividade possui diversos desafios para os envolvidos no projeto, tais como: o software pode ser mal definido, pode haver problemas de entendimento pelos clientes e usuários ou os requisitos podem mudar ao longo do tempo. Para ajudar a contornar esses problemas, os engenheiros de softwares devem abordar a atividade de coleta dos requisitos de um modo organizado;
- *Elaboração*: o objetivo é expandir e refinar as informações obtidas do cliente durante a concepção. Essa atividade enfatiza o desenvolvimento de um modelo técnico refinado das funções, das características e das restrições de software;

- *Negociação*: um processo de negociação entre os envolvidos no projeto é realizado quando objetivo é priorizar e solucionar possíveis conflitos entre os requisitos do software. Essas negociações são necessárias por causa das limitações do negócio e das proposições conflitantes de requisitos. Durante as negociações, os riscos associados a cada requisito são identificados e analisados, além da realização de “estimativas” grosseiras do esforço de desenvolvimento utilizadas para avaliar o impacto de cada requisito no custo do projeto e no prazo de entrega. Usando uma abordagem iterativa, requisitos são eliminados, combinados e/ou modificados de modo que cada parte alcance algum grau de satisfação;
- *Especificação*: o produto de trabalho final é produzido pelo engenheiro de requisitos, podendo ser um documento escrito, um modelo gráfico, um modelo matemático formal, uma coleção de cenários de uso, um protótipo ou qualquer combinação desses elementos. Esse produto descreve as funcionalidades oferecidas pelo software e servirá como fundamento das atividades de Engenharia de Software subsequentes;
- *Validação*: os produtos de trabalho resultantes da ER são avaliados quanto à qualidade durante o processo de validação. A validação de requisitos examina a especificação para garantir que os requisitos do software tenham sido declarados de modo não ambíguo; que as inconsistências, as omissões e os erros tenham sido detectados e corrigidos e que os produtos de trabalho estejam de acordo com as normas estabelecidas para o processo, o projeto e o produto. O principal mecanismo de validação de requisitos é a revisão técnica formal. A equipe de revisão que valida os requisitos inclui engenheiros de software, clientes, usuários e outros interessados que examinam a especificação procurando por erros de conteúdo ou de interpretação, áreas em que esclarecimentos podem ser necessários, informações omissas e inconsistências, e;
- *Gestão de Requisitos*: a gestão de requisitos começa com a identificação de cada requisito com um identificador único. Posteriormente, são desenvolvidas tabelas de rastreamento, relacionando cada requisito com um ou mais aspectos do software. O objetivo dessas tabelas de rastreamento é possibilitar a pesquisa de forma rápida dos requisitos e permitir o entendimento sobre como as modificações em um requisito afetarão o software a ser construído.

2.3.1.3. Processo de ER Proposto por Sommerville

Nessa subseção, é descrito o processo de ER proposto por Sommerville [Sommerville, 2007]. O objetivo desse processo é criar e manter a documentação dos requisitos do software a ser desenvolvido. O processo geral inclui quatro subprocessos de alto nível de ER: avaliação da utilidade do software para a empresa (estudo de viabilidade); obtenção de requisitos (elicitación e análise); conversão desses requisitos em alguma forma padrão (especificação), e; verificação de que os requisitos realmente definem o software que o cliente deseja (validação). O gerenciamento de requisitos é uma atividade adicional da ER do processo proposto pelo autor, que se dedica a gerenciar as modificações nos requisitos.

As atividades de ER, mostradas na Figura 3, dizem respeito à obtenção, à documentação e à verificação dos requisitos [Sommerville, 2007].

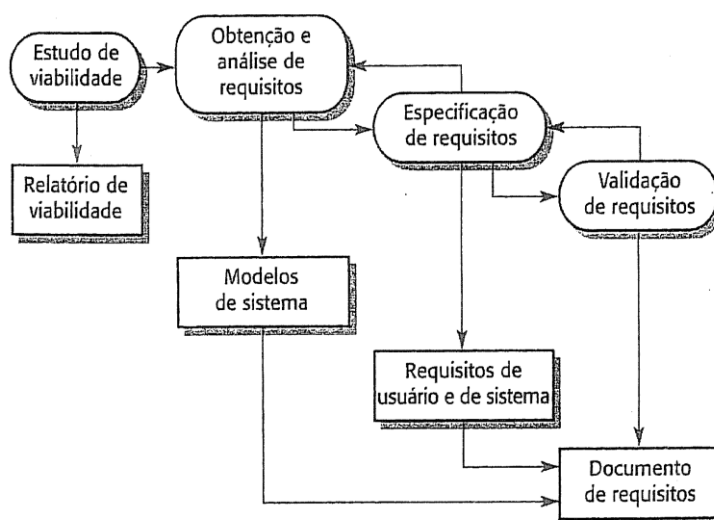


Figura 3 - Processo de ER [Sommerville, 2007].

Estudo de viabilidade: é um estudo breve que se destina a responder algumas perguntas sobre como o software contribui para os objetivos gerais da organização, se o software pode ser implementado com a utilização de tecnologia atual dentro das restrições de custo e de prazo, e se o software pode ser integrado com outros software em operação. A entrada para o estudo de viabilidade é uma descrição geral do software e de como ele será utilizado dentro de uma organização. Os resultados do estudo de viabilidade devem ser um relatório que recomende se vale a pena ou não realizar o processo de ER e o processo de desenvolvimento de software.

Obtenção e análise de requisitos: Nessa atividade, os membros da equipe técnica trabalham para descobrir mais informações sobre o domínio da aplicação, que serviços o software deve fornecer, o desempenho exigido do software, as restrições de hardware e

assim por diante. O levantamento e análise dos requisitos podem envolver diferentes pessoas em uma organização. O termo parte interessada (*stakeholder*) é utilizado para definir qualquer pessoa que terá alguma influência direta ou indireta sobre os requisitos do software. Um modelo genérico do processo de levantamento e análise é mostrado na Figura 4.

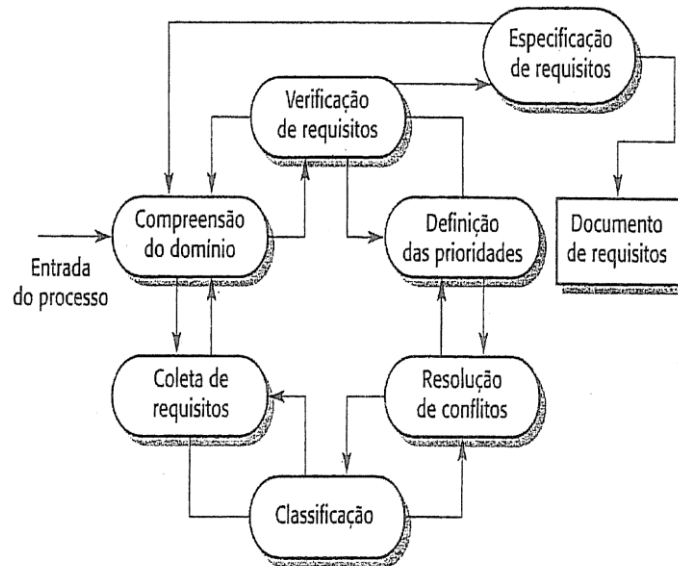


Figura 4 - Processo de obtenção e análise de requisitos [Sommerville, 2007].

As atividades de processo de obtenção e análise de requisitos são:

1. *Compreensão do domínio*: os analistas devem desenvolver sua compreensão do domínio da aplicação;
2. *Coleta de requisitos*: é um processo de interagir com as partes interessadas do software para descobrir seus requisitos. Obviamente, a compreensão do domínio se desenvolve mais durante esta atividade;
3. *Classificação*: essa fase considera o conjunto não estruturado dos requisitos e organiza em grupos coerentes;
4. *Resolução de conflitos*: quando múltiplas partes interessadas estão envolvidas, os requisitos apresentarão conflitos. Essa atividade se ocupa de encontrar e solucionar esses conflitos;
5. *Definição das prioridades*: em qualquer conjunto de requisitos, alguns serão mais importantes do que outros. Esse estágio envolve a interação com as partes interessadas, para descobrir os requisitos mais importantes, e;

6. *Verificação de requisitos*: os requisitos são verificados, a fim de se descobrir se eles são completos, consistentes e se estão em concordância com o que as partes interessadas realmente desejam do software.

Validação de requisitos: A validação dos requisitos se ocupa de mostrar que os requisitos realmente definem o software que o cliente deseja, devendo ela ser atenta para a elaboração de uma versão completa do documento de requisitos. Ela tem muito em comum com a análise de requisitos, uma vez que se preocupa em descobrir problemas nos requisitos. Contudo, esses são processos distintos, uma vez que a análise envolve trabalhar com requisitos incompletos.

Durante o processo de validação de requisitos, diferentes tipos de verificação devem ser realizados sobre os requisitos no documento de requisitos. Dentre as verificações destacam-se:

1. *Verificação de validade*: um usuário pode pensar que um software é necessário para realizar certas funções. Contudo, mais estudos e análises podem identificar funções adicionais ou diferentes, que são exigidas. Os softwares têm diversos usuários com necessidades diferentes e qualquer conjunto de requisitos é inevitavelmente uma solução conciliatória da comunidade de usuários;
2. *Verificação de consistência*: os requisitos em um documento não devem ser conflitantes, ou seja, não devem existir restrições contraditórias ou descrições diferentes para uma mesma função do software;
3. *Verificação de completude*: os documentos de requisitos devem incluir requisitos que definam todas as funções e restrições exigidas pelo usuário do software;
4. *Verificação de realismo*: utilizando o conhecimento da tecnologia existente, os requisitos devem ser verificados, a fim de assegurar que eles realmente podem ser implementados, e;
5. *Facilidade de verificação*: para reduzir o potencial de divergências entre cliente e fornecedor, os requisitos do software devem sempre ser descritos de modo que possam ser verificados.

Gerenciamento de requisitos: o gerenciamento de requisitos corresponde ao processo de compreensão e controle das mudanças nos requisitos, sendo realizado em conjunto com outros processos da ER. O planejamento se inicia ao mesmo tempo em que o levantamento inicial de requisitos, e o gerenciamento ativo de requisitos deve começar assim que um

esboço da versão do documento de requisitos estiver disponível. O processo é dividido em duas subatividades: planejamento de requisitos e gerenciamento de mudança.

Para Sommerville [Sommerville, 2007] o planejamento é o primeiro estágio essencial de gerenciamento de requisitos e é nele que são estabelecidos os níveis de detalhes exigidos para o gerenciamento de requisitos. O estágio de gerenciamento de requisitos apresenta e decide sobre as seguintes tarefas:

1. *Identificação dos requisitos*: cada requisito precisa ser identificado de modo único, facilitando o seu referenciamento e rastreamento;
2. *Processo de gerenciamento de mudanças*: trata-se do conjunto de atividades que avalia o impacto e o custo das mudanças;
3. *Políticas de facilidade de rastreamento*: essas políticas definem as relações entre os requisitos e o projeto, a forma de registro dos requisitos e como esses registros devem ser mantidos, e;
4. *Suporte de ferramentas CASE*: o gerenciamento de requisitos envolve processar uma grande quantidade de informações sobre os requisitos. As ferramentas que podem ser utilizadas vão desde softwares especializados de gerenciamento de requisitos até planilhas de cálculo e softwares de bancos de dados.

Segundo Sommerville [Sommerville, 2007], o gerenciamento de requisitos deve ser aplicado a todas as mudanças propostas para os requisitos (Figura 5). A vantagem de se utilizar um processo formal para o gerenciamento de mudanças é que todas as propostas de mudança são tratadas de modo consistente e que as mudanças no documento de requisitos são feitas de maneira controlada. Há três tarefas principais na subatividade de gerenciamento de mudanças:

1. *Análise do problema e especificação da mudança*: o processo começa com a identificação de um problema com os requisitos ou, algumas vezes, com uma proposta específica de mudança. Nesse estágio, é realizada a análise do problema ou da proposta de mudança, a fim de verificar a sua validade. Uma proposta mais específica de mudança nos requisitos pode então ser efetuada;
2. *Análise e custo da mudança*: o efeito da mudança é avaliado, utilizando-se informações sobre a facilidade de rastreamento e o conhecimento geral dos requisitos do software. O custo da mudança é estimado em termos de modificações no documento de requisitos e, se apropriado, no desenho do software e na

implementação. Uma vez concluída essa análise, é tomada uma decisão sobre prosseguir com a alteração de requisitos ou não, e;

3. *Implementação de Mudança*: o documento de requisitos e, quando for necessário, o desenho do software e a implementação são modificados. O documento de requisitos deve ser organizado de maneira que as mudanças possam ser acompanhadas sem muito esforço. Assim como ocorre com os programas, a facilidade de modificação em documentos é alcançada ao se minimizar referências externas e ao se tornar as seções do documento tão modulares quanto possível.

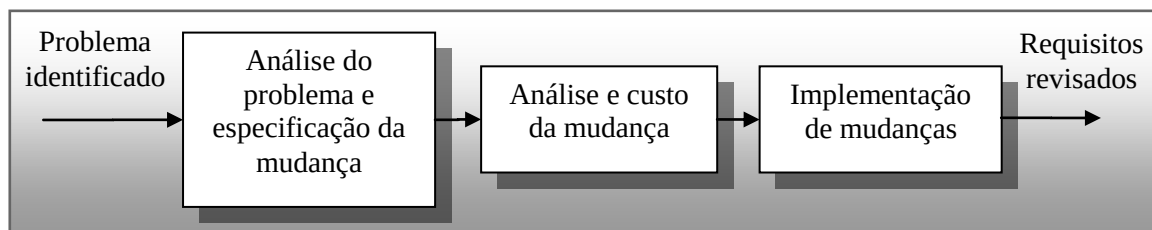


Figura 5 - Gerenciamento de mudanças de requisitos [Sommerville, 2007].

2.3.1.4. Comparação entre os processos

Foram encontradas similaridades e divergências nos processos de ER propostos por Sommerville, Pressman e Wiegers. A Figura 6 apresenta uma comparação entre os processos. Dentre as divergências foi possível interpretar que:

- Sommerville apresenta um estudo de viabilidade como atividade inicial. Os demais autores iniciam a atividade de elicitação proposta por Wiegers e a de Concepção proposta por Pressman;
- Wiegers caracteriza a elicitação, análise, especificação e validação como subdisciplinas da disciplina desenvolvimento de requisitos;
- A atividade de elicitação de Wiegers engloba as atividades de concepção e levantamento propostas por Pressman. Do mesmo modo, a atividade de análise proposta por Wiegers engloba as atividades de elaboração e negociação propostas por Pressman. Por outro lado, Sommerville trata as subdisciplinas de elicitação e análise propostas por Wiegers e as subatividades de concepção, levantamento, elaboração e negociação propostas por Pressman como uma única atividade denominada elicitação e análise de requisitos;
- Os três processos mencionados apresentam as atividades de especificação, validação e gerenciamento em seus processos. No entanto Wiegers subdivide a disciplina de gerenciamento em quatro subdisciplinas: controle de mudanças,

controle de versões, rastreamento do status dos requisitos e rastreamento dos requisitos. Sommerville as subdivide em duas subatividades: planejamento do gerenciamento de requisitos e gerenciamento de mudanças de requisitos, e;

- Em uma visão geral, os três processos apresentam níveis de detalhamento diferenciados para cada atividade. Além disso, eles adotam uma nomenclatura variada para lidar com cada nível de divisão do processo de ER. No entanto, ambos apresentam um conjunto de atividades similares que caracterizam claramente a divisão do processo de ER em: i) duas disciplinas²: Desenvolvimento de Requisitos e Gerenciamento de Requisitos, e; ii) oito atividades: Elicitação, Análise, Especificação de requisitos, Validação, Controle de mudanças, Controle de versões, Rastreabilidade do status dos requisitos e Rastreamento dos requisitos.

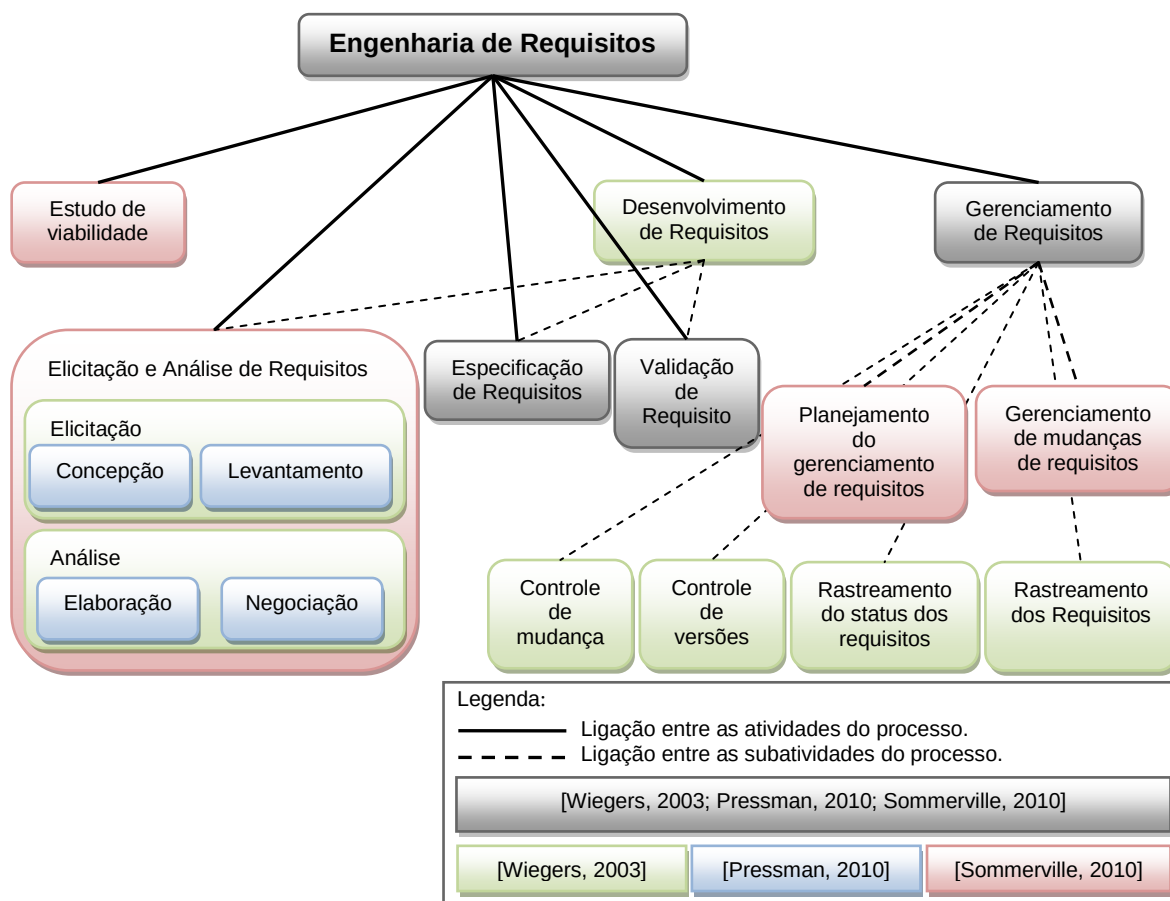


Figura 6 - Comparação entre os processos de ER propostos por Wieggers, Pressman e Sommerville.

² Padrão de nomenclatura proposto por Paula Filho [Paula Filho, 2009].

2.4. Metodologias de Desenvolvimento de Software (MDS)

A década de 1990 foi marcada por uma visão de que a melhor maneira de desenvolver um software de qualidade seria com o uso de métodos atualmente conhecidos como MT's. Estes métodos se baseiam no formalismo, no uso de análise, nos desenhos do software e nos rigorosos processos de desenvolvimento. Contudo, quando essa abordagem de desenvolvimento baseada em planos foi aplicada ao desenvolvimento de software em pequenas e médias empresas, o tempo gasto para determinar como o software deveria ser desenvolvido foi considerado muito extenso proporcionalmente ao tempo empregado no desenvolvimento do programa e em testes. Outro fator agravante era que, a medida que o conjunto de requisitos era alterado, era gerado um grande montante de retrabalho com atualizações da especificação, do desenho do software e do código fonte [Sommerville, 2007].

A insatisfação com essas abordagens levou um considerável número de desenvolvedores de software a propor novos métodos. Métodos que permitissem que a equipe de desenvolvimento se concentrasse mais no software, em vez de focar no desenho do software e na documentação do mesmo. Estes métodos ficaram conhecidos como MA's. Estes novos métodos permitiram o desenvolvimento de softwares nos casos em que a percepção do cliente não era estável, ou quando as plataformas tecnológicas estavam em constante mudança. Acreditava-se que os processos de desenvolvimento tradicionais seriam incapazes de lidar com situações com esta instabilidade (Reed *et al.*, 2004). Assim, muitas empresas que usavam MT's se sentiram compelidas a aderir às práticas destes novos métodos, denominados MA's, com o objetivo de que os ganhos conseguidos com estes métodos fossem refletidos em seus projetos [Sommerville, 2007].

No entanto, os MA's diferem dos MT's, em especial na ER. Os MA's adotam uma abordagem iterativa para lidar com os requisitos e os MT's seguem um procedimento formal para produzir uma especificação completa para esses requisitos. A ERMT visa descrever os requisitos de uma forma mais completa sendo centrada em descobrir os requisitos antes de iniciar o desenvolvimento. Por outro lado, a ERMA é mais dinâmica e adaptável, e está distribuída ao longo do desenvolvimento do software [Paetsch *et al.*, 2003; Cao *et al.*, 2008].

Combinar os benefícios dos MA's com os benefícios dos MT's em uma abordagem híbrida pode ser uma solução efetiva. Esta combinação agruparia a capacidade de

acomodar mudanças, teria um foco maior nas pessoas do que nos processos e agregaria os benefícios da boa documentação voltados para manutenções futuras, suportes e treinamentos [Cao e Ramesh, 2008].

Tendo em vista as divergências entre a forma de como é realizada a ER nos principais MT's e Mas, nesta seção, apresenta-se uma comparação de aspectos de ER. As próximas subseções tratam de comparativos entre a ERMT e ERMA. As comparações estão divididas em duas visões, sendo elas: uma visão em alto nível de abstração, a qual descreve uma abordagem genérica quanto a MT's e MA's; e outra visão mais detalhada e que descreve como cada método aborda as disciplinas da engenharia de requisitos.

2.4.1. ERMT

Existe uma divergência em como alguns autores tratam os MTs. Como exemplo: Boehm e Turner utilizam o termo *métodos baseados em planejamento* [Boehm e Turner, 2005]; Boehm, Turner, Misra e outros também utilizam o termo *ciclo de vida tradicional* [Boehm e Turner, 2005; Misra *et al.*, 2006]; existem também, Nerur e outros que utilizam o termo *métodos tradicionais* [Nerur *et al.*, 2005]; e outros, como Qureshi e Hussain que utilizam as nomenclaturas: *modelos tradicionais de processo* ou *métodos pesados de desenvolvimento de software* [Qureshi e Hussain, 2006]. Para esse trabalho foi adotado o termo métodos tradicionais.

Os MT's são centrados em processos e são guiados pela crença de que as fontes de variações dos processos são identificáveis e podem ser eliminados pela contínua medição e refino dos mesmos. Assim, o desenvolvimento de software na abordagem tradicional é orientado por um modelo de ciclo de vida, tais como o modelo em cascata, o modelo espiral, ou algumas variações deles [Nerur *et al.*, 2005].

Os principais MT's são baseados nos modelos: Modelo Cascata, Modelo Incremental, Modelo *Rapid Application Development* (RAD), Modelo de Prototipagem e Modelo Espiral.

A ER no modelo Cascata está presente em todo ciclo de elaboração do software. No entanto, a especificação do software está concentrada no início do desenvolvimento do software. Ela segue os passos de definição e especificação dos requisitos, seguida de uma fase de desenho do software. Após essas fases os requisitos serão usados nos testes do software e na manutenção do software [Sommerville, 2007; Pressman, 2010].

Em oposição à especificação do software centrada no início do projeto, a ER do modelo Incremental é dividida sequências lineares ao longo do tempo por todo ciclo de desenvolvimento. Ela segue os passos de identificação (em linhas gerais) do software seguida de uma definição dos incrementos. Após essa definição, em cada ciclo, os requisitos são especificados e o desenho do software é construído. Os requisitos de cada ciclo são utilizados novamente na fase de testes, para a especificação do próximo ciclo e manutenção do software [Sommerville, 2007; Pressman, 2010].

A ER do modelo RAD segue os mesmos preceitos do modelo Cascata e é considerada uma adaptação “de alta velocidade” do mesmo, pois nele são paralelizados ciclos de especificação e desenvolvimento e cada software produzido em um ciclo é tratado como um componente [Pressman, 2010].

A ER do modelo Prototipagem está centrada no início do ciclo de elaboração do software onde é criado um documento de especificação de requisitos em caráter inicial. A partir dessa especificação são gerados protótipos do software com ciclos regulares de validação com o cliente até que o protótipo atenda as necessidades do cliente. Por fim, o protótipo é descartado, e uma especificação de requisitos formal é produzida [Pressman, 2010; Sommerville, 2007].

Por último, o modelo Espiral engloba a natureza interativa do Método Prototipagem com os aspectos controlados e sistemáticos do Método Cascata. A principal diferença do Método Espiral dos outros métodos de desenvolvimento de software é avaliação explícita dos riscos que podem ocorrer no projeto de software através de: análise mais detalhada, prototipação e simulação durante todos os *loops* da espiral [Sommerville, 2007; Pressman, 2010].

