UNIVERSIDADE DO SUL DE SANTA CATARINA

ADRIANA MARIA ZENCKE

RASTREABILIDADE DE REQUISITOS:

Utilizando a ferramenta *CASE Enterprise Architect*

Palhoça

2010

ADRIANA MARIA ZENCKE

RASTREABILIDADE DE REQUISITOS:

Utilizando a Ferramenta *CASE Enterprise Architect*

Trabalho de Conclusão de Curso apresentado ao Curso de Ciência da Computação da Universidade do Sul de Santa Catarina, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação.

Orientador: Prof. Osmar de Oliveira Braz Júnior

Palhoça

2010

ADRIANA MARIA ZENCKE

RASTREABILIDADE DE REQUISITOS:

Utilizando a Ferramenta *CASE Enterprise Architect*

Trabalho de Conclusão de Curso apresentado ao Curso de Ciência da Computação da Universidade do Sul de Santa Catarina, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação.

Palhoça, 16 de junho de 2010.

_____

Prof. e orientador Osmar de Oliveira Braz Junior, Msc.
Universidade do Sul de Santa Catarina

_____

Prof. Aran Bey Tcholakian Morales, Dr.
Universidade do Sul de Santa Catarina

_____

Prof. Fernanda Oviedo Bizarro, Esp.
Universidade do Sul de Santa Catarina

A minha família, sem a qual eu jamais chegaria até aqui.

AGRADECIMENTOS

A Deus, por todas as oportunidades de aprendizado, nas derrotas e nas vitórias.

A minha família, pai e mãe amores incondicionais, incentivadores e apoiadores na conquista deste sonho. Luciana, irmã querida que tanto amo e em especial a minha irmã Meri, exemplo de força, garra e dedicação em todos os momentos, alegres e tristes.

Ao meu orientador, Professor. Msc. Osmar de Oliveira Braz Junior, pelas recomendações, dedicação, paciência e acompanhamento integral deste trabalho científico. A Coordenadora dos Trabalhos de Conclusão de Curso, Profa. Dra. Maria Inés Castiñeira, pelo carinho e pelas as orientações prestadas durante o desenvolvimento deste projeto.

Às sinceras amizades que fiz durante estes quatro anos e meio de intensivo trabalho. Em especial, por ordem alfabética, a: André Luis Leal Junior, Carolina Poersch, Daniel Kohn, Felipe Ferreira, Giullyan Kuntze, Gustavo Comeli, Johannes Ferreira, Juliana Martins, Kély Maria Góes, Lucas Vianna, Michel Onesti, Paloma Mott, Priscila Avancine, dentre outros, que mesmo sem saber me fortaleceram nos momentos de tristeza e saudade, sendo diariamente meus companheiros de trabalho e/ou diversão.

A André Ricardo Bento por sua imensa paciência, carinho e atenção.

Aos funcionários do Hemosc do setor CPQ e TI. Em especial a Cida e Monica pelas informações fornecidas e atenção em todos os encontros.

A todos que, direta ou indiretamente, apoiaram durante toda essa trajetória.

“Procure ser uma pessoa de valor, em vez de procurar ser uma pessoa de sucesso. O sucesso é apenas a conseqüência disto” (Albert Einstein).

RESUMO

A rastreabilidade de requisitos auxilia para identificar o impacto da mudança de requisitos. Este trabalho propõe a definição de um processo, baseado na customização do RUP, para realizar a rastreabilidade dos requisitos desde o início do projeto, com o auxilio da ferramenta CASE *Enterprise Architect*. A escolha e customização do RUP, como processo de desenvolvimento de software, destaca como e quais artefatos devem ser gerados na execução do projeto. Nas fases seguintes são definidos documentos a serem rastreados para então ter a visibilidade do sistema desde os requisitos ao código fonte. Após a definição do processo foi desenvolvido um sistema protótipo para validar o processo, que identificou a rastreabilidade como um processo simples de ser implantado, satisfazendo a necessidade idealizada.

Palavras-chave: Rastreabilidade. Gerência de Requisitos. Ferramenta CASE.

ABSTRACT

The traceability of requirements gives assistance to identify the impact of modification of requirements. This work proposes the definition of a process, based on the customization of RUP, to realize the traceability of requirements since the beginning of the project, with the assistance of the Enterprise Architect CASE tool. The choice of customization of RUP, as a software development process, it highlights how and which artfacts should be generated on the project execution. On the following phases the documents to be traced are defined, and then to have system's visibility of the requirements to the source code. After the definition of the process is developed a prototype system to validate the process, that identified the traceability as a simple process to be implemented, satisfying the idealized necessity.

Keywords: Traceability, Requirements Management. CASE tool.

LISTA DE ILUSTRAÇÕES

Figura 1 - Fluxograma das etapas metodológicas. 21
Figura 2 - Proposta da Solução. 23
Figura 3 - Ciclo de vida do *software.* 27
Figura 4 - Arquitetura Geral do RUP. 29
Figura 5 - Análise e Projeto. 29
Figura 6 - Fases de Planejamento. 30
Figura 7 - Análise de Requisitos 31
Figura 8 - Modelo de gerenciamento de Requisitos no *Enterprise Architect.* 34
Figura 9 - Engenharia de Requisitos 36
Figura 10 - Processo de Engenharia de Requisitos 37
Figura 11 - Processo de levantamento e análise de requisitos... **Erro! Indicador não definido.**
Figura 12 - Matriz de Rastreabilidade. 46
Figura 13 – As fases e os marcos de um projeto. 52
Figura 14 - Fase de Concepção 53
Figura 15 - Fase de Elaboração. 54
Figura 16 - Fase de Construção. 55
Figura 17 - Fase de Transição 56
Figura 18 - Visão de Requisitos. 58
Figura 19 - Módulo de Inserção de Comunicado. 58
Figura 20 - Módulo de Controle de Usuários. 59