2.4.2. ERMA

Os MA's possuem abordagem iterativa para especificação, desenvolvimento e entrega de software e foram criados principalmente para apoiar o desenvolvimento de software para automatizar a lógica de negócio cujos requisitos de software mudam dinâmica e constantemente durante o processo de desenvolvimento. Eles consideram muito importante entregar um software funcional rapidamente aos clientes, que podem, então, propor novos requisitos e alterações a serem incluídas nas versões posteriores do software [Sommerville, 2007].

Nessa subseção, é apresentada uma revisão sobre a ERMA. A seleção dos métodos foi baseado nos principais métodos abordados na literatura, com enfoque nos métodos simultaneamente abordados nos trabalhos de Abrahamsson e outros e de Boehm [Abrahamsson *et al.*, 2002; Boehm e Turner, 2005]. São eles:

- *Extreme Programming* (XP) [Beck, 1999; Beck e Fowler, 2000; Beck e Andres, 2004];
- Scrum [Schwaber e Beedle, 2001];
- *Adaptative Software Development* (ASD) [Highsmith, 2000];
- *Dynamic System Development Method* (DSDM) [Stapleton, 1997];
- *Feature Driven Development* (FDD) [Palmer e Felsing, 2002 *apud* Abrahamsson *et al.*, 2002].

Obs.: A Família Crystal [Cockburn, 2002] não foi abordada devido à dificuldade de obtenção de informações sobre o método.

Embora esses métodos estejam baseados na noção de desenvolvimento e entregas incrementais, cada um deles propõe sua própria estruturação considerando diferentes aspectos. Contudo, compartilham um conjunto de princípios comuns, os quais são apresentados em um documento conhecido como Manifesto Ágil [Manifesto Ágil, 2001], o qual apresenta doze princípios:

1. A maior prioridade é satisfazer o cliente por meio de entrega breve e contínua de um “software de valor”³;
2. Aceitar mudanças de requisitos, mesmo que tarde no desenvolvimento. Processos ágeis aproveitam as mudanças, para que o cliente possa tirar vantagens competitivas;
3. Entregar software funcionando com frequência, menos de um ou dois meses, de preferência para os períodos mais curtos;
4. Pessoas relacionadas a negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante o curso do projeto;
5. Construir projetos ao redor de indivíduos motivados, proporcionando a eles ambiente e suporte necessários, e confiar que farão seu trabalho;
6. A maneira mais eficiente e eficaz de transmitir informações para e dentro de um time de desenvolvimento é com conversas face a face;

³ Software de valor: Alguma parte de funcionalidade do negócio que prove ao cliente que a aplicação de software em desenvolvimento atende as suas necessidades de negócio [Manifesto Ágil, 2001].

7. Software funcionando é a principal medida de progresso;
8. Processos ágeis promovem um ambiente sustentável. Patrocinadores, desenvolvedores e usuários devem ser capazes de manter indefinidamente um ritmo constante;
9. Contínua atenção à excelência técnica e bom desenho (design) melhora a agilidade;
10. Simplicidade - a arte de maximizar a quantidade de trabalho que não precisou ser feito - é essencial;
11. As melhores arquiteturas, requisitos e desenhos emergem de times auto organizáveis, e;
12. Em intervalos regulares, o time reflete sobre como se tornar mais efetivo ajustando e otimizando seu comportamento.

As próximas subseções tratam da ERMA de desenvolvimento de software: XP, Scrum, DSDM, FDD e ASD.

2.4.2.1. Método Extreme Programming (XP)

XP é um método ágil de desenvolvimento de software baseado em cinco valores, em concordância com o Manifesto Ágil, sendo eles: Comunicação; Simplicidade; Coragem; Feedback, e; Respeito. Além dos valores, o método XP possui 12 práticas como propostas que traduzem os valores das ações para serem executadas no dia-a-dia do trabalho de uma organização. Essas práticas vêm sofrendo evolução com o intuito de inserir uma maior flexibilidade na aplicação das práticas originais do método. Nesse período é notável que o foco da área de testes sofreu mudanças significativas, sendo que no ano 2000 o método XP era voltado para testes de unidade e no ano de 2004 foi enfatizada também a importância dos testes de caixa preta em direção ao teste exploratório [Beck, 1999; Beck e Fowler, 2000; Paetsch, 2003, Beck e Andres, 2004; Coram e Bohner, 2005].

A ER do Método XP inicia antes da escrita das histórias de usuários⁴, quando os clientes devem pensar sobre o que eles esperam que o software faça. Pensar em uma funcionalidade específica pode levar a mais ideias e mais histórias de usuário. Cada história de usuário é discutida de uma maneira aberta, antes da implementação. Inicialmente, os desenvolvedores pedem detalhes suficientes para serem capazes de

⁴ História de usuários: cartões de indexação que descrevem as características e funcionalidades requeridas para o software a ser construído [Pressman, 2010].

estimar o esforço para a implementação das histórias de usuário e documentá-las em cartões, a equipe de desenvolvimento dividirá estes cartões em tarefas e estimará o esforço e os recursos necessários para a implementação. De posse destas estimativas e cientes do tempo disponível, os clientes definem as histórias de usuário para serem abordadas na próxima iteração, priorizando-as para implementação e escolhendo as que podem ser usadas imediatamente de modo a proporcionar apoio útil ao negócio. Naturalmente, quando os requisitos mudarem, as histórias de usuário que não foram implementadas acompanham a mudança, ou seja, são alteradas ou são descartadas. Se forem necessárias mudanças na parte do software que foi entregue, novos cartões de histórias de usuário serão desenvolvidos e o cliente decidirá se essas mudanças devem ter prioridade sobre as demais funcionalidades [Beck, 1999; Paetsch *et al.*, 2003; Beck e Andres, 2004; Sommerville, 2007].

2.4.2.2. Método Scrum

Scrum é um método ágil de gestão do processo de desenvolvimento de softwares direcionado para a gerência de projetos e focado em como uma equipe deve trabalhar em conjunto para produzir um trabalho de qualidade em um ambiente de constantes mudanças sem determinar como a equipe executará as tarefas de programação [Paetsch *et al.*, 2003; Coram e Bohner, 2005; Franco, 200; Fadel e Silveira, 2010].

A ER do Método Scrum inicia-se com a coleta dos requisitos e apregoa o contato com o cliente, no entanto, o Scrum não define como os requisitos devem ser coletados. Uma vez coletados, todos os requisitos considerados necessários são listados na reserva de pedidos para o produto (*product backlog*) pelo Proprietário do Produto (*Product Owner*) e pelo o Mestre Scrum (*Scrum Master*). Uma vez definida a reserva de pedidos para o produto, ela se torna um documento dinâmico no qual podem ser criados, removidos e atualizados os itens. O Proprietário do Produto prioriza os itens da reserva de pedidos e posteriormente é montada a reserva de pedidos da carreira (*Sprint Backlog*) que é composta pelos itens de prioridade mais alta da reserva de pedidos. Uma vez definida a reserva de pedidos da carreira os itens serão traduzidos em termos de tarefas requeridas para a sua implementação. Não são permitidas alterações na reserva de pedidos da carreira durante a carreira (*sprint*). No final de uma carreira uma reunião de revisão da carreira (*Sprint Review*) é realizada demonstrando o novo incremento do produto para o cliente e solicitando considerações. O conhecimento adquirido na reunião de revisão da carreira e da reserva de pedidos é usado na reunião de planejamento da próxima carreira (*Sprint*

Planning Meeting) [Highsmith, 2002; Paetsch *et al.*, 2003; Schwaber, 2004; Luduvig e Reinert, 2007].

2.4.2.3. Método Dynamic System Development Method (DSDM)

O Método DSDM consiste em um arcabouço (*framework*) para o desenvolvimento rápido de aplicações, cujos princípios estão apoiados em ideias de desenvolvimento iterativo e em uma constante e intensa colaboração entre o cliente e os desenvolvedores [Abrahamsson *et al.*, 2002; Highsmith, 2002; Paetsch *et al.*, 2003].

Segundo Highsmith [Highsmith, 2002] o método DSDM lida com a ER nas primeiras fases: a de estudo de viabilidade e de negócio. Durante essas fases os requisitos básicos são identificados. Posteriormente outros requisitos são identificados durante a fase de desenvolvimento. As iterações do método priorizam a identificação dos requisitos funcionais e não-funcionais mais importantes, que são preferencialmente documentados na forma de protótipos, ao invés de texto. A fase de “*Design*” e de “*Build*” refina os protótipos com o objetivo de levantar o maior número de requisitos (funcionais e não funcionais) e preparam o software para atender a esses requisitos. As fases de “*Design*” e de “*Build*” também identificam mais requisitos para serem implementados no terceiro ciclo iterativo. Quando as iterações de implementação acabam, o projeto pode voltar a qualquer uma das outras etapas, inclusive para o estudo de viabilidade do atendimento dos requisitos [Abrahamsson *et al.*, 2002; Highsmith, 2002].

2.4.2.4. Método FDD

O método Desenvolvimento Dirigido por Funcionalidades ou Características (FDD) é uma abordagem ágil e adaptável para desenvolvimento de software sendo realizado em um ciclo de vida iterativo e embasado em métodos bem definidos que permitem reutilização de requisitos. Este método se concentra nas fases de desenho e construção, com maior ênfase na modelagem e nas atividades de gerenciamento de projetos [Palmer e Felsing, 2002 *apud* Abrahamsson *et al.*, 2002; Highsmith, 2002; Paetsch *et al.*, 2003; Portella, 2009].

O método FDD não define a forma como os requisitos devam ser coletados, mas parte da premissa de que o projeto só inicia no momento em que os requisitos estiverem bem definidos pelos membros da equipe. Inicialmente é desenvolvido um modelo geral no qual os especialistas de domínio fornecem uma visão global dos requisitos. Este modelo

será então refinado em características de modo a construir a Lista de Características (*Features*)⁵. O processo é uma decomposição de uma área subjetiva, para atividades de negócio. Com base no modelo desenvolvido, o segundo processo construirá uma lista completa de características como produto. Se uma característica apresentar prazo de desenvolvimento superior à dez dias ela deverá ser decomposta. Uma vez que a lista de características foi gerada e revisada com o cliente, é feito agendamento pela equipe técnica para as demais características [Highsmith, 2002; Paetsch *et al.*, 2003].

2.4.2.5. Método ASD

O Método Desenvolvimento Adaptativo de Software (Adaptive Software Development - ASD) foi criado por James A. Highsmith como uma evolução do RAD e teve suas origens no ano de 1994. O Método ASD aplica diversos princípios ágeis, como o desenvolvimento incremental e a constante prototipação. Seus estágios iniciais são curtos e objetivam garantias e envolvimento do cliente. O ASD usa de um ciclo de vida especulativo-colaborativo de aprendizado [Abrahamsson *et al.*, 2002, Highsmith, 2002; Paetsch *et al.*, 2003; Portela, 2009].

No ASD os requisitos são identificados em forma de características, funcionalidades ou feições (*features*) do software que serão desenvolvidas durante as iterações do ciclo do ASD. Embora os documentos (por exemplo, um modelo de dados) possam ser definidos como resultados, eles estão sempre em segundo plano quando comparados a um recurso de software que fornece resultados diretos a um cliente. Os primeiros ciclos de um projeto de ASD devem ser curtos, de modo a garantir que o cliente esteja fortemente envolvido e confirmar a viabilidade do projeto. Os demais ciclos objetivam que as características evoluam ao longo de várias iterações à medida que os clientes forneçam retroalimentação nas reuniões de revisão. Os resultados dessas reuniões são documentados como requisições de mudança [Highsmith, 2002, Paetsch *et al.*, 2003].

2.4.3. Comparação entre as ER dos MT's e dos MA's

A forma com que os MA's lidam com os requisitos difere da forma com que os MT's lidam com eles. Na medida em que os MA's adotam uma abordagem iterativa para lidar com os requisitos e os MT's seguem um procedimento formal para produzir uma especificação completa. A ERMT visa descrever os requisitos de uma forma mais

⁵ Características (*Features*): Característica ou funcionalidade pequena o suficiente para ser implementada em no máximo duas semanas [Abrahamsson *et al.*, 2002].

completa sendo centrada em descobrir todos os requisitos antes de iniciar o desenvolvimento. Por outro lado a ERMA é mais dinâmica e adaptável, e está distribuída ao longo do desenvolvimento uma vez que ocorre em um ambiente onde o desenvolvimento completo e exigência de especificações são impossíveis ou mesmo inadequadas de serem elicitadas por completo [Paetsch *et al.*, 2003; Cao *et al.*, 2008].

A ERMT apresenta os pressupostos de que: i) o cliente é capaz de especificar todas as suas necessidades na fase inicial; ii) um ou mais interessados são responsáveis pela coleta de requisitos; iii) a equipe de desenvolvimento pode compreender as necessidades do cliente, e; iv) a estrutura das organizações é hierárquica havendo uma nítida separação das funções diferentes. Por outro lado tem-se os pressupostos relacionados à ERMA que são [Paetsch *et al.*, 2003; Turk *et al.*, 2005; Sillitti *et al.*, 2005; Ramesh *et al.*, 2010]:

- *Pressuposto da mudança nos requisitos:* requisitos constantemente evoluem por causa das mudanças na tecnologia, nas necessidades dos clientes e nos domínios de negócios. Portanto, o desenvolvimento ágil não emprega muito tempo em levantamento de requisitos inicial. Em vez disso, os requisitos emergem durante o ciclo de desenvolvimento;
- *Pressuposto sobre documentação:* desenvolver documentação e modelos extensos é improdutivo. Portanto, não se exige detalhes na elaboração de documentos no desenvolvimento ágil;
- *Pressuposto da interação com cliente:* os clientes estão disponíveis para interação frequente com os desenvolvedores, visto que o desenvolvimento ágil depende dessas interações para obter e validar requisitos, e;
- *Pressuposto do custo da mudança:* o custo de se fazer alterações no software não aumenta dramaticamente ao longo do tempo. Logo, a ERMA pode ser realizada de forma iterativa e incremental sem aumentar o custo do desenvolvimento.

No desenvolvimento ágil de software o levantamento de requisitos é feito através da ER iterativa e pela comunicação face-a-face com os clientes, enquanto o levantamento de requisitos tradicional visa descobrir todos os requisitos através das quatro atividades sequenciais no processo de ERMT: i) levantamento de requisitos, ii) análise de requisitos e negociação, iii) documentação de requisitos e iv) validação dos requisitos antes de iniciar o desenvolvimento. Os MA's reconhecem que os requisitos evoluem ao longo do tempo e que eles são descobertos em todo o ciclo de desenvolvimento, que a análise de requisitos e a negociação são facilitadas pela ER iterativa, que a comunicação face a face presencial e

que o planejamento e priorizações são constantes. Essas práticas de ER ágeis ajudam a refinar, mudar e priorizar as necessidades [Ramesh *et al.*, 2010 *apud* Kotonya e Sommerville, 1998; Ramesh *et al.*, 2010].

Nos MA's os requisitos geralmente não são formalmente documentados em um documento de especificação. Em vez disso, eles são documentados informalmente como listas de *features* ou histórias. A especificação detalhada dos requisitos passa a ser a intensa comunicação entre a equipe de desenvolvimento e o cliente. Finalmente, a validação de requisitos é feita através de reuniões presenciais de revisão. Na ER ágil, validação de requisitos é focada em determinar se os requisitos refletem as necessidades do usuário atual. A consistência e completude dos requisitos não são formalmente controladas e os documentos de requisitos formais raramente são desenvolvidos. Por fim, os requisitos em MA's são formalizados através de reuniões de revisão para validar os requisitos. Estas reuniões são frequentemente realizadas através de uma comunicação informal entre o cliente e os desenvolvedores. [Ramesh *et al.*, 2010].

Na Tabela 3 são comparados os MT's e os MA's para a realização de quatro atividades do ERMT.

Tabela 3 - Abordagem comparativa da ERMT e da ERMA [Ramesh *et al.*, 2010].

ER	ERMT	ERMA
Elicitação de requisitos	Descobrir boa parte dos requisitos iniciais	Iterativo: necessidades evoluem com o tempo e novos requisitos são descobertos em todo o ciclo de desenvolvimento
Análise e negociação de requisitos	Foco em resolver conflitos	Concentra-se no refinamento, mudança e priorização dos requisitos
Documentação de requisitos	Documentação formal contendo detalhamento dos requisitos	Sem documentação formal
Validação de requisitos	A consistência e completude do documento de requisitos	Concentre-se em verificar se os requisitos refletem as necessidades atuais do usuário

Para Sillitti e outros [Sillitti *et al.*, 2005] as abordagens tradicionais e ágeis diferem em sua abordagem para gerenciamento de incertezas e de levantamento de requisitos. O desenvolvimento ágil depende da interação com os requisitos do cliente e ocorre durante todo o ciclo de desenvolvimento. Em contraste, no desenvolvimento tradicional, levantamento de requisitos ocorre principalmente no início do projeto.

Os defensores dos MA's sugerem questão susceptíveis de atingir a melhor satisfação do cliente devido à constante interação com o cliente durante o desenvolvimento do software. No entanto, o caráter informal das práticas de ER ágeis pode ser considerado inaceitável por alguns autores [Paetsch *et al.*, 2003].

Embora as práticas da ERMA proporcionem benefícios, como uma melhor compreensão das necessidades dos clientes e a capacidade de adaptação às necessidades do ambientes dinâmicos atuais, elas se defrontam com vários desafios distintos, como por exemplo, a iteratividade da elicitação de requisitos que faz com que o escopo do projeto fique em constante mudança dificultando estimar custo e tempo para o projeto como um todo. As organizações de desenvolvimento devem comparar cuidadosamente os custos e benefícios das práticas de ERMA antes de as adotarem em seus ambientes de trabalho [Cao *et al.*, 2008].

Através da análise da ERMT e da ERMA descritas nesta seção foi possível estruturar a Tabela 4 e Tabela 5, nas quais foi, possível evoluir o trabalho proposto por Ramesh e outros [Ramesh *et al.*, 2010] com um aprofundamento do detalhamento focado nos métodos descritos.

Tabela 4 – Detalhamento do comparativo entre ERMT e ERMA.

RE	ERMT	ERMA
Elicitação de requisitos	• [Cascata, RAD] Através de consultas ao usuário são definidos os serviços, restrições e objetivos do software. Posteriormente na fase de manutenção há novamente um contato para que novos requisitos sejam identificados de modo ao software evoluir e permanecer útil. • [Incremental] O cliente identifica, em linhas gerais, os serviços a serem fornecidos pelo software. • [Prototipagem] O protótipo é desenvolvido com uma versão inicial do documento de especificação dos requisitos, com os requisitos vagamente entendidos pelas partes envolvidas. • [Espiral] Os objetivos específicos dessa fase do projeto são definidos. As restrições sobre o processo e o produto são identificadas e um plano detalhado de gerenciamento é elaborado. Os riscos de projeto são identificados.	• [XP] As histórias de usuário são coletadas através de técnicas utilizadas pelo XP que podem variar desde o “<i>Planning game</i>”, a entrevistas, “<i>brainstorming</i>” ou priorizações de requisitos. • [Scrum] Os requisitos no Scrum podem ser coletados de inúmeras formas, variando desde requisitos coletados por meio de Casos de Uso a Histórias de usuário. • [DSDM] Requisitos funcionais e não funcionais mais relevantes são levantados junto ao cliente; • [FDD] É desenvolvido um modelo que fornecem uma visão global dos requisitos. • [ASD] Seus estágios iniciais são curtos e objetivam garantias e envolvimento do cliente para a definição dos serviços, restrições e objetivos do software.
Análise e negociação de requisitos	• [Cascata, RAD] Na fase de Desenho de sistemas e software, os requisitos são divididos em requisitos de hardware ou software e são utilizados para elaboração do desenho. Os requisitos são novamente abordados na fase de Operação e Manutenção de modo a corrigir os erros não detectados nos estágios anteriores. • [Incremental] Os clientes priorizam os serviços são mais importantes e definem o número de incrementos que serão entregues a cada incremento. • [Prototipagem] Após a elaboração do protótipo o cliente o utiliza e o avalia repassando, aos desenvolvedores, as impressões do que precisa ser alterado, o que esta faltando e o que não é preciso. • [Espiral] Para cada risco de projeto identificado, uma análise detalhada é realizada.	• [XP] Os cartões de história são divididos em tarefas pela equipe, que estima o esforço e os recursos necessários para a implementação. Com base nestas tarefas os clientes priorizam as histórias para implementação que podem ser usadas de modo a proporcionar apoio útil ao negócio. • [Scrum] É feita a priorização e a inserção dos requisitos nos <i>backlogs</i>. • [DSDM] Equipe de desenvolvimento e clientes discutem quais requisitos são mais importantes. • [FDD] Os requisitos são refinados em <i>features</i> de modo a construir a Lista de <i>Features</i>. • [ASD] Durante as reuniões de revisão, um aplicativo de trabalho é explorado e os resultados das reuniões são documentados como requisições de mudança.

Tabela 5 – Detalhamento do comparativo entre ERMT e ERMA (Continuação).

RE	ERMT	ERMA
Documentação de requisitos	• [Cascata, RAD] Os requisitos são detalhados e armazenados em um documento de requisitos e servem como uma especificação do software. Posteriormente é desenvolvido o desenho do software, que contém a identificação e a descrição das abstrações fundamentais do software a ser desenvolvido. Em ambos os casos é exigido que os requisitos passem por fases de produção e aprovação. • [Incremental] Um documento de requisitos formal é proposto para cada um dos serviços propostos. • [Prototipagem] No final os requisitos iniciais são alterados para produzir a especificação formal dos requisitos. • [Espiral] Os requisitos são documentados por meio de planos formais.	• [XP] Os requisitos são registrados em cartões de histórias e as histórias a serem incluídas em uma entrega são determinadas pelo tempo disponível e pela prioridade. Os desenvolvedores dividem essas histórias em tarefas. Também é realizado um projeto suficiente para atender aos requisitos e nada mais. • [Scrum] Todos os requisitos considerados necessários ou úteis para o produto estão listados no <i>product backlog</i>. • [DSDM] Realizado por meio de protótipos; • [FDD] Todos os requisitos são documentados como requisições de mudança. A criação de um documento orientado para o cliente como um manual do usuário também é um recurso para documentação dos requisitos. • [ASD] A documentação surge de dos resultados das reuniões que são documentados como requisições de mudança.
Validação de requisitos	• [Cascata, RAD] São desenvolvidos e testados todos os requisitos levantados na fase de elicitação pela equipe do projeto. Após os testes, o software, contendo todos os requisitos implementados, é entregue. • [Incremental] Após a identificação dos incrementos do software, os requisitos dos serviços a serem entregues no primeiro incremento são definidos detalhadamente e ele é desenvolvido. Durante o desenvolvimento, pode ser utilizada a análise dos próximos requisitos para incrementos posteriores, mas não são aceitas mudanças de requisitos para o incremento atual. • [Prototipagem] O cliente avalia o protótipo verificando se ela atende a todos os requisitos estipulados e ao final autoriza a construção do produto final. • [Espiral] Após a avaliação de risco, um modelo de desenvolvimento para o software é selecionado.	• [XP] Para a validação dos cartões de história são definidos testes de aceitação pelo cliente. Estes testes são baseados em pequenas e frequentes entregas, assim, os clientes podem analisar e testar a funcionalidade e concepção do programa entregue, discutir questões que deseja alterar ou que sejam adicionadas na próxima versão. • [Scrum] No final de um <i>sprint</i> uma reunião de revisão do <i>sprint</i> é realizada demonstrando a nova funcionalidade para o cliente e solicitando comentários. • [DSDM] Uma variedade considerável de protótipos pode ser criada a fim de se avaliar a usabilidade do software em desenvolvimento, sua performance e sua capacidade. • [FDD] Ciclos de revisão em períodos curtos favorecem a compreensão dos detalhes e a validação do que foi planejado. • [ASD] A validação é feita através de ciclos de revisão em períodos curtos que favorecem a compreensão dos detalhes do que foi planejado.

2.5. Trabalhos Relacionados

Alguns trabalhos presentes na literatura estão relacionados aos temas abordados nesta dissertação e foram organizados na Tabela 6, onde as colunas representam os temas que são: *migração de MT's para MA's (Migração)*, *híbridos de metodologias que adotam*

princípios tanto dos MA's quanto dos MT's (Híbridos) e comparações entre ERMT e a ERMA (ER Comp.) e as linhas representam as citações dos trabalhos. Segue uma breve descrição dos trabalhos indexados por ordem cronológica para com cada um dos temas, que embora possuam enfoques diferentes eles abordam temas similares.

Tabela 6 – Trabalhos Relacionados.

Trabalho(s)/Tema	Migração	Híbridos	ER Comp.
[Boehm e Turner, 2005]	X		
[Nerur <i>et al.</i> , 2005]	X		
[Misra <i>et al.</i> , 2006]	X		
[Petersen e Wohlin, 2010]	X		
[Laanti <i>et al.</i> , 2011]	X		
[Donzelli e Iazeolla, 2001]		X	
[Qureshi e Hussain, 2008]		X	
[Paetsch <i>et al.</i> , 2003]			X
[Soundararajan e Arthur, 2009]			X
[Ramesh <i>et al.</i> , 2010]			X

Boehm e Turner apresentam, em seu trabalho [Boehm e Turner, 2005], os resultados da implantação, realizada por um grupo de pesquisadores e profissionais, de processos ágeis em grandes projetos de engenharia de softwares baseados em processos que dependem de desenvolvimento tradicional e artefatos. Os resultados foram um conjunto de desafios relacionados à mudança e uma lista de cerca de 40 barreiras percebidas para a implementação de práticas de MA's, estas barreiras foram categorizadas como problemas apenas em termos de escopo ou escala, ou como importantes questões gerais que precisam de resolução. A partir dessas duas categorias, foram identificadas três áreas: conflitos de processo de desenvolvimento; conflitos de processos de negócios, e; conflitos entre pessoas. Esses foram identificados com os desafios essenciais para gerentes de software de grandes organizações em trazer abordagens ágeis para seus projetos. Por fim, os autores discutem várias abordagens para enfrentá-los.

Nerur e outros apresentam em seu trabalho [Nerur *et al.*, 2005], uma comparação entre os MT's e os MA's apontando as vantagens e desvantagens de cada categoria de método. Nessa comparação os autores focam na análise dos tópicos a respeito das metodologias: quais seus pressupostos fundamentais, onde está centrado o controle, como é feita a gestão do projeto e a do conhecimento, como são feitas as atribuições de função, como é feita a comunicação, qual é o papel do cliente, como é gerido o ciclo do projeto, quais os modelos de desenvolvimento adotados, como é a estrutura organizacional e quais os padrões de tecnologia adotados. Também são identificadas as chaves de gestão, organizacional, pessoas, processos e questões tecnológicas na adoção de MA's.

Misra e outros oferecem, em seu trabalho [Misra *et al.*, 2006], uma visão consolidada das importantes mudanças necessárias e dos desafios envolvidos em projetos que estão em migração de práticas de MT's para MA's. Também é fornecido um quadro conceitual com o intuito de ajudar gestores de projeto a focar nas mudanças necessárias e nos desafios envolvidos dos projetos de desenvolvimento ágil de software.

Petersen e Wohlin apresentam, em seu trabalho [Petersen e Wohlin, 2010], uma investigação dos pontos de estrangulamento causados pelos trabalhos desnecessários e retrabalhos ocasionados por alterações ao se migrar de um MT para uma método incremental com práticas ágeis. O contexto do trabalho está com foco orientado para o mercado e no desenvolvimento em grande escala. Para isso, foi realizado um estudo de caso em um local de desenvolvimento da Ericsson, na Suécia no final de 2007, no qual foram realizadas 33 entrevistas. Foram identificadas 64 questões, sendo 24 de caráter geral e os 40 restantes de caráter locais. A maioria das questões diz respeito aos MT's. Também foram identificadas questões em relação os métodos ágeis, que estavam relacionadas com a condução e cobertura dos testes, versionamento do software, e a sobrecarga de coordenação. Com a adoção de práticas de MA's as questões encontradas com os MT's não mais aconteceram, permitindo compreender que as práticas ágeis agregam valor em MT's, o que fica evidente na solução de problemas críticos encontrados no desenvolvimento com MT's.

Laanti e outros fornecem evidências de uma transformação em grande escala ágil dentro da Nokia em seu estudo [Laanti *et al.*, 2011], na qual foram aplicados questionários, para mais de 1000 entrevistados, em sete países na Europa, América do Norte e Ásia, com o objetivo de coletar evidências concretas sobre a percepção do impacto da migração para um MA quando esta é implantada em um ambiente muito complexo de desenvolvimento de software. Os resultados revelaram que a maioria dos indivíduos que responderam o questionário concordam com todos os benefícios geralmente ditos de MAs, incluindo maior satisfação, um sentimento de eficácia, o aumento da qualidade e da transparência, aumento da autonomia e felicidade, e detecção precoce de defeitos. Constatou-se também que 60% dos entrevistados não gostaria de voltar à velha forma de trabalhar.

O trabalho de Donzelli e Iazeolla [Donzelli e Iazeolla, 2001] propõe uma combinação de três métodos de modelagem tradicional (analítico, contínuo e eventos discretos) dentro de uma abordagem híbrida de modelagem única em dois níveis, para resolver questões de modelagem de simulação de processo de software. Na abstração de alto nível, o processo é modelado como uma fila de eventos discretos que representa suas

atividades componentes, suas interações e artefatos modificados. Na abstração de baixo nível, são usados os métodos analíticos e contínuos para descrever o comportamento das atividades iniciais. A abordagem híbrida é aplicada em um processo de software cascata para que se possa estudar os efeitos da instabilidade dos requisitos em vários processos de qualidade de atributos, esforço, prazo de entrega, produtividade, porcentagem de retrabalho e qualidade do produto. Resultados da simulação mostram que o uso do modelo pode prover tanto sugestões qualitativas e quantitativas de melhoria para o processo de software como satisfação de necessidades específicas organizacionais. Contudo, o modelo foi inicialmente concebido para representar o comportamento hipotético de projetos e permitir aos pesquisadores visualizar as implicações de suas pressuposições, podendo estendê-las e customizá-las tornando-as ferramentas para analisar e prever o comportamento atual dos projetos.

Qureshi e Hussain apresentam, em seu trabalho [Qureshi e Hussain, 2008], discussões sobre as vantagens dos MA's em comparação com MT's reforçando que os MA's tem ganhado grande popularidade junto à comunidade de desenvolvimento de software nos últimos 5 anos. Também discorrem sobre os inconvenientes dos MA's, tais como documentação pobre e de baixa qualidade. No entanto, o foco do trabalho está evolução do modelo de processo XP e na proposição um modo de processo adaptativo baseado nestas evoluções.

Paetsch e outros fazem comparações entre as abordagens tradicionais e as ágeis de ER em seu trabalho [Paetsch *et al.*, 2003], realizando uma análise de semelhanças e diferenças das duas abordagens e também determina maneiras possíveis de desenvolvimento ágil de software pode se beneficiar de métodos de ER.

Soundararajan e Arthur elaboraram, em seu trabalho [Soundararajan e Arthur, 2009], um quadro combinando os princípios de desenvolvimento de software ágil e desenvolvimento tradicional que aborda o problema da rápida mudança de requisitos para softwares de larga escala. O quadro abordado é composto por duas partes: (1) uma abordagem de recolhimento de requisitos no desenvolvimento tradicional que reflete a filosofia ágil, o modelo de geração de requisitos ágeis e (2) um processo de desenvolvimento adaptado que pode ser aplicado a qualquer software de pequena escala ou maior.

Ramesh e outros descrevem uma pesquisa empírica [Ramesh *et al.*, 2010] em ER de metodologias ágeis. Com base em uma análise dos dados coletados em 16 organizações de desenvolvimento de software dos EUA, onde foi possível identificar seis práticas ágeis e

sete desafios que são criados pelo uso dessas práticas. Também foi analisada de que forma esse conjunto de práticas podem atenuar ou agravar riscos na ER. Com isso, o trabalho visa auxiliar os desenvolvedores sobre a necessidade de avaliar cuidadosamente os fatores de risco no seu ambiente de projeto de modo a ponderar se os benefícios das práticas ágeis de ER compensam os custos impostos pelos desafios da adoção das mesmas.

2.6. Considerações finais

Neste capítulo, foram realizadas descrições sobre os modelos e métodos de desenvolvimento de software com o foco em seus processos e na forma com que eles realizam a gerência de seus requisitos. Apresentou-se uma visão comparativa sobre a ERMT e a ERMA. Por fim, foram apresentados os principais trabalhos relacionados à dissertação.

Capítulo 3

Processo de Engenharia de Requisitos Adaptável e Flexível (PERAF)

3.1. Considerações iniciais

Nesse capítulo, é esboçado um processo de Engenharia de Requisitos (ER) elaborado com o objetivo de (i) minimizar o impacto no processo de ER e na cultura organizacional de equipes de desenvolvimento de software que adotam um Método Tradicional (MT) e querem migrar para um Método Ágil (MA) e (ii) ser utilizado por organizações desenvolvedoras de software que ainda não têm um processo de ER definido e formalizado. Esse processo foi denominado Processo de Engenharia de Requisitos Adaptável e Flexível (PERAF).

O processo foi esboçado sob a luz do décimo princípio do manifesto ágil [Manifesto Ágil, 2001] (*Simplicidade: a arte de maximizar a quantidade de trabalho que não precisou ser feito*) e objetiva deixar livre para a equipe de desenvolvimento decidir quais artefatos ou estratégias adotar. Assim, foi adotada uma estratégia de substituir os artefatos “fixos” do processo tradicional por “assuntos” a serem abordados. Como exemplo, a geração dos artefatos *Documento de Visão* e *Documento de Escopo* da Atividade *Elicitação* de um desenvolvimento comumente tradicional, foram substituídos pelo conceito “Documentos adotados para o entendimento inicial da aplicação”. Assim, o PERAF se torna adaptável às necessidades de documentos específicos de diferentes tipos de projetos e métodos ao mesmo tempo em que mantém as necessidades traduzidas pelo conceito dos artefatos.

O esboço do Processo de Engenharia de Requisitos Adaptável e Flexível (PERAF) é explicado na Seção 3.2. As disciplinas, as atividades e as tarefas são apresentadas na Seção 3.3 e, por fim, na Seção 3.4 são apresentadas as considerações finais a certa do capítulo.

3.2. Elaboração do PERAF

O esboço do PERAF consistiu em realizar cinco etapas, representadas no mapa conceitual⁶ da elaboração do PERAF (Figura 7) e descritas nos itens:

- **Primeira etapa:** foi realizado um levantamento bibliográfico em busca da existência de um processo de ER detalhado, formalizado e genérico, quando à metodologia de desenvolvimento de software, para ser utilizado em organizações desenvolvedoras de software. Esse levantamento foi feito em bibliotecas digitais, artigos científicos em eventos e periódicos nacionais e internacionais, e em dissertações e teses em diversos programas de pós-graduação do Brasil e do exterior. Alguns processos foram encontrados, mas suas características inviabilizavam o foco da busca (detalhado, formalizado e genérico quando à metodologia de desenvolvimento de software);

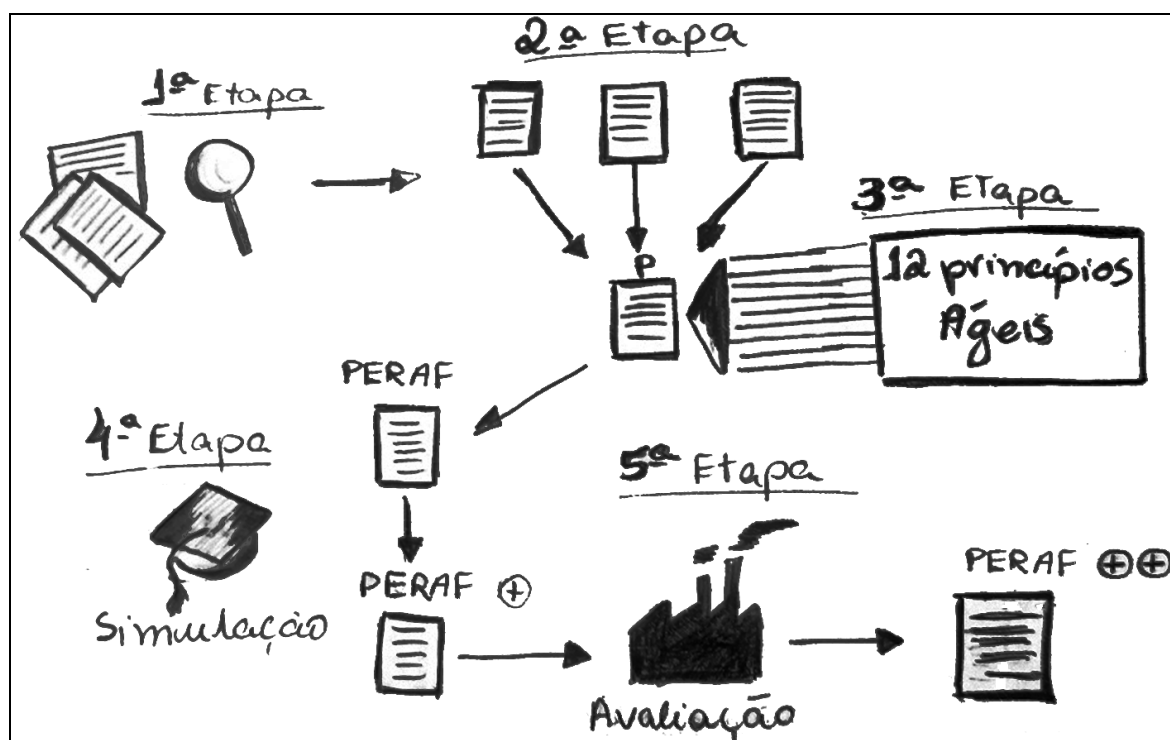


Figura 7 - Mapa Conceitual da Elaboração do PERAF.

- **Segunda etapa:** com o foco da busca não contemplado, recorreu-se a três fontes clássicas da área de Engenharia de Software os livros de Pressman [Pressman, 2010], Sommerville [Sommerville, 2010] e Wiegers [Wiegers, 2003]. Com base

⁶ Mapa conceitual: é uma representação abstrata dos recursos, ideias e pensamentos necessários para realizar uma tarefa cujo principal objetivo é gerar uma ideia do todo. Para a elaboração do mapa conceitual, não se deve preocupar com estilo, mas com a criatividade [Ashby, 2000].

nessas fontes, que apresentam indicações para realizar a ER de maneira tradicional, foi possível modelar formalmente um processo de ER;

- **Terceira etapa:** a partir de um processo de ER de maneira tradicional formalmente modelado e analisando os princípios ágeis presentes no Manifesto Ágil [Manifesto Ágil, 2001], foi possível identificar que seis dos doze princípios poderiam ser incorporados ao processo, o que alteraria sua estrutura. O processo foi alterado e o resultado foi o PERAF (Processo de Engenharia de Requisitos Adaptável e Flexível). Os princípios ágeis considerados e as alterações realizadas no processo foram:

- Segundo princípio (*Aceitar mudanças de requisitos, mesmo que tarde no desenvolvimento. (Processos ágeis aproveitam as mudanças para que o cliente possa tirar vantagens competitivas)*). Foram propostas as Atividades *Controle de Alteração* e *Rastreabilidade de Requisitos*, as quais tem como objetivo possibilitar mudanças constantes e garantir que as mudanças sejam propagadas pelos demais requisitos e elementos do software com a realização da rastreabilidade dos requisitos e dos demais elementos afetados durante a alteração;
- Quinto princípio (*Construir projetos ao redor de indivíduos motivados, proporcionando a eles ambiente e suporte necessários, e confiar que farão seu trabalho*). Foi proposta a Tarefa *Definir Contato para Elicitação* na Atividade *Elicitação*. O seu objetivo é definir quais são os representantes dos clientes durante o processo de ER e, se necessário, definir os líderes e os pontos de contato, de modo a ter um projeto construído em torno de indivíduos motivados;
- Sexto princípio (*A maneira mais eficiente e eficaz de transmitir informações para e dentro de um time de desenvolvimento é com conversas face a face*). Foi reforçada a importância de ter contato direto com o cliente em várias atividades do processo, o que permite detectar a presença desse princípio e atuação do cliente nas Tarefas *Garantir o Entendimento Geral aos Envolvidos*, *Definir Necessidades de Usuário e dos Sistema/Software*, *Levantar Necessidades Faltantes*, *Garantir o Entendimento dos Requisitos*, *Priorizar os Requisitos*, *Revisar Documentos e Diagramas Adotados*, *Gerar Critérios de Aceitação* e *Tomar de Decisão de Alteração*;

- Sétimo princípio (*Software funcional é a principal medida de progresso*). Foi criada a Atividade *Rastreabilidade/Manutenção do Estado dos Requisitos*. O seu objetivo é contribuir para a manutenção do estado de completude dos requisitos do software e servir como medidor de progresso baseado na quantidade de software funcionando;
- Oitavo princípio (*Processos ágeis promovem um ambiente sustentável. Patrocinadores, desenvolvedores e usuários devem ser capazes de manter indefinidamente um ritmo constante*). Foi criada a Disciplina *Gerenciamento de Requisitos*. O seu objetivo é permitir a evolução sustentável do software em desenvolvimento e buscar junto aos desenvolvedores garantias de manter a evolução do software de forma consistente até o término de seu desenvolvimento;
- Décimo princípio (*Simplicidade - a arte de maximizar a quantidade de trabalho que não precisou ser feito - é essencial*). Foram propostas junções de tarefas específicas para o desenvolvimento tradicional em tarefas que generalizam o que deveria ser feito. Dessa forma, a equipe de desenvolvimento do projeto tem flexibilidade para decidir como realizar as tarefas. As tarefas modificadas, resultantes e descartadas são apresentadas na Tabela 7;

Tabela 7 - Tarefas Modificadas.

Tarefa Processo Tradicional (Modificadas)	Tarefa Processo PERAF (Resultantes)
- Escrever Documento de Visão - Escrever Documento de Escopo	- Garantir o Entendimento Geral aos Envolvidos
- Definir Casos de Uso	- Definir Necessidades do Usuário e do Sistema/Software
- Desenhar Diagrama de Contexto - Gerar Protótipo - Criar Dicionário de Dados	- Garantir o Entendimento dos Requisitos
- Refinar Documento de Requisitos - Construir Diagrama de Casos de Uso - Definir Gabarito para Detalhamento dos Casos de Uso - Descrever Casos de Uso	- Refinar Documentos Utilizados - Refinar Diagramas Utilizados - Refinar Requisitos
- Revisar o Documento de Requisitos - Revisar Casos de Uso	- Revisar Documentos e Diagramas Adotados
- Adotar Práticas de Medida Preventiva - Mensurar a Volatilidade dos Requisitos - Tomar Medidas Preventivas	- Foram descartadas, pois, são contrárias ao segundo princípio que apregoa que mudanças são sempre bem vindas.

- Os princípios 1 (*A maior prioridade é satisfazer o cliente, por meio de entrega breve e contínua de um “software de valor”*), 3 (*Entregar software funcionando com frequência, menos de um ou dois meses, com preferência para os períodos mais curtos*), 4 (*Pessoas relacionadas a negócios e desenvolvedores devem trabalhar em conjunto e diariamente, durante o curso do projeto*), 9 (*Contínua atenção à excelência técnica e bom desenho (design) melhora a agilidade*), 11 (*As melhores arquiteturas, requisitos e desenhos emergem de times auto organizáveis*) e 12 (*Em intervalos regulares, o time reflete sobre como se tornar mais efetivo e ajustam e otimizam seu comportamento*); não foram incorporados e proposições não foram feitas, pois são dependentes da metodologia de desenvolvimento e da forma de trabalho adotadas pela equipe para o projeto.
- **Quarta etapa:** foram realizadas revisões e simulações (aplicação do PERAF em projetos não reais) com o PERAF, as quais permitiram identificar a necessidade de adequação dos artefatos de software de entrada e saída das tarefas contidas no processo, e;
 - **Quinta etapa:** foi feita a avaliação do PERAF em dois estudos de caso reais (projetos reais de desenvolvimento de software) com equipes com características distintas e contextos organizacionais diferentes que estão detalhados no Capítulo 4.

3.3. Descrição do PERAF

O PERAF procura definir um processo de engenharia de requisitos adaptável e flexível esboçado a partir da adaptação de indicações para realizar a engenharia de requisitos dos métodos tradicionais combinado com características de agilidade. O PERAF está estruturado em duas disciplinas, sendo uma condicionada à existência de um conjunto inicial de requisitos (Figura 8). Caso esse conjunto não exista, o fluxo do processo segue para a Disciplina *Desenvolvimento de Requisitos*, responsável pela geração desse conjunto inicial de requisitos; caso o conjunto de requisitos exista ou seja requisitada alguma alteração nesse conjunto, o fluxo do processo segue para a Disciplina *Gerenciamento de Requisitos*, responsável por qualquer alteração realizada no conjunto inicial de requisitos.

O fluxo do processo permanece na Disciplina *Gerenciamento de Requisitos* até que o *Marco Projeto Finalizado* seja atingido e o projeto termine.

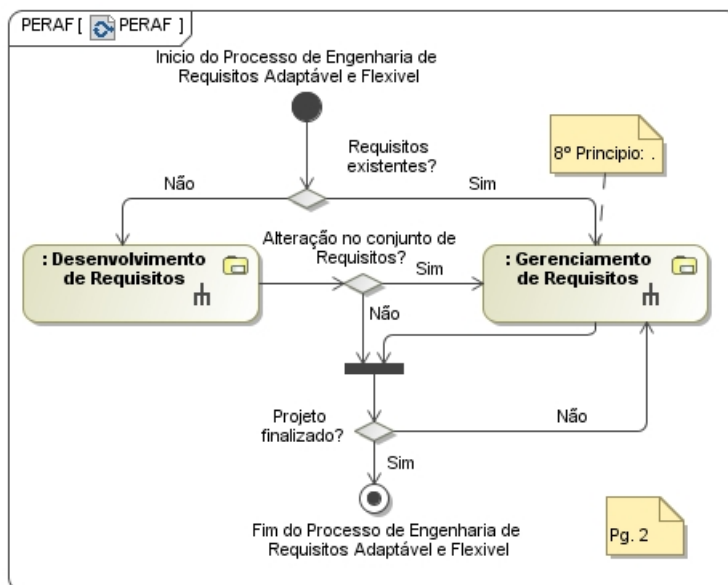


Figura 8 - Diagrama de Fluxo de Trabalho do PERAF.

O PERAF se adéqua a boa parte das necessidades dos MAs, no entanto, ele não deve ser mais importante que a interação com pessoas, devendo ser aplicado de acordo com as necessidades gerencias da empresa. O PERAF leva as equipes tradicionais a terem alguns indícios de agilidade através das características ágeis incorporadas.

No PERAF as tarefas podem ser suprimidas de acordo com as necessidades do projeto, ou seja, elas não precisam ser realizadas caso sejam desnecessárias para a execução do projeto. Essa supressão de atividades, assim como a redução dos documentos adotados será gradativa à medida que as organizações que se encontram em processo de migração forem se distanciando dos MT's e se aproximando dos MA's, este distanciamento está relacionada à diminuição dos documentos necessários para a execução do projeto e com a adoção dos paradigmas do MA adotado.

3.3.1. Disciplina Desenvolvimento de Requisitos

A Disciplina *Desenvolvimento de Requisitos* envolve as atividades *Elicitação*, *Análise*, *Especificação* e *Validação* (Figura 9). O objetivo dessa disciplina é levantar um conjunto inicial de requisitos para o software a ser desenvolvido. Essa não é uma

“disciplina atômica”⁷, ou seja, caso haja alteração no conjunto de requisitos elicitados, após a realização de uma de suas atividades, pode ocorrer desvio do fluxo para a Disciplina *Gerenciamento de Requisitos* de modo a tratar mudanças no conjunto de requisitos elicitados. Outro fluxo alternativo pode ocorrer após as Atividades *Especificação* e *Validação* quando existem requisitos viáveis não priorizados, especificados ou validados. Para esse caso, o fluxo retorna para a Atividade *Análise*. Por outro lado, caso não haja alterações no conjunto de requisitos, o fluxo segue naturalmente. Espera-se, ao término dessa disciplina, que um conjunto inicial de requisitos esteja especificado e validado.

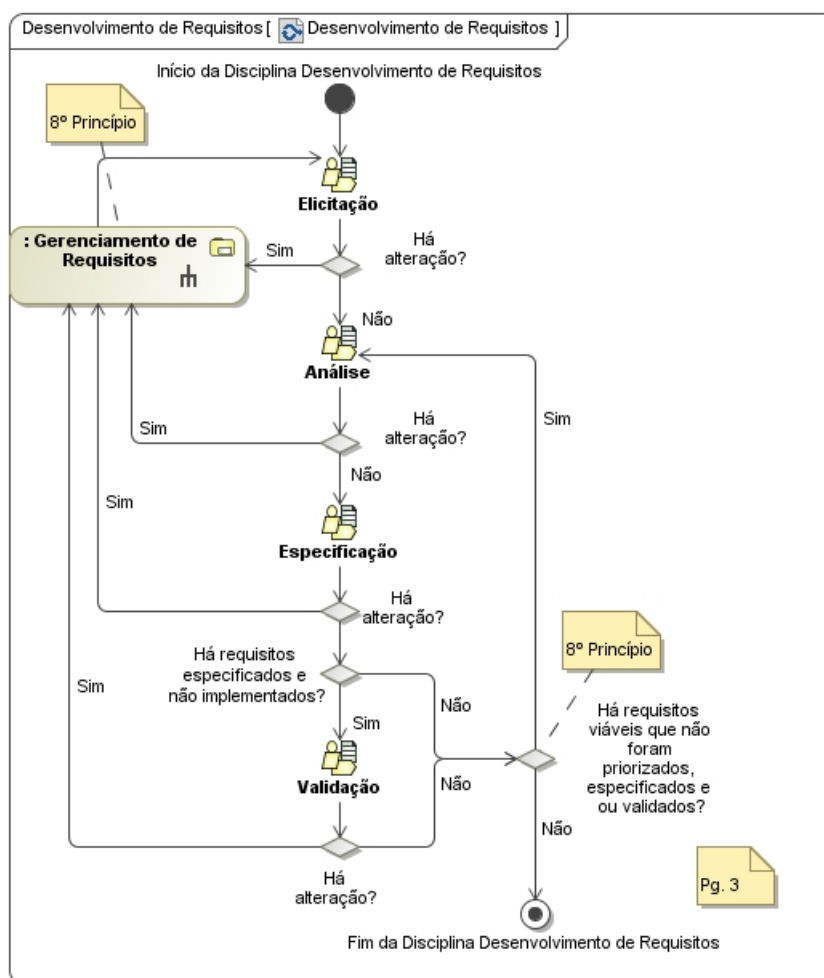


Figura 9 - Diagrama de Fluxo de Trabalho da Disciplina *Desenvolvimento de Requisitos*.

3.3.1.1. Atividade *Elicitação*

Na Atividade *Elicitação* são realizadas tarefas relacionadas ao contato inicial com o cliente, com o planejamento sobre como será realizada a ER e com a definição dos artefatos a serem utilizados no processo de ER. Ao final dessa atividade, espera-se ter um conjunto de requisitos inicial suficiente para gerar uma visão geral do software. Essa

⁷ Disciplina atômica: disciplina que ocorre sem desvio do fluxo principal de atividades.

atividade possui o fluxo de Tarefas *Planejar a Elicitação*, *Garantir Entendimento Geral aos Envolvidos*, *Definir Contato para Suporte no Desenvolvimento de Requisitos*, *Definir Necessidades do Usuário e do Sistema/Software*, *Levantar Necessidades Faltantes*, *Verificar a Possibilidade de Reúso dos Requisitos* e *Reutilizar Requisitos entre Projetos*. Uma exceção no fluxo pode ocorrer, quando forem adicionados novos requisitos após a realização da Tarefa *Levantar Necessidades Faltantes*. Neste caso, o fluxo retorna para a Tarefa *Garantir Entendimento Geral aos Envolvidos*. Outra exceção ocorre quando há a possibilidade de reutilizar requisitos; nesse caso, o fluxo do processo não realiza a Tarefa *Reutilizar Requisitos entre Projeto*. O fluxo de tarefas dessa atividade é representado no Diagrama de Fluxo de Trabalho da Atividade *Elicitação* (Figura 10).

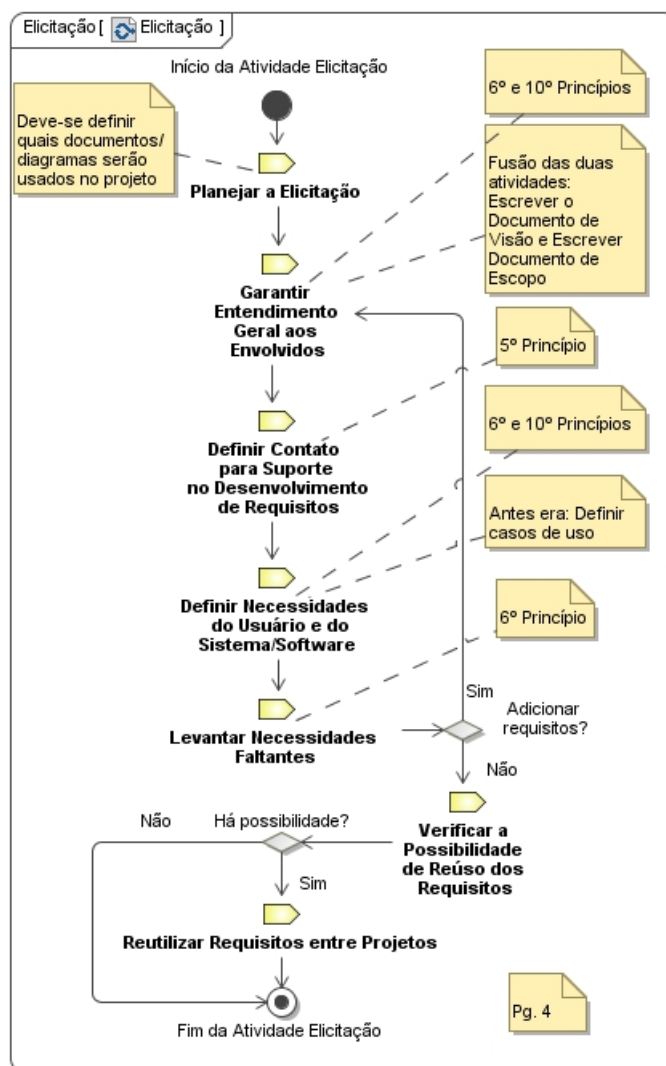


Figura 10 - Diagrama de Fluxo de Trabalho da Atividade *Elicitação*.

No PERAF os requisitos são levantados de forma iterativa e incremental, e interpretados e elicitados de diferentes maneiras que variam de acordo com o método de desenvolvimento de software adotado. A ideia do PERAF é apresentar um conjunto de

atividades que suportem minimamente as necessidades de elicitação de requisitos de cada método.

A descrição das tarefas da Atividade *Elicitação* é apresentada na Tabela 8 e a descrição das interações entre atores, artefatos e ferramentas é apresentada na Figura 11.

Tabela 8 - Descrição das Tarefas da Atividade *Elicitação*.

ATIVIDADE ELICITAÇÃO	
Tarefas	Descrição
<i>Planejar a Elicitação</i>	Deve-se definir como será feita a elicitação de requisitos, selecionar técnicas de elicitar e padrões para documentação dos requisitos e definir os <i>gabaritos</i> dos documentos a serem utilizados no Projeto. A lista de <i>gabaritos</i> gerados é: i) Lista de Documentos e Diagramas a serem utilizados no projeto; ii) Documentos e Diagramas a serem utilizados no projeto, e; iii) Lista de Contatos para o Desenvolvimento de Requisitos.
<i>Garantir Entendimento Geral aos Envolvidos</i>	Deve-se gerar documentos que garantam entendimento inicial da aplicação de forma a certificar que os envolvidos no projeto tenham conhecimento sobre os objetivos do software e sobre o que cada <i>release</i> do software conterá.
<i>Definir Contato para Suporte no Desenvolvimento de Requisitos</i>	Definir com os clientes quais serão os fornecedores de requisitos do software. Se for necessário, estabelecer grupos e líderes dentre eles para evitar requisitos redundantes.
<i>Definir Necessidades do Usuário e do Sistema/Software</i>	Deve-se documentar as necessidades requeridas pelos usuários para que o software ofereça suporte na realização de suas atividades e listar os eventos externos do software e suas respectivas respostas.
<i>Levantar Necessidades Faltantes</i>	Deve-se avaliar o conjunto de requisitos elicitados nos documentos gerados e detectar, a partir deles, requisitos faltantes de modo a completar possíveis falhas no conjunto de requisitos e sugerir soluções aos problemas encontrados pelos usuários nas versões disponibilizadas. Não é necessário realizar o versionamento, pois a Atividade <i>Elicitação</i> é responsável por garantir um entendimento inicial das necessidades dos usuários e sistema/software e está sujeita a constantes alterações.
<i>Verificar a Possibilidade de Reúso dos Requisitos</i>	Verificar nos documentos de requisitos de outros projetos se há funções similares.
<i>Reutilizar Requisitos entre Projetos</i>	Pode-se optar por reutilizar e adaptar um componente padrão ou desenvolvido pela empresa.

3.3.1.2. Atividade *Análise*

Na Atividade *Análise* são realizadas tarefas relacionadas à avaliação e análise da viabilidade de implementação do conjunto de requisitos e à geração de protótipos para melhor entendimento sobre os requisitos e à priorização dos requisitos. Essa atividade possui o fluxo de Tarefas *Garantir o Entendimento dos Requisitos*, *Analisar a Viabilidade dos Requisitos* e *Priorizar os Requisitos*, representado no Diagrama de Fluxo de Trabalho da Atividade *Análise* (Figura 12).

A descrição das tarefas da Atividade *Análise* é apresentada na Tabela 9 e a descrição das interações entre atores, artefatos e ferramentas é apresentada na Figura 13.

Tabela 9 - Descrição das Tarefas da Atividade *Análise*.

ATIVIDADE ANÁLISE	
Tarefa	Descrição
Garantir o Entendimento dos Requisitos	Deve-se adotar documentos, diagramas e/ou artefatos que garantam o entendimento sobre o software, podendo ser diagramas mostrando os limites do software ou protótipos ou dicionários de dados.
Analisar a Viabilidade dos Requisitos	Deve-se analisar a viabilidade de implementação dos requisitos, considerando fatores como: risco, custo e conflitos com outros requisitos.
Priorizar os Requisitos	Deve-se realizar uma análise dos requisitos de modo a priorizá-los. É necessário definir quais requisitos estarão em cada <i>release</i> do software.

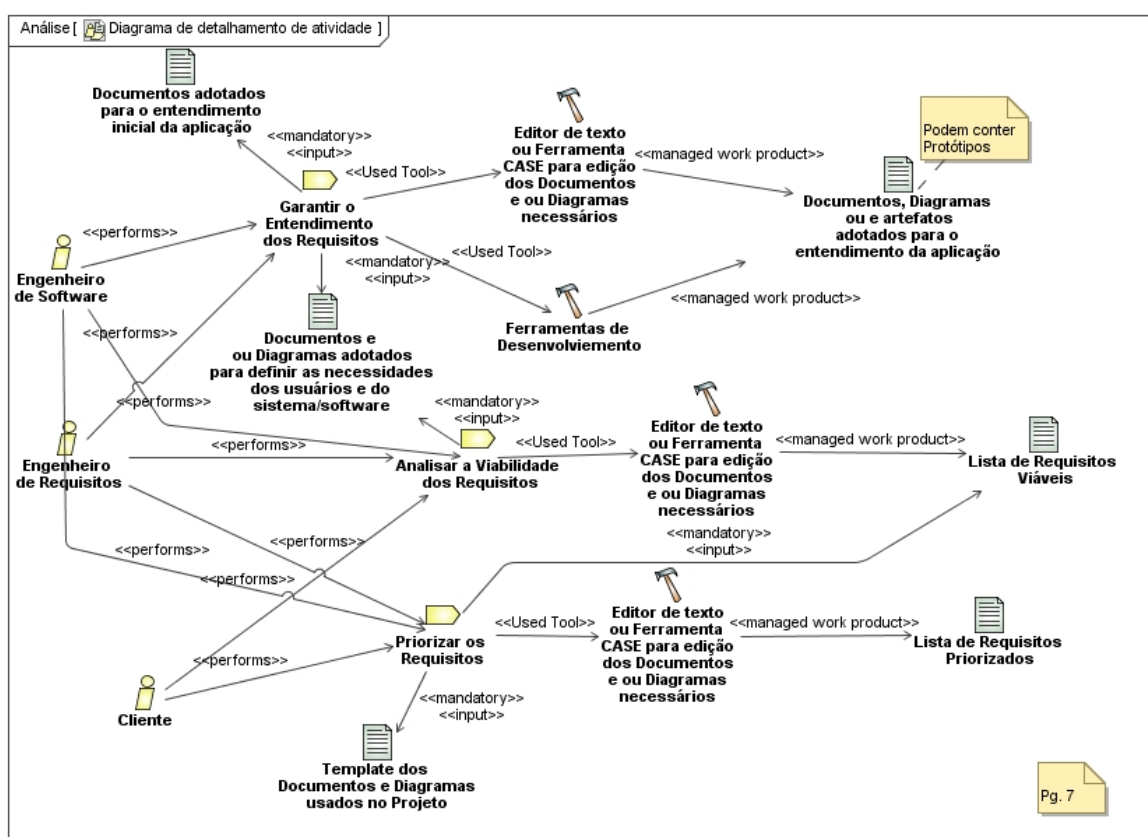


Figura 13 - Diagrama de detalhamento de atividade da Atividade *Análise*.

3.3.1.3. Atividade Especificação

Na Atividade *Especificação* são realizadas tarefas relacionadas ao refinamento, à padronização de documentação e ao melhor detalhamento dos requisitos elicitados. Essa atividade possui o fluxo de Tarefas *Refinar Documentos Utilizados*, *Refinar Diagramas Utilizados* e *Refinar Requisitos*, realizadas em paralelo e representadas no Diagrama de Fluxo de Trabalho da Atividade *Especificação* (Figura 14). A descrição das tarefas da Atividade *Especificação* é apresentada na Tabela 10 e a descrição das interações entre atores, artefatos e ferramentas é apresentada na Figura 15.

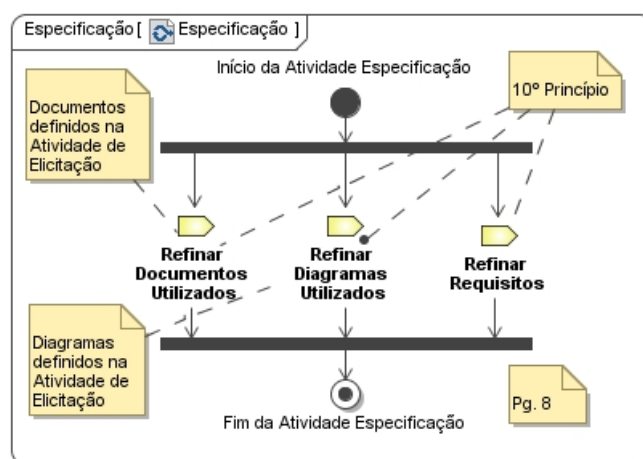


Figura 14 - Diagrama de Fluxo de Trabalho da Atividade *Especificação*.

Tabela 10 - Descrição das Tarefas da Atividade *Especificação*.

ATIVIDADE ESPECIFICAÇÃO	
Tarefas	Descrição
<i>Refinar Documentos Utilizados</i>	Deve-se refinar os documentos utilizados para descrever os requisitos de forma a definir padrões de documentação e de detalhamento exigidos pelo projeto. Um ponto importante são os <i>gabaritos</i> documentarem as origens dos requisitos de modo a possibilitar a rastreabilidade entre o requisito e seu autor. Além disso, é necessário definir um identificador único para os documentos de modo a garantir a sua unicidade.
<i>Refinar Diagramas Utilizados</i>	Deve-se refinar os diagramas utilizados para descrever os requisitos de forma a definir padrões de documentação e de detalhamento exigidos pelo projeto. Além disso, é necessário definir um identificador único para os diagramas de modo a garantir a sua unicidade.
<i>Refinar Requisitos</i>	Deve-se refinar os padrões utilizados para descrever os requisitos de forma a definir padrões de documentação e de detalhamento exigidos pelo projeto. Além disso, é necessário definir um identificador único para os requisitos.

3.3.1.4. Atividade *Validação*

Na Atividade *Validação* são realizadas tarefas relacionadas à revisão do conjunto de requisitos, à definição de critérios de aceitação e à geração de casos de teste. Essa atividade possui o fluxo de Tarefas *Revisar Documentos e Diagramas Adotados*, *Definir Template para Casos de Teste e de Aceitação*, *Gerar Casos de Teste* e *Gerar Critérios de Aceitação*, representado no Diagrama de Fluxo de Trabalho da Atividade *Validação* (Figura 16).

A descrição das tarefas da Atividade *Validação* é apresentada na Tabela 11 e a descrição das interações entre atores, artefatos e ferramentas é apresentada na Figura 17.

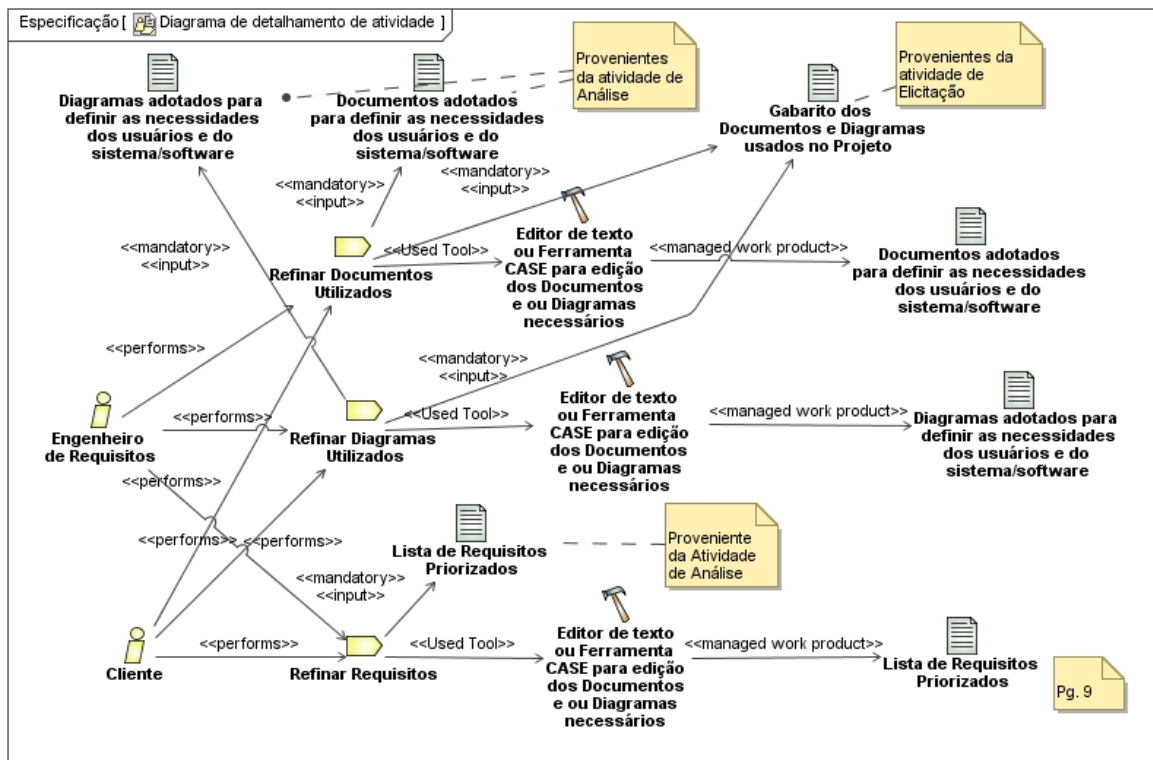


Figura 15 - Diagrama de detalhamento de atividade da Atividade Especificação.

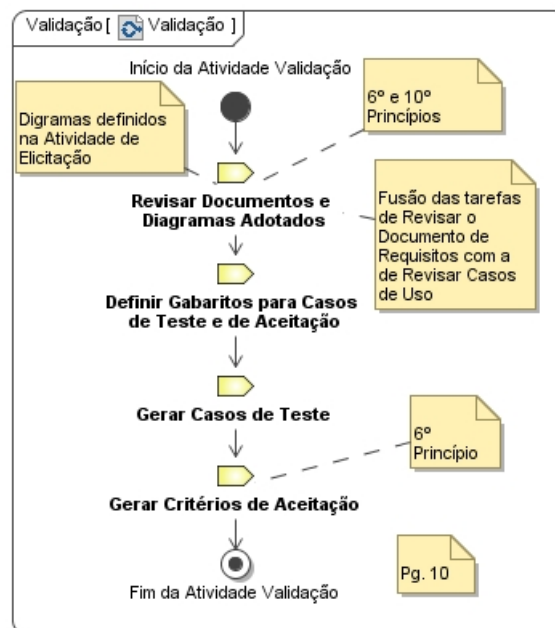
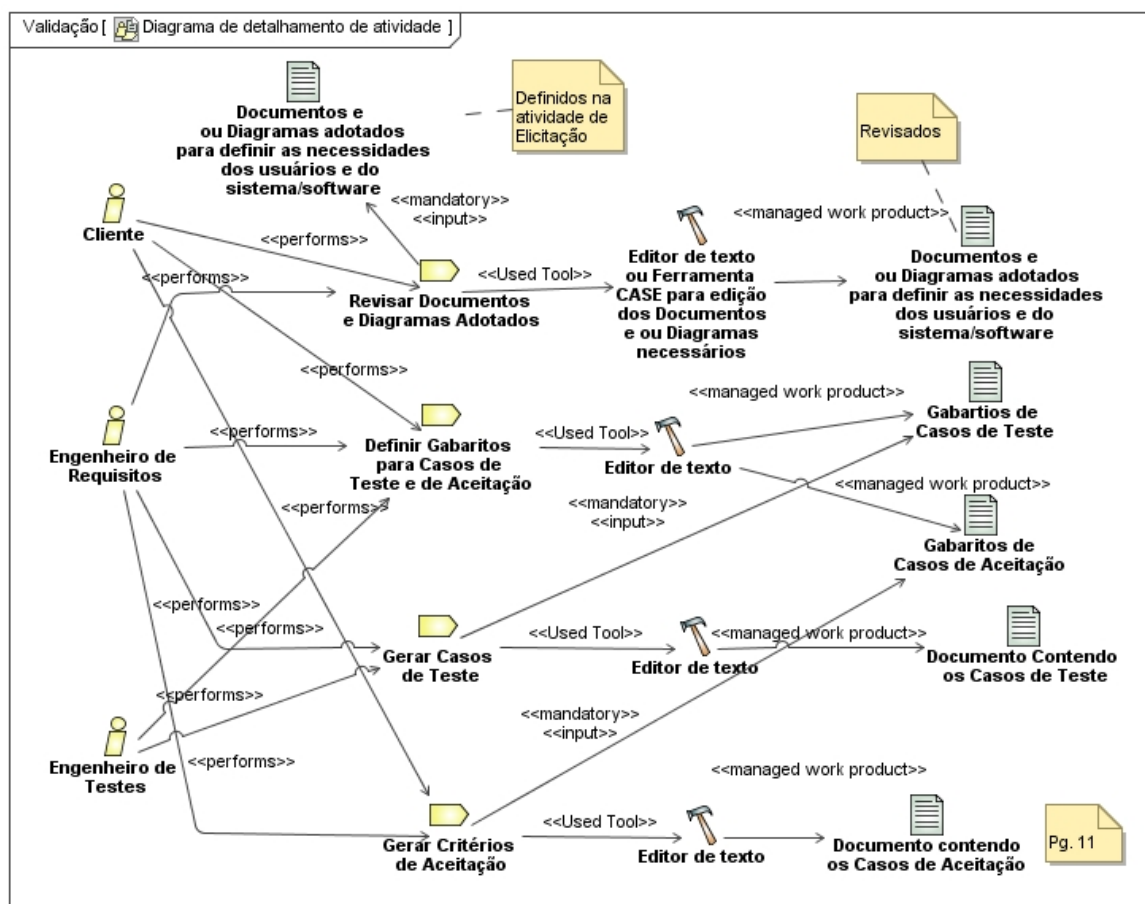


Figura 16 - Diagrama de Fluxo de Trabalho da Atividade Validação.

Tabela 11 - Descrição das Tarefas da Atividade *Validação*.

ATIVIDADE VALIDAÇÃO	
Tarefas	Descrição
Revisar Documentos e Diagramas Adotados	Deve-se revisar documentos e diagramas para garantir o menor número de defeitos possíveis e os interesses das partes envolvidas no negócio.
Definir Gabaritos de Casos de Teste e de Aceitação	Deve-se definir <i>gabaritos</i> para os casos de teste e para os critérios de aceitação.
Gerar Casos de Teste	Deve-se definir casos de testes dos requisitos para garantir que os objetivos do cliente sejam refletidos no complemento do software.
Gerar Critérios de Aceitação	Deve-se definir com os usuários os critérios de aceitação do software, ou seja, quando o software atenderá às necessidades e estará pronto para ser entregue.

Figura 17 - Diagrama de detalhamento de atividade da Atividade *Validação*.

3.3.2. Disciplina Gerenciamento de Requisitos

A Disciplina *Gerenciamento de Requisitos* reúne as atividades necessárias para assegurar que o conjunto de requisitos elicitados possa evoluir de maneira consistente e ser adaptável a mudanças. Essa disciplina possui o fluxo de Atividades *Controle de Alteração* e em paralelo ocorrem as atividades *Rastreabilidade e Manutenção do Estado dos*

Requisitos, Controle de Versão e Rastreabilidade dos Requisitos, representado no Diagrama de Fluxo de Trabalho da Disciplina *Gerenciamento de Requisitos* (Figura 18). Nesta Disciplina, pode ocorrer um fluxo alternativo quando for necessário atualizar o estado dos requisitos mas não existem alterações nos requisitos. Para esse fluxo, é realizada a Atividade *Rastreabilidade e Manutenção do Estado dos Requisitos*. Espera-se, com essa Disciplina, que o conjunto de requisitos possa ser alterado (inserções, alterações e remoções de requisitos). Essas alterações são tratadas de maneira que as dependências geradas sejam atualizadas controladamente.

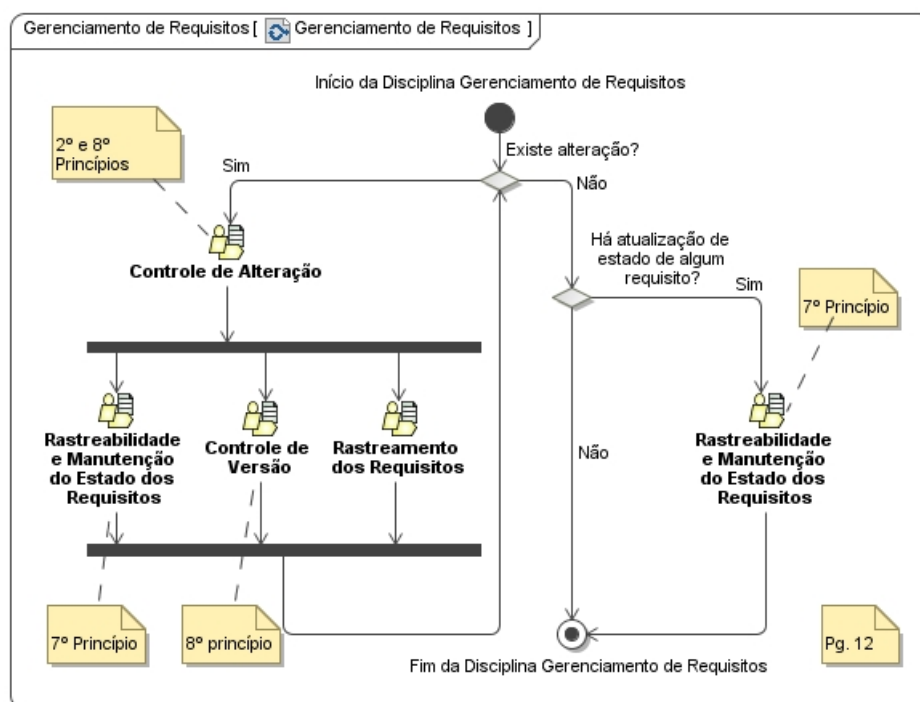


Figura 18 - Diagrama de Fluxo de Trabalho da Disciplina *Gerenciamento de Requisitos*.

Uma maneira de permitir as mudanças é garantir que os requisitos possam ser gerenciados tanto nos MT's quanto nos MA's. Para isso, o PERAF possui foco no Gerenciamento dos Requisitos para conseguir atender ao princípio dos MA's relacionado à aceitação de mudanças e ao mesmo tempo garantir a necessidade dos MT's de controlar as mudanças. O nível de rigor na realização das tarefas e das atividades varia de acordo com o grau de migração no qual a organização se encontra. Organizações que estão no início da migração, ou seja, que ainda possuem muita herança dos MT's podem adotar um maior nível de controle através da realização de todas as atividades e tarefas do PERAF, enquanto que organizações que estão no final da migração, ou seja, adotando plenamente os MA's, podem suprimir atividades e tarefas de modo a usar somente o que for necessário para o projeto.

3.3.2.1. Atividade Controle de Alteração

Na Atividade *Controle de Alteração* são realizadas tarefas relacionadas ao controle da evolução do conjunto de requisitos, à descrição da alteração, à análise da viabilidade da implementação e à atualização dos documentos adotados no projeto. Essa atividade possui o fluxo de Tarefas *Caracterizar Alteração*, *Analisar o Impacto da Alteração*, *Tomar Decisão de Alteração*, em paralelo ocorrem as Tarefas *Atualizar Documentos Adotados*, *Atualizar Diagramas Adotados* e *Atualizar Requisitos* e, seguindo o fluxo principal, ocorre a Tarefa *Atualizar Planejamento*. Esse fluxo de tarefas é representado no Diagrama de Fluxo de Trabalho da Atividade *Controle de Alteração* (Figura 19). Nesta Atividade, pode ocorrer um fluxo alternativo que encerra o fluxo da atividade caso a decisão de alteração seja negativa.

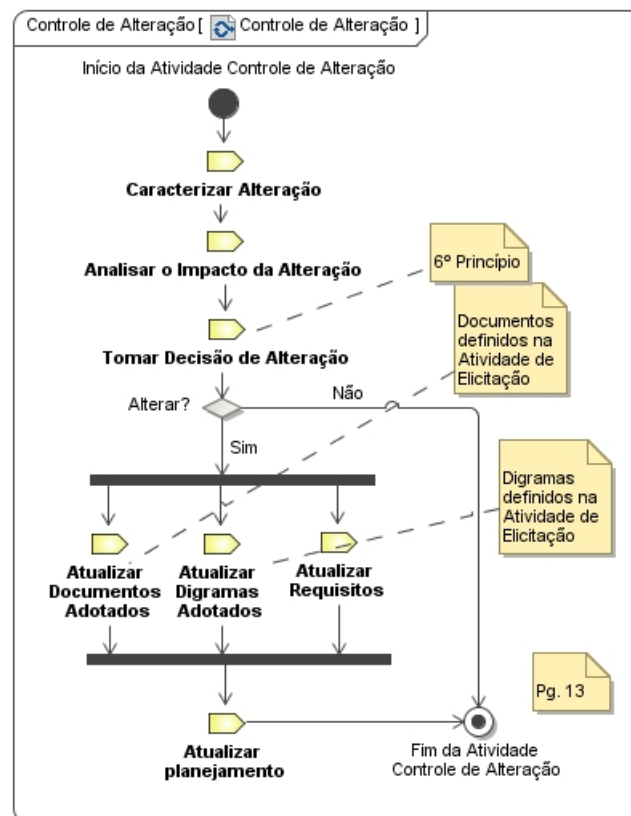


Figura 19 - Diagrama de Fluxo de Trabalho da Atividade *Controle de Alteração*.

A descrição das tarefas da Atividade *Controle de Alteração* é apresentada na Tabela 12 e a descrição das interações entre atores, artefatos e ferramentas é apresentada na Figura 20.

3.3.2.2. Atividade Rastreabilidade e Manutenção do Estado dos Requisitos

Na Atividade *Rastreabilidade e Manutenção do Estado dos Requisitos* são realizadas tarefas relacionadas à administração do estado de completude dos requisitos, o que consiste desde a atribuição de estado de completude aos requisitos até a geração de relatórios que reflitam o montante de requisitos completos. Essa atividade possui dois fluxos de tarefas, sendo que o primeiro ocorre quando não existe um esquema de classificação do estado dos requisitos, para esse fluxo são realizadas as Tarefas *Definir o Esquema de Classificação do Estado dos Requisitos*, *Atribuir Estado aos Requisitos* e *Gerar Relatório de Estado dos Requisitos*. O segundo fluxo ocorre quando existe um esquema de classificação do estado dos requisitos, para esse fluxo são realizadas as Tarefas: *Atualizar Estado dos Requisitos* e *Gerar Relatório de Estado dos Requisitos*. O fluxo de tarefas dessa atividade é representado no Diagrama de Fluxo de Trabalho da Atividade *Rastreabilidade e Manutenção do Estado dos Requisitos* (Figura 21).

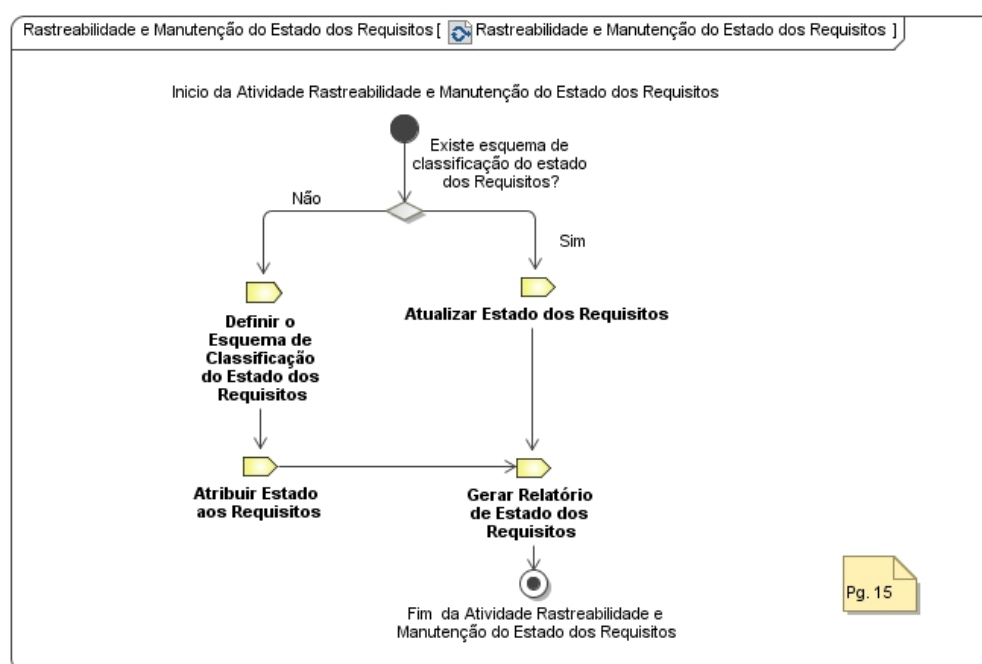


Figura 21 - Diagrama de Fluxo de Trabalho da Atividade *Rastreabilidade e Manutenção do Estado dos Requisitos*.

A descrição das tarefas da Atividade *Rastreabilidade e Manutenção do Estado dos Requisitos* é apresentada na Tabela 13 e a descrição das interações entre atores, artefatos e ferramentas é apresentada na Figura 22.

Tabela 13 - Descrição das Tarefas da Atividade Rastreabilidade e Manutenção do Estado dos Requisitos.

ATIVIDADE RASTREABILIDADE e MANUTENÇÃO DO ESTADO DOS REQUISITOS	
Tarefas	Descrição
Definir o Esquema de Classificação dos Requisitos	Deve-se definir o esquema a ser adotado para a classificação dos requisitos, o que pode ser desde porcentagem de progresso à gradações.
Atribuir o Estado aos Requisitos	Deve-se atribuir o estado definido na atividade <i>Definir o Esquema de Classificação dos Requisitos</i> aos requisitos.
Atualizar Estado dos Requisitos	Deve-se atribuir os estados definidos na atividade <i>Definir o Esquema de Classificação dos Requisitos</i> aos requisitos de modo a mantê-los atualizados.
Gerar Relatório de Estado dos Requisitos	Deve-se gerar relatórios que permitam a visualização do estado de completude de todos os requisitos. A ferramenta de gerência de requisitos definida para o gerenciamento de requisitos no projeto pode ser capaz de retornar o estado dos requisitos.

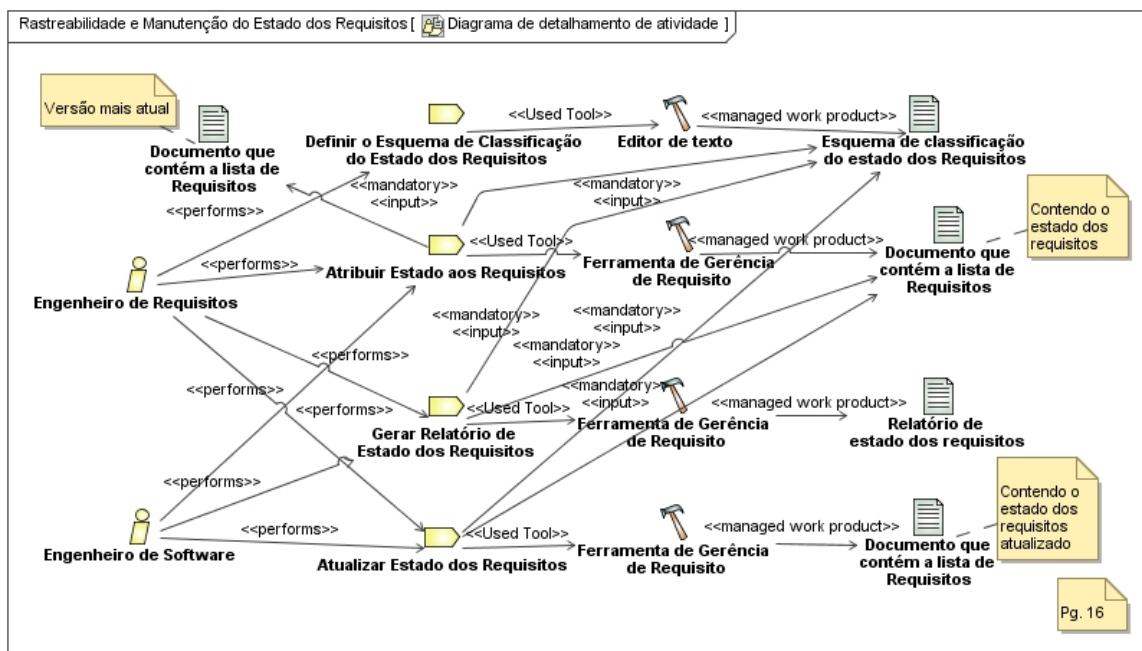


Figura 22 - Diagrama de detalhamento de atividade da Atividade Rastreabilidade e Manutenção do Estado dos Requisitos.

3.3.2.3. Atividade Controle de Versão

Na Atividade *Controle de Versão* são realizadas tarefas relacionadas a como o versionamento deve ser gerenciado, assegurando que os requisitos e os documentos sejam versionados na ocorrência de alguma alteração. Essa atividade possui dois fluxos de tarefas, sendo que o primeiro ocorre quando não existe o esquema de versão, para esse fluxo são realizadas as Tarefas *Definir Controle de Versão* e, em paralelo, *Usar o Controle de Versão nos Documentos e Diagramas Utilizados* e *Usar o Controle de Versão nos Requisitos Utilizados*. O segundo fluxo ocorre quando existe o esquema de versão para esse fluxo, são realizadas as Tarefas *Alterar a Versão dos Requisitos*, realizada caso haja

alteração na versão dos requisitos, e *Alterar a Versão dos Documentos Utilizados*, realizada caso haja alteração nos documentos gerados. O fluxo de tarefas é representado no Diagrama de Fluxo de Trabalho da Atividade *Controle de Versão* (Figura 23).

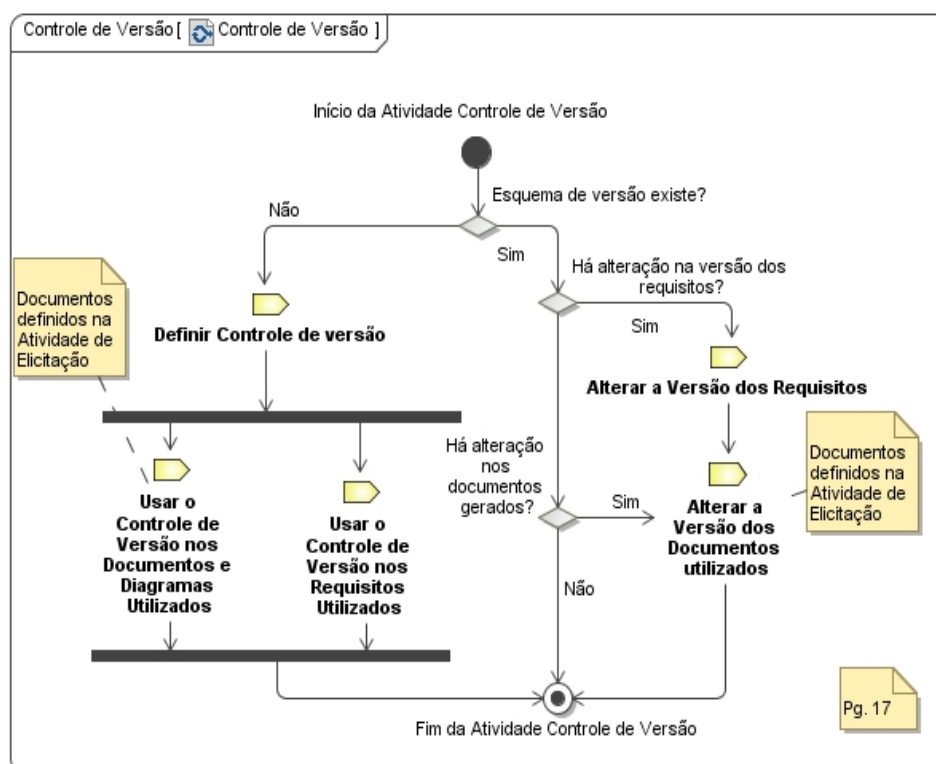


Figura 23 - Diagrama de Fluxo de Trabalho da Atividade *Controle de Versão*.

A descrição das tarefas da Atividade *Controle de Versão* é apresentada na Tabela 14 e a descrição das interações entre atores, artefatos e ferramentas é apresentada na Figura 24.

Tabela 14 - Descrição das Tarefas da Atividade *Controle de Versão*.

ATIVIDADE CONTROLE DE VERSÃO	
Tarefas	Descrição
Definir Controle de Versão	Deve-se definir um esquema de versão para os requisitos e para os documentos e diagramas utilizados no projeto de modo a garantir a sua unicidade.
Usar o Controle de Versões nos Documentos Utilizados	Deve-se aplicar o esquema de versionamento definido na tarefa <i>Definir Controle de Versões</i> nos documentos e nos diagramas utilizados no projeto.
Usar o Controle de Versão nos Requisitos Utilizados	Deve-se aplicar o esquema de versionamento definido na tarefa <i>Definir Controle de Versões</i> nos requisitos utilizados.
Alterar a Versão dos Documentos Utilizados	Deve-se alterar a versão dos documentos e dos diagramas utilizados no projeto sempre que houver alterações, remoções e adições de requisitos.
Alteração a Versão dos Requisitos	Deve-se alterar a versão dos requisitos sempre que houver alteração em sua sintaxe ou semântica.

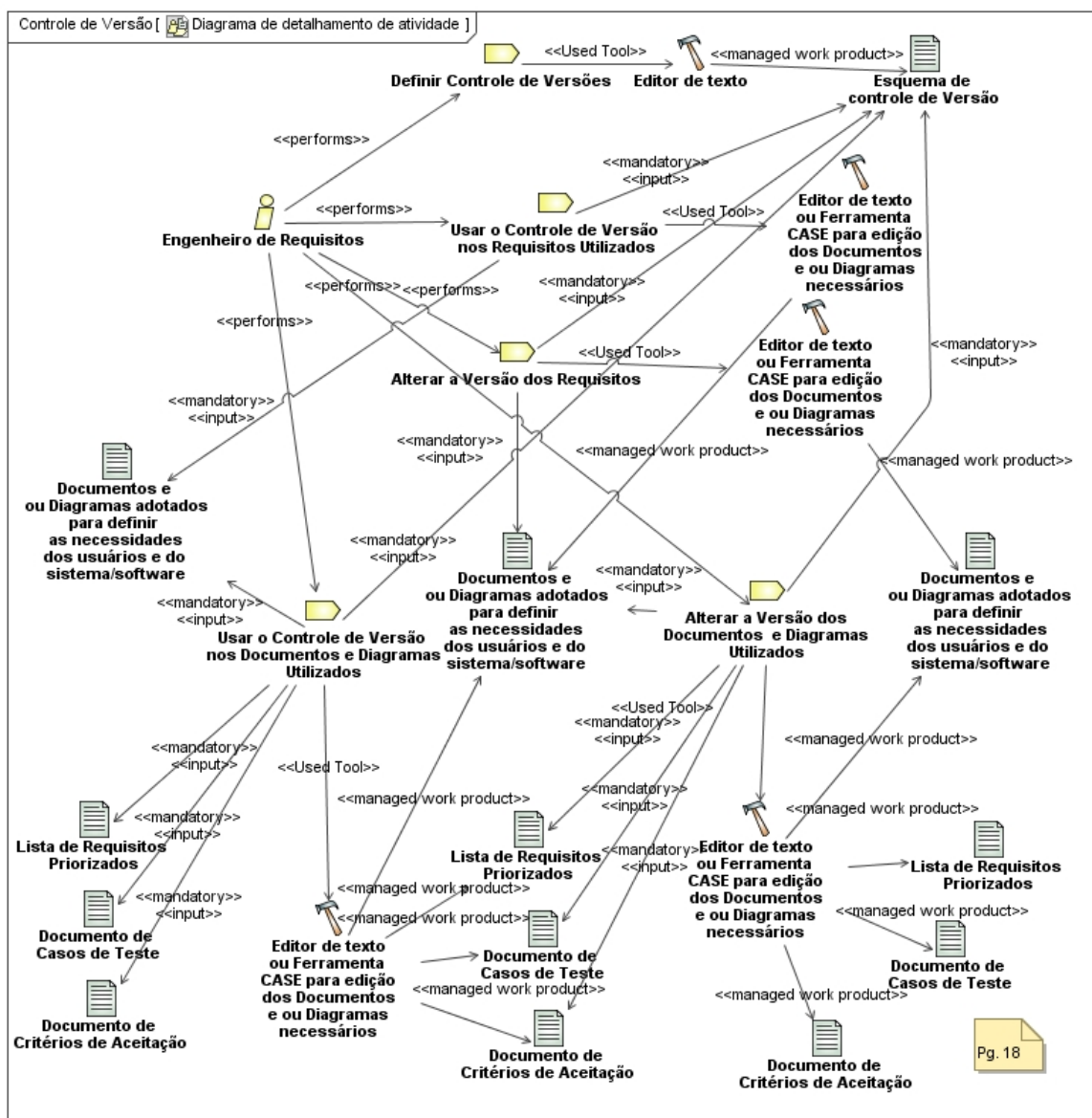


Figura 24 - Diagrama de detalhamento de atividade da Atividade Controle de Versão.

3.3.2.4. Atividade Rastreabilidade dos Requisitos

Na Atividade *Rastreabilidade dos Requisitos* são realizadas tarefas relacionadas à rastreabilidade dos requisitos entre os próprios requisitos e a outros elementos do software, assegurando que alterações sejam propagadas aos elementos dependentes do requisito alterado. Essa atividade possui dois fluxos de tarefas: o primeiro ocorre quando não existe um esquema de rastreabilidade entre os requisitos, para esse fluxo são realizadas as Tarefas *Definir Forma para a Rastreabilidade dos Requisitos* e, em paralelo, *Definir Rastreabilidade entre os Requisitos* e *Definir Rastreabilidade entre os Elementos do Sistema/Software*, e; o segundo fluxo ocorre quando existe rastreabilidade entre os requisitos e existe alguma alteração nos requisitos, para esse fluxo é realizada a Tarefa *Atualizar Rastreabilidade*. O fluxo de tarefas é representado no Diagrama de Fluxo de Trabalho da Atividade *Rastreabilidade dos Requisitos* (Figura 25).

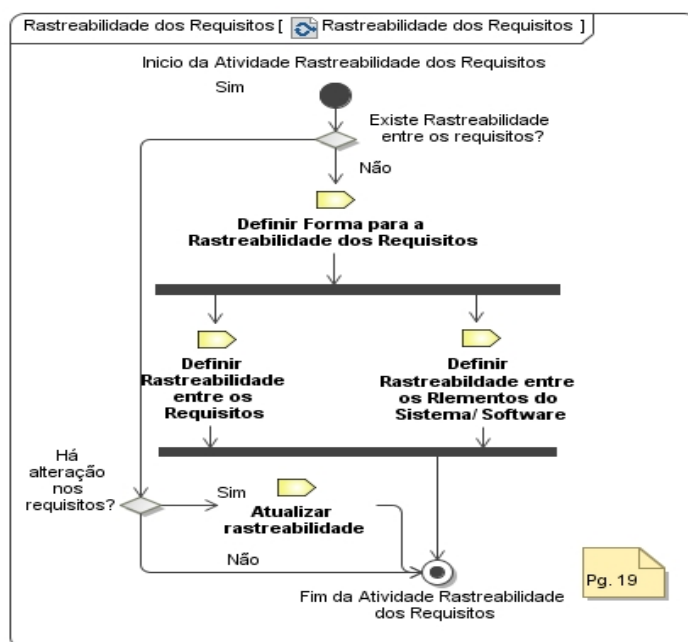


Figura 25 - Diagrama de Fluxo de Trabalho da Atividade *Rastreabilidade dos Requisitos*.

A descrição das tarefas da Atividade *Rastreabilidade dos Requisitos* é apresentada na Tabela 15 e a descrição das interações entre atores, artefatos e ferramentas é apresentada na Figura 26.

Tabela 15 - Descrição das Tarefas da Atividade *Rastreabilidade dos Requisitos*.

ATIVIDADE RASTREABILIDADE DOS REQUISITOS	
Tarefas	Descrição
Definir Forma para a Rastreabilidade dos Requisitos	Deve-se definir como será feita a rastreabilidade entre os requisitos, abordando desde a adoção de ferramentas até a forma de como serão criadas e armazenadas as <i>ligações</i> .
Definir Rastreabilidade entre os Requisitos	Deve-se definir <i>ligações</i> entre os requisitos do software.
Definir Rastreabilidade entre os Requisitos e os Elementos do Sistema/Software	Deve-se definir <i>ligações</i> entre os outros elementos do software.
Atualizar Rastreabilidade	Deve-se atualizar a definição das <i>ligações</i> entre os requisitos do software e das <i>ligações</i> entre os outros elementos do software.

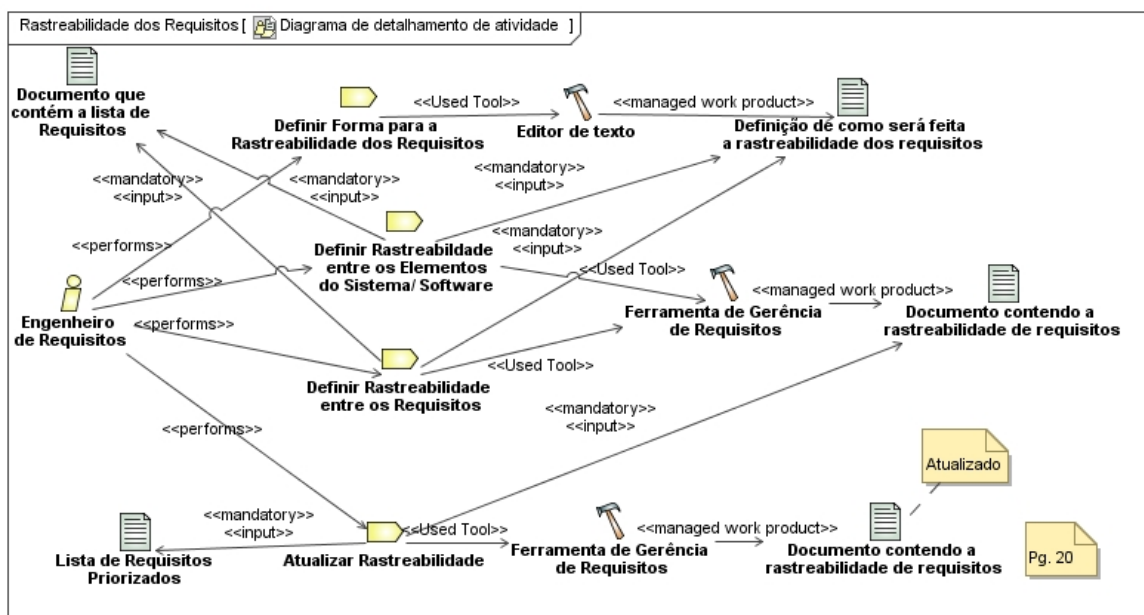


Figura 26 - Diagrama de detalhamento de atividade da Atividade *Rastreabilidade dos Requisitos*.

3.4. Considerações Finais

Neste capítulo, foi apresentada a descrição do PERAF com o objetivo de esclarecer a sua elaboração e o seu funcionamento. Foram apresentadas suas disciplinas, suas atividades e suas tarefas. O processo sinaliza que o desenvolvimento seja feito forma adaptável, deixando a cargo das equipes desenvolvedoras definirem quais artefatos, documentos, ferramentas, atividades e tarefas são necessários para seus projetos.

Capítulo 4

Avaliação do PERAF

4.1. Considerações Iniciais

A análise apresentada nesse capítulo considera os dados produzidos pelo uso do PERAF em dois estudos de caso, as observações do pesquisador e as opiniões dos envolvidos nesses estudos de caso.

Para cada um dos estudos de caso foi feita uma análise de discrepância (divergência entre o modelo formal do processo e a execução do mesmo), com a aplicação de questionários para avaliar cada atividade do PERAF. A instanciação do PERAF apresentou alguns problemas e além dos questionários, também foi feita a aplicação de um ciclo de melhoria de processo no estilo proposto no método DMAIC⁸, seguido no seis sigma⁹ e estudado por Werkema [Werkema, 2004]. Mesmo usando apenas parcialmente as prescrições do DMAIC acreditamos em sua utilidade para a detecção de problemas e auxílio na proposta de soluções.

A seção 4.2 apresenta o Estudo de Caso 1 realizado em um cenário onde a organização que ainda não têm um processo de Engenharia de Requisitos (ER) definido e formalizado e quer adotar um MA. A seção 4.3 apresenta o Estudo de Caso 2 realizado em um cenário de uma organização que está em processo de migração de uma MT para uma MA. Por fim, a Seção 4.4 apresenta a os resultados da aplicação parcial do DMAIC durante a avaliação do processo.

⁸ DMAIC: Método de resolução de problemas cuja sigla significa: D = definir o escopo do projeto; M = medir: determinar a localização do problema; A = analisar: determinar a causa do problema; I = melhorar: avaliar e implementar soluções, e; C = controlar: garantir que o alcance da meta seja mantido a longo prazo [Werkema, 2004].

⁹ Seis sigma: Conceito introduzido em 1986 por Bill Smith, engenheiro e cientista da divisão de comunicações a Motorola, em resposta ao aumento de queixas e pedidos de garantia [Barney, 2002].

4.2. Estudo de Caso 1

O Estudo de Caso 1 foi realizado no setor de Sistema de Informação do Laboratório de Estudos e Projetos em Manejo Florestal no Departamento de Ciências Florestais (DCF) da Universidade Federal de Lavras (LEMAF/DCF/UFLA), cujo objetivo é realizar pesquisa, ensino e extensão relacionados ao manejo de florestas nativas e plantadas. O LEMAF conduz diversos projetos através de parcerias e convênios com órgãos estaduais, federais e iniciativa privada.

4.2.1. Capacitação a Distância no Sistema de Zoneamento Econômico e Ecológico do Estado de Minas Gerais

Com recursos do governo do estado, em 2005, foi firmado um convênio entre o governo do estado Minas Gerais e a UFLA para a elaboração do Zoneamento Ecológico-Econômico de Minas Gerais (ZEE-MG). Um dos objetivos desse convênio é ofertar um curso de capacitação de servidores dos órgãos de governo e representantes de conselhos, associações, comitês, federações, organizações não governamentais e sociedade civil em geral no acesso e na utilização do ZEE-MG de modo que possam realizar o planejamento territorial e futuras análises dos processos de regularização e gestão ambiental.

Em termos estruturais, o curso foi planejado para capacitar 1450 pessoas, organizado em 9 módulos e estruturado na modalidade a distância (EAD/ZEE-MG), realizado em um ambiente virtual de ensino. O enfoque tecnológico da capacitação foi voltado para a estruturação de salas virtuais, adequação de material e criação da infraestrutura necessária para a realização e manutenção do curso. Para isso, foram levantados 29 requisitos os quais são apresentados na Tabela 16.

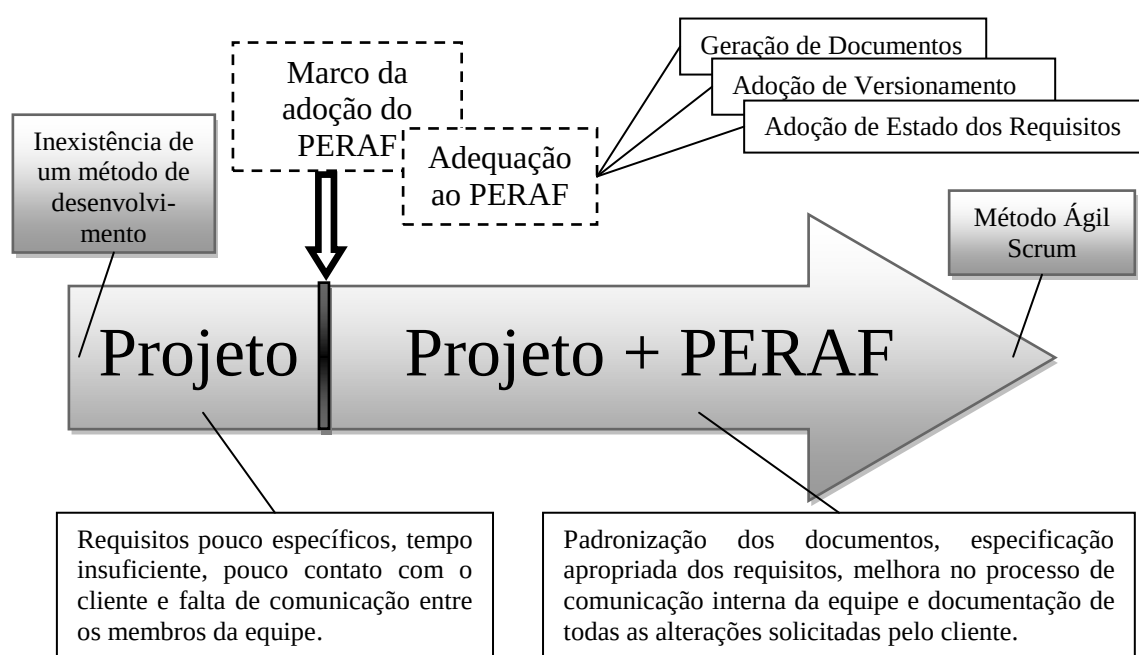
4.2.2. Caracterização do Cenário

A equipe de desenvolvedores apresenta um cenário na tentativa de adoção da metodologia ágil de desenvolvimento de software Scrum a partir da inexistência um método e de um processo formalizado para o desenvolvimento, sem um processo de gerenciamento de requisitos e lidando com a não documentação das alterações de requisitos.

Tabela 16 – Requisitos do Projeto EAD/ZEE-MG.

Tabela de Requisitos	
1) Preparação do servidor	16) Construção das salas
2) Adequar material didático	17) Definição do processo de contratação dos tutores
3) Definição/Customização da interface	18) Definição do número de horas de trabalho para os tutores
4) Elaboração do edital de inscrição do curso	19) Definição do processo de contratação das secretárias
5) Elaboração das fichas de inscrição no curso	20) Definição do número de secretárias necessárias
6) Definição do número de alunos por sala	21) Definição do número de horas de trabalho de cada secretária
7) Definição do número de tutores por disciplina	22) Definição dos tipos de relatórios necessários
8) Definição do conjunto de atividades para cada disciplina	23) Definição da metodologia de avaliação para o curso
9) Definição do período de duração de cada disciplina	24) Migração
10) Definição do processo de capacitação dos tutores	25) Inserção de nova disciplina
11) Definição do processo de capacitação dos alunos	26) Definição de metodologia para a disciplina de reposição
12) Definição do cronograma do curso	27) Disponibilizar disciplinas aos participantes
13) Definição do meio de comunicação com os alunos	28) Disponibilizar o curso aos participantes
14) Cadastro dos usuários	29) Criação de uma disciplina de reposição
15) Envio de <i>login</i> e senha para os usuários	

O PERAF foi inserido no projeto após o contato inicial com o cliente. Para isso, foi feita uma adequação do estado no qual o projeto se encontrava para atender as necessidades do PERAF. A partir desse momento o PERAF foi seguido pela equipe do projeto. A linha do tempo da execução do projeto e o instante da inserção do PERAF são apresentados na Figura 27, na qual é caracterizado o cenário da inexistência de um método de desenvolvimento de software antes da aplicação do PERAF. No marco foi adotado conjuntamente o método ágil Scrum. Por fim, mostra as adequações que foram necessárias para a sua adoção, assim como mostra os cenários de abordagem dos artefatos de ER antes e depois da implantação do PERAF.

**Figura 27 - Adoção do PERAF no Projeto EAD/ZEE-MG (Linha do Tempo).**

A Figura 28 apresenta uma descrição das principais atividades realizadas no projeto que geraram algum produto ou utilizaram de alguma ferramenta CASE de desenvolvimento de software. Uma visão de como o projeto pode ser compreendido em função dos produtos nele gerados. Pode-se identificar a participação de seis atores: *Cliente, Equipe de TI, Equipe de Design Gráfico, Professores, Turma Piloto e Equipe da Empresa Responsável Pela Migração*. Além disso, as seguintes tarefas foram realizadas: *Fornecer Servidor, Configurar Servidor, Customizar a Plataforma do Curso, Gerar Material, Adequar o Material, Planejar Estrutura do Curso, Criar um Protótipo para Avaliação, Validar o Protótipo, Criar Versão Estável do Curso, Gerar o Material para Migração e Realizar a Migração*. Como resultados da realização dessas tarefas, foram gerados os seguintes artefatos de software: *Servidor Configurado, Plataforma LMS¹⁰ Customizada, Material Customizado, Material Adequado, Protótipo, Versão Estável do Curso e Material para Migração*.

Inicialmente foi realizada uma apresentação do PERAF para todos os envolvidos no projeto com o intuito de dar uma visão geral sobre o processo. Posteriormente e durante o projeto foram usadas cópias do processo e painéis com a descrição do processo para o acompanhamento do fluxo de atividades do mesmo, contribuindo assim para sua familiarização pela equipe. Para a coleta de dados foi realizado um acompanhamento diário sobre o andamento das atividades do projeto de modo a coletar os dados para a avaliação do processo.

4.2.3. Análise do questionário de discrepâncias

Nesta seção, são descritos os resultados da aplicação do questionário de discrepância sobre uso do PERAF no projeto EAD/ZEE-MG. A aplicação do questionário objetivou encontrar discrepâncias entre o modelo formal do PERAF e sua instanciação no projeto EAD/ZEE-MG.

Durante a análise dos questionários aplicados percebeu-se:

- Conflitos de responsabilidades de diferentes equipes sobre o projeto pela inexistência de projetos semelhantes para serem consultados ou quando os requisitos não estavam diretamente envolvidos com a tarefa;

¹⁰ LMS: *Learning Management Systems* ou Sistema de Gestão da Aprendizagem (SGA).

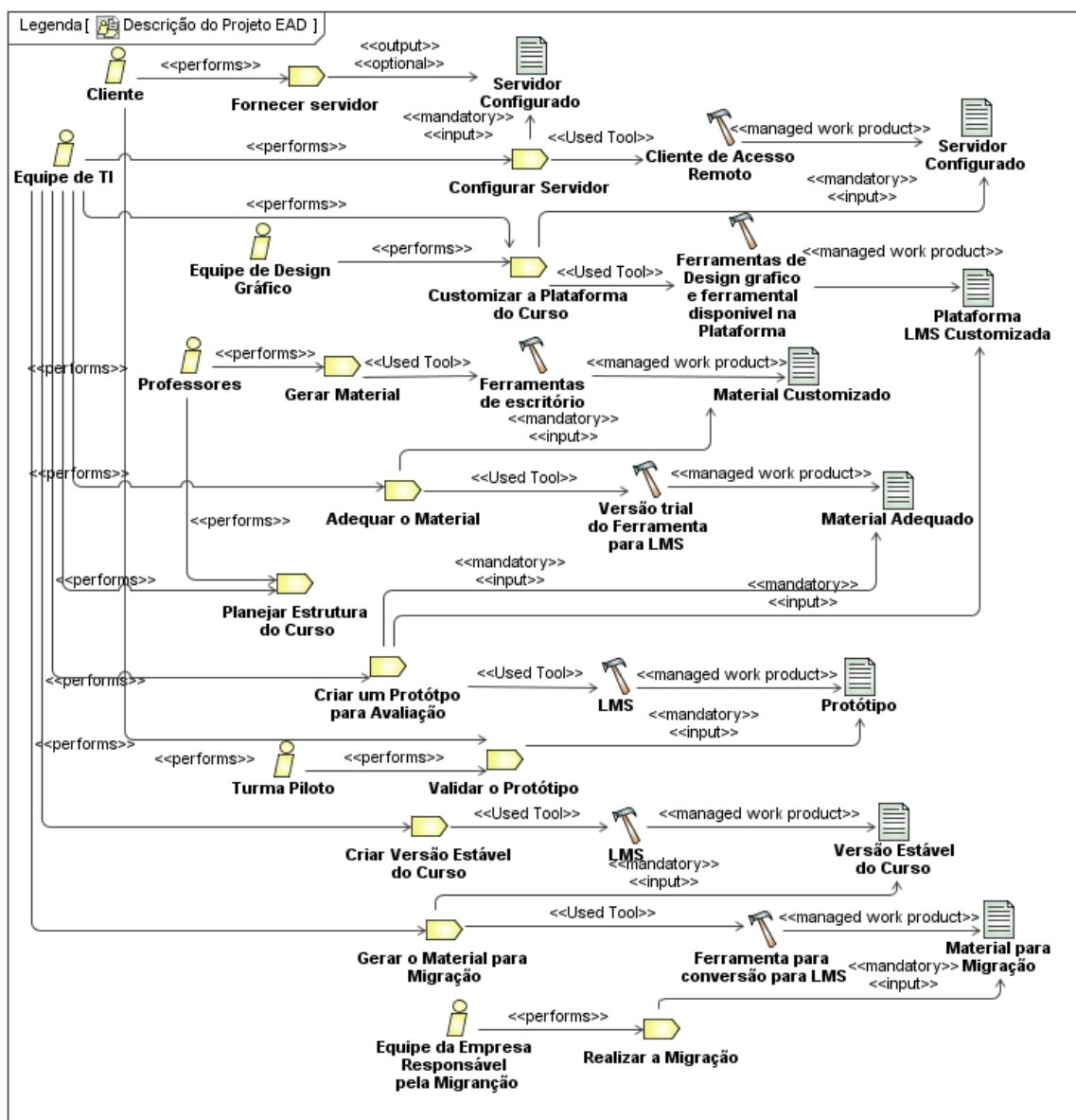


Figura 28 - Descrição do Processo de Desenvolvimento do Projeto EAD/ZEE-MG.

- Uma discrepância do tipo inconsistência detectada na definição das responsabilidades dos atores. Este usuário justificou sua resposta alegando que "Não foi definida a responsabilidade sobre quem geraria a lista de contatos". O usuário alegou que "a lista de contatos foi apenas repassada a ele, não sendo parte de suas atribuições realizá-la". Justifica-se a resposta pelo fato de que o cliente não decidiu sobre a lista, ela foi simplesmente repassada a estes. No término da análise de discrepância, o autor do processo constatou: "Fica claro que ambos os atores devem estar envolvido na tarefa, no entanto não fica claro o que cada ator deve desempenhar caso eles não atuem conjuntamente";

- Não foram utilizados diagramas para os requisitos trabalhados pela equipe de desenvolvimento; logo os artefatos e as atividades relacionados a eles não foram abordados na execução do processo;
- Não houve a adoção de casos de teste e casos de aceitação no projeto; logo, os artefatos e as atividades relacionados a eles não foram abordados na execução do processo;
- Não foi utilizada nenhuma ferramenta de gerência de requisitos para os requisitos trabalhados pela equipe de desenvolvimento. Sendo utilizadas apenas ferramentas de escritório;
- Não foram gerados relatórios de estados para os requisitos, e;
- Não foi utilizado controle de versão nos requisitos trabalhados pela equipe de desenvolvimento.

Na aplicação dos questionários de discrepância sobre as atividades do processo constatou-se que a utilização do PERAF permitiu a identificação de duas discrepâncias na Atividade Elicitação. Essas discrepâncias estão relacionadas à necessidade da definição clara dos papéis dos atores na Atividade Definir contato para dar suporte no Desenvolvimento de Requisitos.

Todavia foi constatado que a utilização do PERAF conferiu liberdade aos envolvidos no projeto ao escolherem como lidar com os requisitos e documentações do projeto, assim como se mostrou adaptável aos artefatos requeridos pelo projeto e adequado às necessidades do mesmo.

4.2.4. Análise dos Questionários de Pré e Pós Experimento

A Tabela 17 apresenta um quadro com a comparação dos questionários de pré e pós experimento. Foi constatado que o PERAF solucionou os problemas: “*Requisitos pouco específicos*”, “*Tempo insuficiente*” e “*Falta de comunicação entre os membros da equipe do projeto*”. Houve melhoria nas respostas dos questionamentos: i) “*Como sua equipe lida com as mudanças solicitadas pelo cliente após a elicitação dos requisitos?*” cuja resposta passou de: “*Decide o que fazer em conjunto com a equipe*” para: “*Decide o que fazer em conjunto com a equipe e com o cliente*”, e; ii) “*Como sua equipe lida com a documentação das mudanças solicitadas pelo cliente após a elicitação dos requisitos?*” cuja resposta

passou de: “*Documenta apenas as alterações significativas*” para: “*Documenta todas as alterações*”.

De acordo com o questionário de pós experimento, uma série de benefícios foram levantados a partir da utilização do PERAF, sendo eles: i) padronização dos documentos (acabou facilitando a leitura dos mesmos); ii) definição dos papéis; iii) especificação apropriada dos requisitos; iv) mensurar de modo realista o tempo a ser empregado no desenvolvimento do projeto; v) melhora no processo de comunicação interna da equipe; vi) as alterações solicitadas pelo cliente passaram a ser tratadas pelo conjunto equipe de desenvolvimento e cliente, e; vii) passou-se a documentar todas as alterações solicitadas pelo cliente.

Tabela 17 – Análise de questionários EAD/ZEE.

Análise da evolução das respostas do questionário de pré e pós-experimento			
Questionamento	Respostas	Resposta no pré-questionário	Resposta no pós-questionário
Principal problema encontrado por sua equipe no desenvolvimento de software: / qual(is) problemas foram solucionados pela aplicação do PERAF?	Falta de experiência		
	Requisitos pouco específicos	X	X - Solucionou
	Tempo insuficiente	X	X - Solucionou
	Pouco contato com o cliente		
	Falta de comunicação entre os membros da equipe	X	X - Solucionou
Como sua equipe lida com a documentação dos projetos?	Não existe documentação		
	A documentação é inserida no próprio código do projeto		
	Documentos formais são elaborados	X	X
Como sua equipe lida com as mudanças solicitadas pelo cliente após a elicitação dos requisitos?	ignora		
	Decide o que fazer em conjunto com a equipe	X	
	Decide o que fazer em conjunto com a equipe e com o cliente		X
	Acata o pedido de mudanças		
Como sua equipe lida com a documentação das mudanças solicitadas pelo cliente após a elicitação dos requisitos?	Não documenta		
	Documenta apenas as alterações significativas	X	
	Documenta todas as alterações		X

Os usuários do processo elogiaram sua clareza e intuitividade de leitura, e alegam que o processo atendeu de modo satisfatório ao processo de desenvolvimento da organização, uma vez que conferiu liberdade à equipe de desenvolvimento sobre quais documentos utilizar.

Foi sugerido pelo usuário do processo que futuramente sejam listados gabaritos de documentação e diagramas para os métodos de desenvolvimento de software a fim de

minimizar o tempo gasto pela equipe de desenvolvimento na busca de quais artefatos utilizar.

4.3. Estudo de Caso 2

O Estudo de Caso 2 foi realizado na Divisão de Sistemas de Informação da Diretoria de Gestão de Tecnologia da Informação da Universidade Federal de Lavras (DGTI) que é um órgão suplementar diretamente subordinado à Pró-Reitora de Planejamento e Gestão e que tem por objetivo desenvolver as atividades de gestão da Tecnologia da Informação da UFLA.

Possui 5 desenvolvedores e 4 estagiários (o quadro de estagiários apresenta alta rotatividade). Na época em que este trabalho foi desenvolvido os desenvolvedores usavam a ferramenta CASE DotProject¹¹ e estavam com dificuldades em gerenciar as alterações nos requisitos e atualização de seus documentos de requisitos.

4.3.1. DES - Sistema de Resíduos Químicos

O Laboratório de Gerenciamento de Resíduos Químicos tem como funções gerenciar e tratar resíduos assim como é responsável também pela conscientização sobre os riscos dos resíduos químicos. Para melhor desenvolver essas tarefas foi criado um software de gerenciamento de resíduos químicos na UFLA, desenvolvido pelo DGTI e enfocando ações preventivas de minimização (redução, reuso e reciclagem) com o objetivo de conscientizar a comunidade universitária da importância do gerenciamento de resíduos laboratoriais.

O software a ser desenvolvido para a UFLA irá gerenciar o controle de produtos químicos, resíduos, reagentes, e produtos químicos recuperados. Ele irá controlar atividades de: compra, distribuição e chamadas de recolhimento. O Software foi dividido em dois módulos, sendo que para o Módulo 1 foram levantados 19 requisitos iniciais para o desenvolvimento do software (Tabela 18). O Módulo 2 não foi abordado no Estudo de Caso devido aos prazos do projeto serem futuros com relação à defesa dessa dissertação.

¹¹ DotProject: ferramenta livre de gerência de projetos.

4.3.2. Caracterização do Cenário

A equipe de desenvolvedores apresenta um cenário de tentativa de migração do MT para o MA Scrum. No entanto, não existe o detalhamento e formalização do processo tradicional para o desenvolvimento, tão pouco existe um processo de gerenciamento de requisitos ou documentação das alterações de requisitos.

Tabela 18 – Requisitos do Projeto DES - Sistema de resíduos químicos.

Tabela de Requisitos do Módulo 1 do Projeto	
• RF001 - Acessar o Sistema	• RF0011 – Alterar Usuário
• RF002 – Gerenciar Estoque (Buscar)	• RF0012 – Inserir Reagentes
• RF003 – Gerenciar Estoque (Listar)	• RF0013 – Solicitação de Recolhimento
• RF004 – Gerenciar Estoque (Distribuir)	• RF0014 – Inserir Bombonas
• RF005 – Inserir Fornecedor	• RF0015 – Inserir Resíduos
• RF006 – Inserir Comprador	• RF0016 – Fechamento do Mês
• RF007 – Lançar Nota Fiscal	• RF0017 – Fechamento Geral
• RF008 – Inserir Departamento	• RF0018 – Reserva Bombonas
• RF009 – Inserir Laboratório	• RF0019 – Solicitação de Resíduos
• RF0010 – Inserir Usuário	

As diretrizes do PERAF foram inseridas no desenvolvimento do projeto após o contato inicial com o cliente, assim como ocorreu no Estudo de Caso 1. Foi feita uma adequação do estado no qual projeto se encontrava para atender as necessidades do PERAF, e a partir desse momento o PERAF foi seguido pela equipe do projeto (Figura 29). Nela é caracterizado o cenário da existência de um método tradicional de desenvolvimento de software (Cascata) antes da aplicação do PERAF. No marco de adoção do PERAF foi adotado conjuntamente o método ágil Scrum. Por fim, são mostradas as adequações que foram necessárias para a sua adoção, assim como mostra os cenários de abordagem dos artefatos de ER antes e depois da implantação do PERAF.

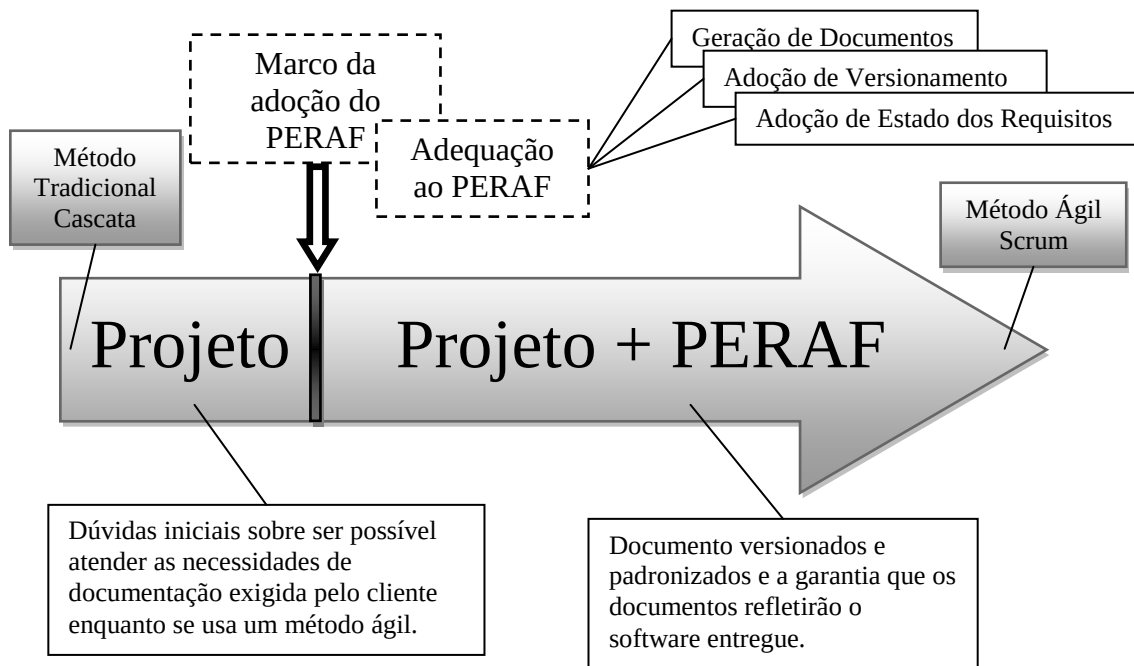


Figura 29 – Linha do tempo para a adoção do PERAF no projeto DES.

A Figura 30 apresenta uma descrição das principais atividades realizadas no projeto que geraram algum produto e ou utilizaram de alguma ferramenta CASE de desenvolvimento de software. Essa figura nos fornece uma visão de como o projeto pode ser compreendido em função dos produtos gerados. É possível distinguir a participação de três atores sendo eles: *Engenheiro de Requisitos*, *Cliente* e *Desenvolvedores*. As tarefas realizadas foram: *Levantar/Documentar Requisito*, *Desenhar Diagramas de Casos de Uso*, *Modelar Banco de Dados*, *Cadastrar Tarefas no DotProject*, *Validar Requisitos e Desenvolver o Sistema*. Os produtos gerados foram: *Documento de Requisitos*, *Diagramas de Casos de Uso*, *Modelagem do Banco de Dados*, *Product Backlog* e *Versão Preliminar do Produto*.

Inicialmente foram feitas reuniões diárias com os envolvidos no projeto com o intuito de explicar de forma clara o funcionamento do PERAF, além das reuniões foram disponibilizadas cópias do fluxo do processo e da descrição das tarefas. Após a fase de conhecimento do processo foram realizadas reuniões semanais através das quais foi possível realizar o acompanhamento da adoção do PERAF pelos desenvolvedores.

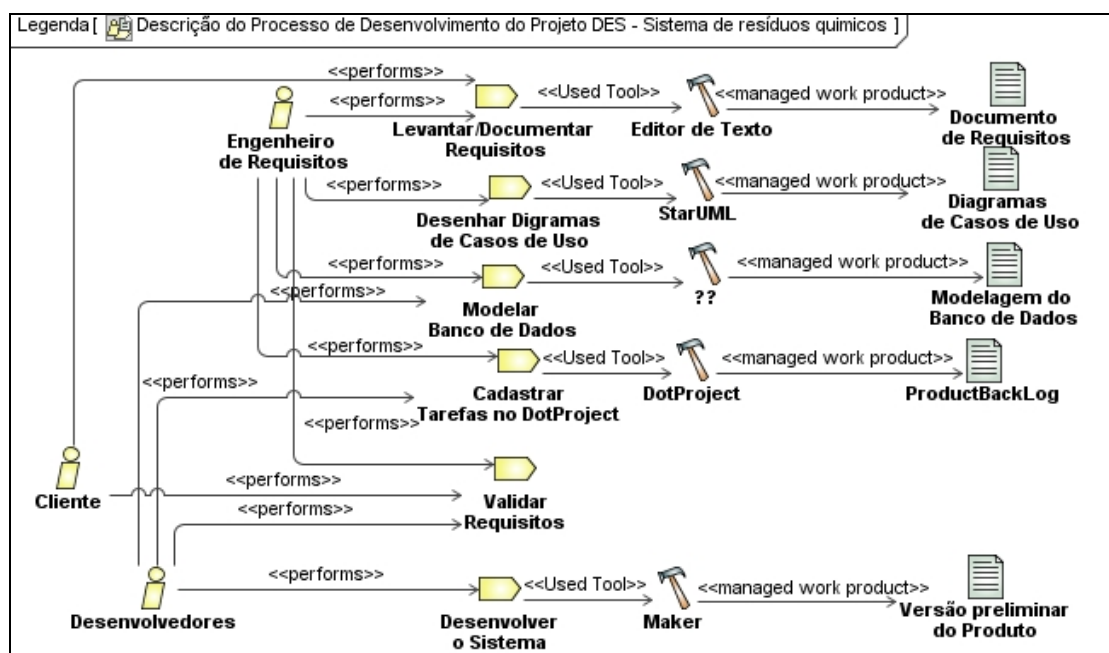


Figura 30 - Descrição do Processo de Desenvolvimento do Projeto DES.

4.3.3. Análise do questionário de discrepâncias

Nesta seção é descrita a avaliação do PERAF no projeto DES - Sistema de resíduos químicos, onde são apresentados os resultados de análises que objetivam encontrar discrepâncias entre o modelo formal do PERAF e sua instanciação no projeto EAD/ZEE-MG.

Durante a análise dos questionários aplicados percebeu-se:

- Houve falha na execução de tarefas por falta de definição de quem é o responsável por realizá-las no caso em que dois ou mais atores estejam envolvidos;
- Não houve formalização da análise de viabilidade de requisitos uma vez que todos os requisitos foram considerados viáveis e analisados;
- A priorização dos requisitos foi feita somente para definir o que seria entregue a cada módulo, sendo feita em uma granularidade maior do que a analisada no questionário de discrepância e, assim, não afetou a ordem em que os 19 requisitos tratados nos questionários;
- Não foram usados gabaritos para a elaboração dos diagramas utilizados no projeto, pois, o processo de construção dos diagramas é conhecido pelos engenheiros de requisitos;

- Não houve a adoção de casos de teste e casos de aceitação no projeto; logo, os artefatos e as atividades relacionados a eles não foram abordados na execução do processo;
- Não foi utilizada nenhuma ferramenta CASE para edição dos documentos e dos diagramas das tarefas dessa atividade;
- As alterações nos requisitos não foram caracterizadas formalmente, uma vez que as alterações foram mínimas e não impactantes;
- Não foram gerados relatórios dos estados dos requisitos;
- Não foi definido o esquema classificação dos requisitos, uma vez que foi usado o esquema de da ferramenta *DotProject*;
- Não foi utilizado o controle de versão dos requisitos e consequentemente não foi necessário o uso de uma ferramenta para o gerenciamento de requisitos para essa tarefa, e;
- Não foi preciso atualizar o documento contendo a rastreabilidade dos requisitos uma vez que as alterações foram mínimas e não impactantes.

Em uma análise geral sobre a avaliação de todas as atividades do processo, foi constatado que a utilização do PERAF, assim como para o Estudo de Caso 1, conferiu liberdade aos envolvidos no projeto de escolherem as ferramentas para lidar com os requisitos e documentações do projeto, permitindo variações, desde editores de texto até ferramentas de gestão de requisitos. Assim como se mostrou adaptável aos artefatos requeridos pelo projeto.

Com base no Estudo de Caso do projeto DES - Sistema de resíduos químicos o PERAF se mostrou adequado às necessidades do projeto.

4.3.4. Análise dos Questionários de Pré e Pós Experimento

A Tabela 19 apresenta um quadro com a comparação dos questionários de pré e pós experimento. Foi constatado que o PERAF solucionou o problema: “*Requisitos pouco específicos*”. Houve melhoria nas respostas dos questionamentos: “*Como sua equipe lida com a documentação dos projetos?*” cujas respostas passaram de: “*A documentação é inserida no próprio código do projeto*” e “*Documentos formais são elaborados*” para

somente: “*Documentos formais são elaborados*”. As ações se mantiveram iguais para os demais questionamentos.

De acordo com o questionário de pós experimento, uma série de benefícios foram levantados sobre a utilização do PERAF, sendo eles: i) O principal ganho para a equipe foi uma documentação mais detalhadas do projeto que possibilitou um melhor entendimento dos requisitos evitando assim erros futuros, outros ganhos foram: ii) Experiência da equipe com a aplicação de um processo de ER até então inexistente no setor; iii) Melhoras nas especificações, e validação dos requisitos; iv) Diminuição das inconsistências entre o software desenvolvido e as necessidades do cliente, e; v) A geração dos artefatos do processo possibilitou identificar erros futuros e uma melhor viabilização dos requisitos.

A aplicação de um questionário após a finalização do projeto DES - Sistema de resíduos químicos no qual o PERAF foi utilizado explicita que o grau de dificuldade de execução do PERAF varia de mediano, pois a inexperiência da equipe com processos de ER gerou confusão entre os conceitos de desenvolvimento software e de ER, a fácil, por se adaptar a boa parte das necessidades da equipe e ao método de desenvolvimento adotado pela equipe (Scrum).

Tabela 19 – Análise de questionários DES.

Análise da evolução das respostas do questionário de pré e pós-experimento			
Questionamento	Respostas	Resposta no pré-questionário	Resposta no pós-questionário
Principal problema encontrado por sua equipe no desenvolvimento de software: / qual(is) problemas foram solucionados pela aplicação do PERAF?	Falta de experiência		
	Requisitos pouco específicos	X	X - Solucionou
	Tempo insuficiente	X	
	Pouco contato com o cliente		
	Falta de comunicação entre os membros da equipe	X	
Como sua equipe lida com a documentação dos projetos?	Não existe documentação		
	A documentação é inserida no próprio código do projeto	X	
	Documentos formais são elaborados	X	X
Como sua equipe lida com as mudanças solicitadas pelo cliente após a elicitação dos requisitos?	ignora		
	Decide o que fazer em conjunto com a equipe		
	Decide o que fazer em conjunto com a equipe e com o cliente	X	X
	Acata o pedido de mudanças		
Como sua equipe lida com a documentação das mudanças solicitadas pelo cliente após a elicitação dos requisitos?	Não documenta		
	Documenta apenas as alterações significativas	X	X
	Documenta todas as alterações		

4.4. Ciclo de melhoria

O método DMAIC, seguido no seis sigma, foi utilizado para uma avaliação parcial do PERAF enquanto conjunto de diretrizes para ER com o intuito de detectar, e propor soluções para os seus principais problemas. Para este estudo foram abordadas apenas as etapas: definir, medir, analisar e melhorar; uma vez que a etapa controlar requer que as soluções propostas sejam implantadas e testadas por um período de tempo que excede o tempo final para a realização desta dissertação. A avaliação foi feita nas Disciplinas *Desenvolvimento de Requisitos* e *Gerenciamento de Requisitos*.

Foi definido um problema prioritário para cada Disciplina do PERAF (Tabela 20 e Tabela 21) (Etapa Definir);

Tabela 20 – Definição do problema Prioritário – Desenvolvimento de Requisitos.

Declaração do Problema: Definir quais documentos e diagramas serão necessários, uma vez que eles dependem da metodologia adotada. Em outras palavras, existe um número considerável de metodologias para elicitación de requisitos e consequentemente uma série de documentos e diagramas para sua utilização. Esta combinação torna o trabalho das partes interessadas complicado no que se refere à seleção destes itens.
Observação: Existe um número considerável de metodologias para elicitación de requisitos e consequentemente uma série de documentos e diagramas para sua utilização. Esta combinação torna o trabalho das partes interessadas complicado no que se refere à seleção destes itens.
Análise: Conhecimento parcial da metodologia que está sendo empregada, bem como dos documentos e diagramas que lhe são adequados.
Plano de Ação: No início de cada projeto, convocar todos os envolvidos para um dia de discussão sobre o projeto. Neste dia, a metodologia de desenvolvimento será escolhida, seus documentos e diagramas escolhidos/explicados a partir de uma lista previamente sugerida pelo engenheiro de requisitos.
Ação: Implantação do plano de ação acima descrito;
Verificação: Avaliação de resultados para verificação se a ação foi eficaz na eliminação ou minimização do problema. Esta avaliação será baseada na comparação dos tempos de desenvolvimentos de projetos semelhantes, na verificação de documentação de projetos similares e em entrevistas com os desenvolvedores.
Padronização: A partir da escolha da metodologia de desenvolvimento, introduzir os diagramas e documentos que lhe são pertinentes.

Tabela 21 – Definição do problema Prioritário – Gerenciamento de Requisitos.

Declaração do Problema: Definir a(s) ferramenta(s) para dar suporte no processo de engenharia de requisitos, uma vez que a escolha depende das necessidades do projeto associado às funcionalidades oferecidas pelas ferramentas. Existe uma grande quantidade de ferramentas com diferentes funcionalidades tornando complexa a escolha da ferramenta adequada para o projeto.
Observação: A escolha da ferramenta é complicada devido à grande quantidade de ferramentas disponíveis.
Análise: Dificuldade de analisar uma grande gama de ferramentas e conhecimento parcial sobre as funcionalidades necessárias para o projeto.
Plano de Ação: No início de cada projeto, convocar todos os envolvidos para um dia de discussão sobre o projeto. Neste dia, a(s) ferramenta(s) deverá(ão) ser escolhida(s) a partir da lista de ferramentas sugerida pelo engenheiro de requisitos.
Ação: Implantação do plano de ação acima descrito;
Verificação: Avaliação de resultados para verificação se a ação foi eficaz na eliminação ou minimização do problema. Esta avaliação será baseada na comparação dos tempos de desenvolvimentos de projetos semelhantes com a utilização de diferentes ferramentas, na verificação de documentação de projetos gerados por essas diferentes ferramentas e em entrevistas com os desenvolvedores.
Padronização: Fornecer juntamente com a Atividade de Elicitação a lista das ferramentas mais indicadas para a(s) metodologia(s) adotada(s) no projeto.

Os problemas foram identificados para cada uma das Disciplinas do PERAF: para a Disciplina *Desenvolvimento de Requisitos* o problema foi detectado na Tarefa *Planejar a Elicitação* da Atividade *Elicitação* que contém a (Figura 31 e Figura 32); e para a Disciplina *Gerenciamento de Requisitos* o problema foi detectado nas Atividades *Rastreabilidade e Manutenção do Estado dos Requisitos*, *Controle de Versão e Rastreabilidade dos Requisitos* (Figura 34, Figura 35, Figura 33 e Figura 36).

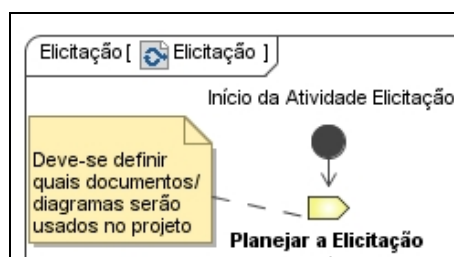


Figura 31 - Parte do Diagrama de Fluxo de Trabalho da Atividade *Elicitação* que contém a Tarefa *Planejar a Elicitação*.

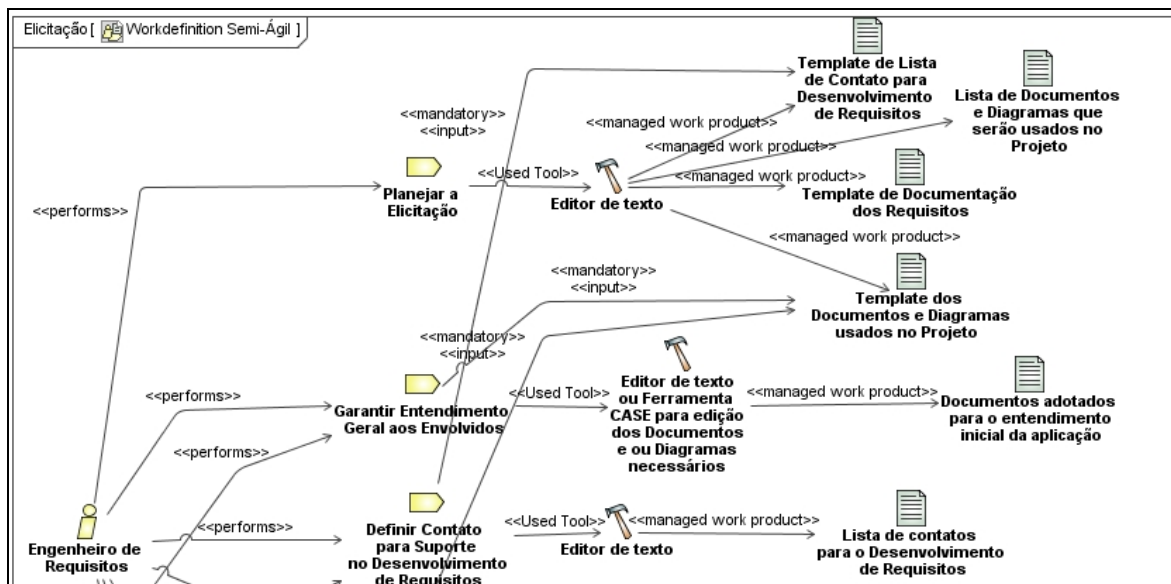


Figura 32 - Parte do Diagrama de detalhamento de atividade da Atividade *Elicitação* que contém a Tarefa *Planejar a Elicitação*.

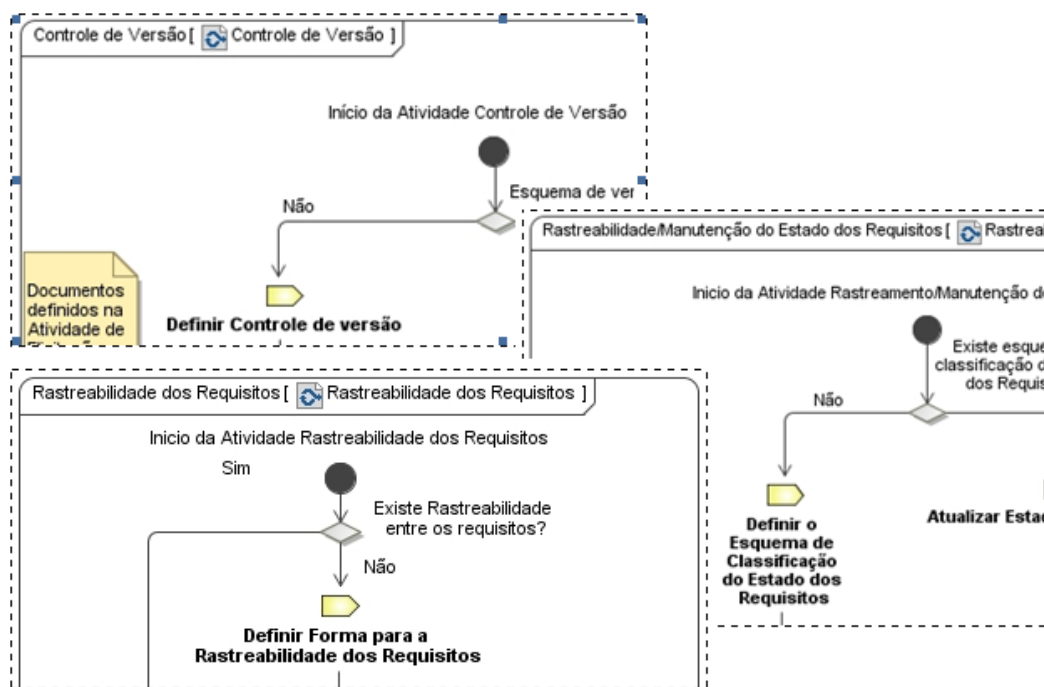


Figura 33 - Partes dos Diagramas de Fluxo de Trabalho das Atividades Rastreabilidade e Manutenção do Estado dos Requisitos, Controle de Versão e Rastreabilidade dos Requisitos.

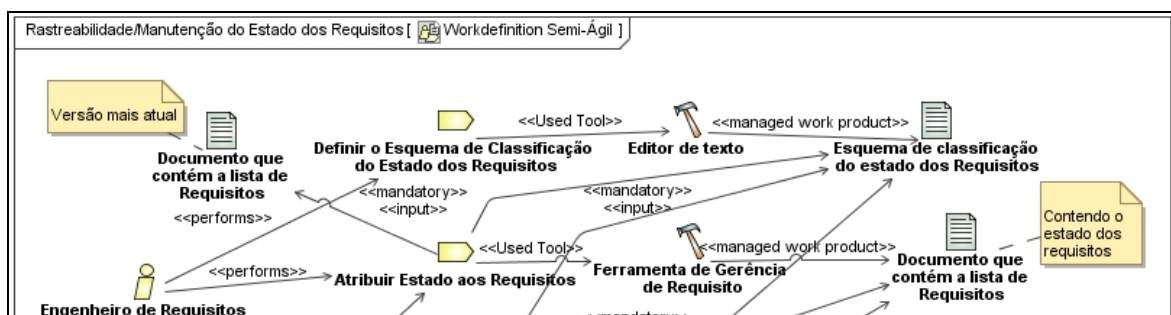


Figura 34 - Parte do Diagrama de detalhamento de atividade da Atividade *Rastreabilidade e Manutenção do Estado dos Requisitos*.

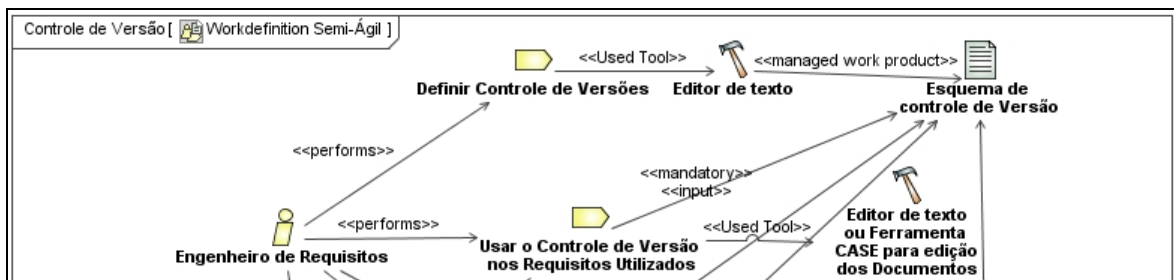


Figura 35 - Parte do Diagrama de detalhamento de atividade da Atividade Controle de Versão.

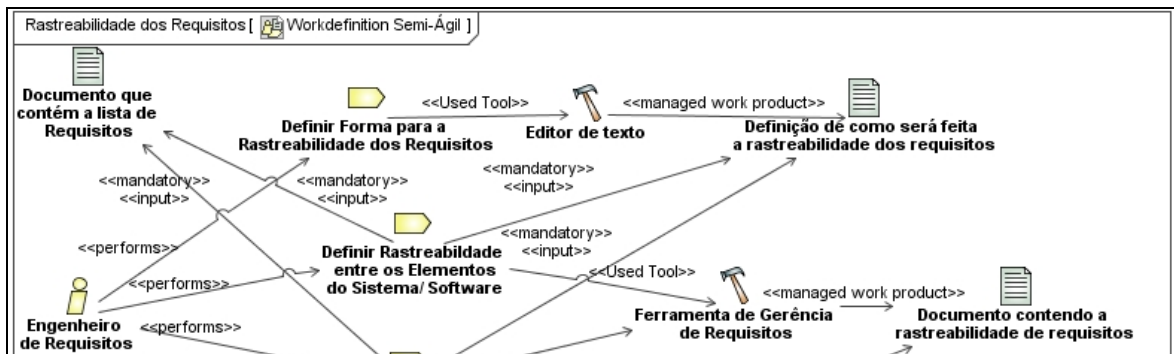


Figura 36 - Parte do Diagrama de detalhamento de atividade da Atividade Rastreabilidade dos Requisitos.

Foram estabelecidas metas para os problemas prioritários sendo elas (Etapa Medir):

a. Meta para Problema prioritário da Disciplina Desenvolvimento de Requisitos

A médio prazo analisar as metodologias listadas na dissertação, montar gabaritos dos documentos necessários para a execução de projetos através deles e criar um fluxo de artefatos para cada uma destas metodologias.

b. Meta para Problema prioritário da Disciplina Gerenciamento de Requisitos –

A médio prazo analisar os processos de seleção de ferramentas de ER existentes. A partir disso propor um processo para seleção dessas ferramentas e/ou elaborar uma lista das mesmas que contemplem as necessidades de migração de metodologias tradicionais para ágeis.

Foram determinadas as causas fundamentais do problema prioritário associado a cada uma das metas definidas na etapa anterior com o uso da ferramenta Diagrama de Causa e Efeito¹² (etapa Análise).

a. Causas para o problema prioritário da Disciplina Desenvolvimento de Requisitos (Figura 37):

¹² Diagrama de Causa e Efeito é utilizado para apresentar a relação entre o resultado de um processo (efeito) e os fatores (causas) que, por razões técnicas, possam afetar o resultado considerado [Werkema, 2004].

Causa A: inexistência de Gabaritos de Diagramas/Documentos para nos métodos de desenvolvimento de software.

Causa B: conhecimento inicial insuficiente para a definição dos Diagramas/Documentos que serão necessários para o método de desenvolvimento de software adotado para o projeto;

Causa C: definição insuficiente das necessidades do negócio;

Causa D: incertezas iniciais sobre a metodologia de desenvolvimento de software a ser empregado no projeto;

Causa E: conhecimento insuficiente sobre o processo;

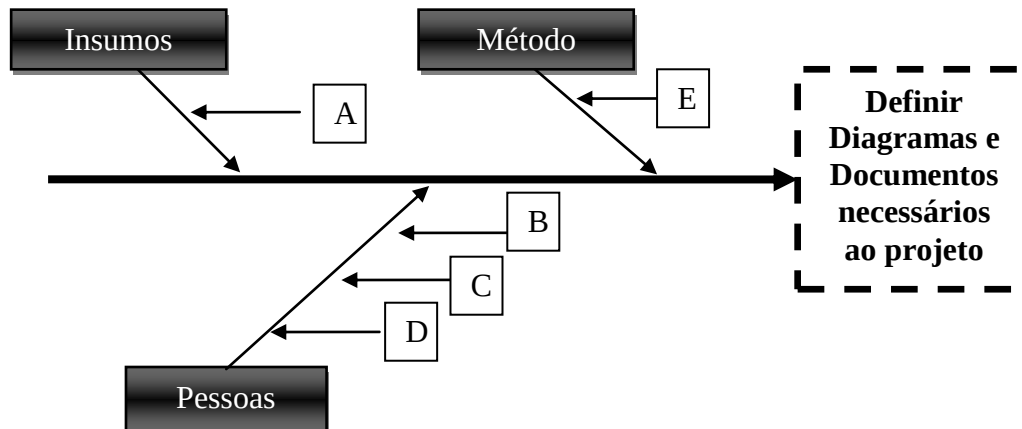


Figura 37 – Diagrama de Causa e Efeito – Disciplina Desenvolvimento de Requisitos.

b. Causas para o problema prioritário da Disciplina Gerenciamento de Requisitos (Figura 38):

Causa A: não vinculação de um guia para a seleção da(s) ferramenta(s) que será(ão) necessária(s) para o projeto.

Causa B: conhecimento inicial insuficiente para a escolha da(s) ferramenta(s) que será(ão) necessária(s) para o projeto;

Causa C: definição insuficiente das necessidades do negócio;

Causa D: incertezas iniciais sobre as funcionalidades necessárias das ferramentas de ER a ser empregada no projeto;

Causa E: conhecimento insuficiente sobre o processo;

Causa F: existência de uma grande quantidade de ferramenta de ER.

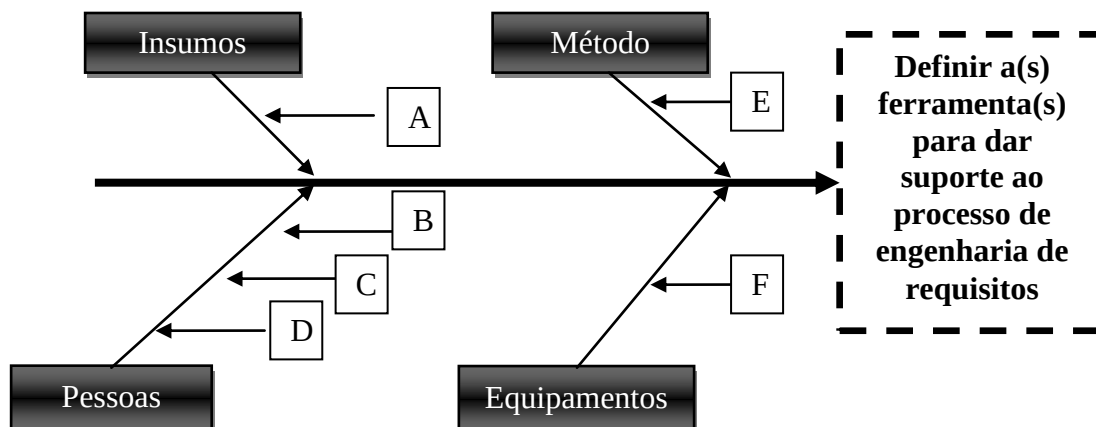


Figura 38 – Diagrama de Causa e Efeito – Disciplina Gerenciamento de Requisitos.

Por fim, foram propostas soluções potenciais para a eliminação das causas fundamentais do problema prioritário detectado em cada Disciplina do PERAF sendo elas (etapa Melhorar):

a. Soluções para as causas do problema prioritário da Disciplina Desenvolvimento de Requisitos:

Soluções para Causa A: a médio prazo, analisar as metodologias listadas na dissertação, montar gabaritos dos documentos necessários para a execução de projetos através deles e criar um fluxo de artefatos para cada umas destas metodologias.

Soluções para Causa B: a médio prazo, analisar as metodologias listadas na dissertação, montar gabaritos dos documentos necessários para a execução de projetos através deles e criar um fluxo de artefatos para cada umas destas metodologias.

Soluções para Causa C: adicionar o Cliente como ator responsável pela realização da Tarefa *Planejar a Elicitação*.

Soluções para Causa D: na primeira semana do projeto, definir com a equipe, responsável pelo mesmo, qual metodologia melhor se adéqua às necessidades do projeto.

Soluções para Causa E: Realizar treinamentos sobre o uso do processo.

b. Solução para as causas do problema prioritário da Disciplina Gerenciamento de Requisitos:

Soluções para Causa A: a médio prazo analisar os processos de seleção de ferramentas de ER existentes. A partir disso propor um processo para seleção dessas ferramentas e ou elaborar uma lista das mesmas que contemplem as necessidades de migração de metodologias tradicionais para ágeis.

Soluções para Causa B: acrescentar descrições sobre as funcionalidades das ferramentas à lista de ferramentas de ER proposta na solução anterior.

Soluções para Causa C: adicionar o Cliente como ator responsável pela realização da Tarefa *Planejar a Elicitação*.

Soluções para Causa D: fornecer uma lista contendo as funcionalidades das ferramentas de ER a serem analisadas.

Soluções para Causa E: realizar treinamentos sobre o uso do processo;

Soluções para Causa F: criar filtros para diminuir a gama de ferramentas a serem analisadas.

Em resumo, com a aplicação parcial do DMAIC foi possível detectar os problemas: *“Definir Diagramas e Documentos necessários ao projeto”* e *“Definir a(s) ferramenta(s) para dar suporte ao processo de engenharia de requisitos”*. Uma vez detectado os problemas foi possível propor as metas: *“A médio prazo analisar as metodologias listadas na dissertação, montar gabaritos dos documentos necessários para a execução de projetos através deles e criar um fluxo de artefatos para cada umas destas metodologias”* e *“A médio prazo analisar os processos de seleção de ferramentas de ER existentes. A partir disso propor um processo para seleção dessas ferramentas e ou elaborar uma lista das mesmas que contemplem as necessidades de migração de metodologias tradicionais para ágeis”* para a solução dos mesmos. Também foram levantadas as causas dos problemas anteriormente listados e propostas soluções para cada uma delas.

4.5. Considerações Finais

Neste capítulo foram apresentados dois estudos de casos reais desenvolvidos na UFLA, nos quais o PERAF foi aplicado sendo abordados dois cenários: i) o projeto EAD/ZEE-MG no qual o uso do PERAF o atendeu boa parte das necessidades de ER da organização que não tinha um processo de ER definido e formalizado e pretendia adotar um MA de desenvolvimento de software, e; ii) o projeto DES - Sistema de resíduos químicos no qual o uso do PERAF atendeu boa parte das necessidades de ER da organização que estava em processo de migração de um processo baseado em MT para um processo baseado em MA.

Para a avaliação os dois estudos de casos foram seguidas recomendações previstas na literatura. Inicialmente foram aplicados questionários que buscavam a viabilidade da aplicação do PERAF às organizações, sendo os cenários favoráveis. Seguiu-se a aplicação de um segundo questionários para detecção de discrepâncias entre o processo formal e a execução destes pelos usuários. Para a avaliação do processo também foi utilizado parcialmente o DMAIC. Como resultado detectou-se duas inconformidades, uma na

Disciplina *Desenvolvimento de Requisitos* e outra na Disciplina *Gerenciamento de Requisitos* do PERAF que foram considerados como prioritários e descritas de modo a garantir sua compreensão aos envolvidos. Além disso, foram definidas metas e propostas soluções para esses problemas. Por fim, foi aplicado um terceiro questionário, de pós experimento, com o objetivo de verificar o impacto do PERAF sobre as organizações as quais ele foi aplicado.

Em termos gerais, os métodos de avaliação demonstram que o PERAF atendeu a boa parte das necessidades de gerência de requisitos das organizações nos dois estudos de caso proporcionando um melhor planejamento das tarefas além de ter auxiliado no gerenciamento das mesmas. O PERAF também se mostrou flexível e adaptável à realidade dos projetos quanto a artefatos gerados e ao processo adotado.

Capítulo 5

Conclusão

5.1. Considerações Iniciais

Este trabalho, apresentou um estudo que pode ajudar a diminuir as barreiras técnicas e culturais enfrentadas por organizações que procuram migrar de métodos de desenvolvimento de software tradicionais (MT) para métodos ágeis (MA).

O trabalho propôs um conjunto de diretrizes para um Processo de Engenharia de Requisitos Adaptável e Flexível (PERAF), que auxilia nessa migração. O PERAF procura oferecer, também, suporte às necessidades de ER de organizações que ainda não têm um processo de ER definido e formalizado.

O nome inicial proposto para o conjunto de diretrizes era Processo de Engenharia de Requisitos Semi-Ágil (PERSÁ), no entanto esse nome foi alterado para PERAF. O motivo da mudança do nome foi o fato de que a agilidade ou semi-agilidade das diretrizes propostas não poder ser defendida de uma maneira plena, cabendo ainda discussões sobre o assunto e de não se poder dizer que um processo que adota o PERAF ser ágil ou tradicional, pois o PERAF apenas apresenta um conjunto de diretrizes para lidar com os requisitos.

Por permitir que atividades, tarefas ou artefatos sejam suprimidos do fluxo do processo, o PERAF não pode ser considerado nem ágil nem tradicional. Em comparação com outros métodos, o PERAF não é tão leve quanto o XP no qual as histórias de usuário são criadas e depois descartadas dando origem a testes, nem tão pesado quando o RUP¹³ onde existe uma dependência das atividades e artefatos em relação aos casos de uso, vitais para o processo. Assim, preferiu-se adotar o nome de processo de engenharia de requisitos

¹³ RUP: IBM Rational Unified Process (ou Processo Unificado Racional) [RUP, 2011].

adaptável e flexível em função de não haver uma demarcação quanto à agilidade subjacente às diretrizes.

5.2. Resultados

Como resultados deste trabalho o PERAF foi idealizado, modelado, aplicado e avaliado.

Com a avaliação do PERAF realizada no Capítulo 4 constatou-se que o PERAF atendeu boa parte das necessidades de ER da organização do Estudo de Caso 1 que não tinha um processo de ER definido e formalizado e pretendia adotar uma metodologia ágil de desenvolvimento de software, assim como também atendeu boa parte das necessidades de ER da organização do Estudo de Caso 2 que estava em processo de migração de uma MT para uma MA.

5.3. Contribuições

Este trabalho apresenta as seguintes contribuições nas áreas de Engenharia de Requisitos (ER) e de metodologias de desenvolvimento de software:

- A idealização, modelagem, aplicação e avaliação do Processo de Engenharia de Requisitos Adaptável e Flexível (PERAF).
- A realização de uma comparação entre a ER dos MT (Cascata, Incremental, RAD, Prototipagem e Espiral) e a ER dos MA(XP, Scrum, DSDM, FDD e ASD). Estas características foram melhor detalhadas na seção 2.4.3;
- A evidenciação de que a ER possa ser realizada de forma menos traumática em situação de adoção de uma MA ou de migração de MT para MA com a adequação do PERAF às necessidades da organização. Estas características foram melhor detalhadas nas seções 4.2 e 4.3;
- A evidenciação de que o PERAF se mostrou flexível e adaptável à realidade dos projetos dos estudos de caso quanto aos artefatos gerados e ao método adotado e que sua adoção proporcionou um melhor planejamento e gerenciamento das tarefas dos projetos. Estas características foram melhor detalhadas nas seções 4.2 e 4.3;

5.4. Problemas e Limitações do Trabalho

Na elaboração deste trabalho foi observado alguns problemas e limitações:

- Definição do escopo do projeto: a definição do escopo do projeto demorou demasiado tempo. O que resultou em limitações para a busca de estudos de caso mais abrangentes;
- Dificuldade no uso do SPEM: foi adotado o metamodelo SPEM para a modelagem do PERAF, no entanto o manual para seu uso foi considerado pouco informativo e de difícil compreensão;
- Dificuldade de encontrar locais para a realização dos estudos de caso: a necessidade de acompanhar um projeto envolvendo a aplicação do PERAF nas áreas de Desenvolvimento e de Gerenciamento de Requisitos culminou na limitação de tempo, pois, em grande maioria os projetos demandam mais tempo do que o que foi disponibilizado para a realização dos estudos de caso;
- Grande volume de dados na avaliação, para o qual foram adotados questionários de inspeção que objetivam analisar o que foi praticado nos estudos de caso e o que foi proposto no modelo formal. Isso gerou uma quantidade considerável de dados. Devido a esse motivo disponibilizamos apenas resultados da aplicação dos questionários durante a inspeção e durante a aplicação parcial do DMAIC, e;
- Foram feitos dois estudos de caso, esse número pode não traduzir certeza sobre a eficiência do PERAF, no entanto, pontos fracos que possam vir a surgir podem ser sanados com guias de boas práticas. Além desse guia também se apoiou na ideia de que todos os processos evoluem com o tempo e possíveis problemas poderão ser sanados com essa evolução.

5.5. Trabalhos Futuros

Como trabalhos futuros propõe-se as seguintes sugestões de pesquisa:

- Incorporar as alterações detectadas durante a aplicação da avaliação nas disciplinas *Desenvolvimento e Gerenciamento de Requisitos* e aplicar o PERAF a novos estudos de caso com o intuito de completar o ciclo DMAIC, uma vez que, durante

os trabalhos correspondentes a esta dissertação ficou pendente a etapa Controlar sobre as mudanças propostas;

- Verificar se a extensão das ideias do PERAF para as outras áreas do Desenvolvimento de Software podem trazer os benefícios que acreditamos ser possíveis na área de ER;
- Elaborar guias de boas práticas do PERAF com o intuito de sanar possíveis pontos fracos advindos de novas experiências de uso do PERAF, e;
- Desenvolver uma ferramenta de ER que dê suporte ao PERAF.

Referências Bibliográficas

- ABRAHAMSSON, P.; SALO, O.; RONKAINEN J.; WARSTA J. (2002): Agile Software Development Methods. Review and analysis. ESPOO 2002, VTT Publications 478.
- AMECHE, N. E KETABCHI, E. (2007). Design knowledge: A framework for requirement traceability. Poster presented at the Consortium for Computing Sciences in Colleges, 23rd Annual Eastern Conference 2007.
- ASHBY, M. (2000). How to write a paper. Engineering Department, University of Cambridge, Version 5, 38 p.
- BARNEY, M (2002). Motorola's second generation. Six Sigma, Forum Magazine, 1 (3) (2002), pp. 13–16.
- BECK, K. (1999). Extreme Programming Explained. First Edition, September 29, 1999; ISBN: 0201616416, 224 pages
- BECK, K. E FOWLER, M. (2000). Planning Extreme Programming. First Edition, October 12, 2000; ISBN: 0-201-71091-9, 160 pages.
- BECK, K. ANDRES, C. (2004) Extreme Programming Explained: Embrace Change (2nd Edition). 224 pages. Publisher: Addison Wesley Professional; 2 edition.
- BOEHM, B.; TURNER, R. (2005). Management Challenges to Implementing Agile Processes in Traditional Development Organizations. IEEE Software 22, 5 (September 2005), 30-39. DOI=10.1109/MS.2005.129 <http://dx.doi.org/10.1109/MS.2005.129>.
- CAO, L.; RAMESH B. (2008). Agile Requirements Engineering Practices: An Empirical Study. IEEE Software Published by the IEEE Computer Society.
- CHENG, BETTY H. C. E ATLEE, JOANNE M. (2007). Research Directions in Requirements Engineering. In *2007 Future of Software Engineering (FOSE '07)*. IEEE Computer Society, Washington, DC, USA, 285-303. DOI=10.1109/FOSE.2007.17 <http://dx.doi.org/10.1109/FOSE.2007.17>.
- COCKBURN, A. (2002). Agile Software Development. Boston, Addison-Wesley.

- CORAM, M. E BOHNER, S. (2005). The impact of agile methods on software project management. 12th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems (ECBS'05), 363-370. Visitado em dezembro 2010 from Web Site: <http://www.umuc.edu/library>.
- CURTIS, B.; KELLNER, M.; OVER, J. (1992). Process Modeling. Communications of the ACM, p.75-90, v.35, n.9, September, 1992. DOI=10.1145/130994.130998 <http://doi.acm.org/10.1145/130994.130998>.
- GOTEL, O. E MADER, P. (2009). How to Select a Requirements Management Tool: Initial Steps. Requirements Engineering, IEEE International Conference on, pp. 365-367, 2009 17th IEEE International Requirements Engineering Conference.
- DONZELLI, P. E IAZEOLLA, G. (2011). Hybrid simulation modeling of the software process. The Journal of Systems and Software 59. 227-235
- FADEL, A.C. E SILVEIRA, H.M. (2010): Metodologias ágeis no contexto de desenvolvimento de software: XP, Scrum e Lean, Faculdade de Tecnologia da Universidade Estadual de Campinas, 26p.
- FRANCO, E. F. (2007). Um modelo de gerenciamento de projetos baseado nas metodologias ágeis de desenvolvimento de software e nos princípios da produção enxuta. Escola Politécnica da Universidade de São Paulo (Dissertação de Mestrado).
- HIGHSMITH, J. (2002). Agile Software Development Ecosystems. Publisher: Addison Wesley; Pub Date: May 26, 2002; ISBN: 0-201-76043-6; Pages: 448.
- INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS - IEEE. (1990). IEEE Std 610.12-1990 - Standard glossary of software engineering terminology. Piscataway: IEEE, 1990.
- LAANTI, M; SALO, O; ABRAHAMSSON, P. (2011). Agile methods rapidly replacing traditional methods at Nokia: A survey of opinions on agile transformation, Information and Software Technology, Volume 53, Issue 3, March 2011, Pages 276-290, ISSN 0950-5849, DOI: 10.1016/j.infsof.2010.11.010. (<http://www.sciencedirect.com/science/article/B6V0B-51M609J-1/2/bab92eab1f4cc1d1ad9498e8f43d32c9>)
- LEFFINGWELL, DEAN E WIDRIG, DON. (2003). Managing Software Requirements: A Use Case Approach (2 ed.). Pearson Education.
- LUDUVIG, D. REINERT, J. D. (2007): Estudo do uso de Metodologias Ágeis no desenvolvimento de uma Aplicação de Governo Eletrônico, Universidade Federal de Santa Catarina -

Departamento de Informática e Estatística - Curso de Sistemas de Informação, 157p, Dissertação de mestrado.

MAGICDRAW (2011). <http://www.magicdraw.com/>. Acessado em Maio de 2011.

MANIFESTO ÁGIL. (2001). <http://agilemanifesto.org>; Acessado em Setembro de 2010.

MISRA, S.C.; KUMAR, U.; KUMAR, V.; GRANT, G. (2006). The Organizational Changes Required and the Challenges Involved in Adopting Agile Methodologies in Traditional Software Development Organizations. Digital Information Management, 2006 1st International Conference on, vol., no., pp.25-28, 6-6 Dec. 2006. Doi: 10.1109/ICDIM.2007.369325. URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4221862&isnumber=4221857>

NERUR, S.; MAHAPATRA, R.; AND MANGALARAJ G. (2005). Challenges of migrating to agile methodologies. Commun. ACM 48, 5 (May 2005), 72-78. DOI=10.1145/1060710.1060712 <http://doi.acm.org/10.1145/1060710.1060712>.

OMG - Object Management Group. (2005). Software Process Engineering Metamodel Specification, January, 2005.

PAETSCH, F., EBERLEIN, A., E MAURER, F. (2003). Requirements Engineering and Agile Software Development. In Proceedings of the Twelfth international Workshop on Enabling Technologies: infrastructure For Collaborative Enterprises (June 09 - 11, 2003). WETICE. IEEE Computer Society, Washington, DC, 308.

PAULA FILHO, Wilson de Pádua. (2009) Engenharia de Software: fundamentos, métodos e padrões. 3ª Edição. Rio de Janeiro: LTC. 6.

PETERSEN, K.; WOHLIN, C. (2010). The effect of moving from a plan-driven to an incremental software development approach with agile practices. Journal Name: Empirical Software Engineering. Cover Date: 2010-12-01. Publisher: Springer Netherlands. ISSN: 1382-3256. Subject: Computer Science. Page: 654-693. Volume: 15. Issue: 6. URL: <http://dx.doi.org/10.1007/s10664-010-9136-6>. Doi: 10.1007/s10664-010-9136-6@article

PFLEEGER, SHARI LAWRENCE. (2001). Software Engineering: theory and practice 2nd ed. ISBN: 1-13-029049-1.

PORTELA, C. S. (2009). Uma proposta de gerenciamento ágil dos projetos de desenvolvimento de software do CTIC / UFPA, Instituto de Ciências Exatas e Naturais – Faculdade de Computação – Universidade Federal do Pará; 94p, Dissertação de mestrado.

- PRESSMAN, ROGER S. (1995). Engenharia de Software; tradução José Carlos Barbosa dos Santos; revisão técnica José Carlos Maldonado, Paulo Cesar Masiero, Rosely Sanches. - São Paulo: Makron Books, 1995.
- PRESSMAN, R. S. (2010). Engenharia de Software. 6ª ed. McGraw-Hill.
- QURESHI, M. R, J.; HUSSAIN, S. A. (2008). An adaptive software development process model; *Advances in Engineering Software* 39, 654-658
- RAMESH, B., CAO, L.; BASKERVILLE, R. (2010), Agile requirements engineering practices and challenges: an empirical study. *Information Systems Journal*, 20: 449–480. Doi: 10.1111/j.1365-2575.2007.00259.x.
- RATIONAL SOFTWARE CORPORATION. (2002). Rational Unified Process – RUP. Versão 2002.05.00.
- REED, K.; DAMIANI, E.; GIANINI, G. E COLOMBO, A. (2004). Agile management of uncertain requirements via generalizations: a case study. QUTE-SWAP 04: Proceedings of the 2004 workshop on Quantitative techniques for software agile process, pp. 40–45. DOI Bookmark: Use this link to bookmark this Article <http://doi.acm.org/10.1145/1151433.1151439>.
- RUP (2011). <http://www-01.ibm.com/software/awdtools/rup/>. Acessado Outubro de 2011.
- SCHWABER, KEN. (2004) Agile Project Management with Scrum, Microsoft Press, 2004.
- SCHWABER, K. E BEEDLE M. (2001) Agile development witch Scrum, Prentice Hall, 2001.
- SILLITTI, A.; CESCHI, M.; RUSSO, B.; SUCCI, G. (2005) "Managing uncertainty in requirements: a survey in documentation-driven and agile companies," *Software Metrics*, 2005. 11th IEEE International Symposium, vol., no., pp.10 pp.-17, 1-1 Sept. 2005 Doi: 10.1109/METRICS. 2005.29. URL: <http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1509295&isnumber=32322>
- SOMMERVILLE (2007). Engenharia de Software - 8ª Edição – Pearson, 552p.
- SOUNDARARAJAN, S. E ARTHUR, J.D. (2009) Engineering of Computer Based Systems, 2009. ECBS 2009. 16th Annual IEEE International Conference and Workshop on the 14-16 April 2009 Page(s):187 – 195 Digital Object Identifier 10.1109/ECBS.2009.21

- SPEM (2008). Software & Systems Process Engineering Metamodel specification. Versão 2.0
Release data: Abril de 2008. Disponível em: <http://www.omg.org/spec/SPEM/2.0/>. Acessado em: 25 setembro de 2010.
- STAPLETON, J. (1997). Dynamic systems development method – The method in practice. Addison Wesley.
- SWEBOK (2004) - Guide to the Software Engineering Body of Knowledge, A project of the IEEE Computer Society Professional Practices Committee, 2004. Disponível em: <<http://www.computer.org/portal/web/swebok>>. Acessado em: 06/12/2010.
- TOYBEE, J. A. (ED.) (2004). Palavras mágicas sobre os sonhos. Editora: Vergara & Riba ISBN: 8587213849.
- TURK, D., FRANCE, R. E RUMPE, B. (2005). Assumptions underlying agile software development process, Journal of Database Management, 16, 62-87.
- WERKEMA, MARIA CRISTINA CATARINO. (2004). Criando a cultura seis sigma. 3.ed. BELO HORIZONTE: Werkema, 2004. 253p.
- WIEGERS, K. E. (2003). Software Requirements (2 ed.). Microsoft Press, Redmond, WA, USA.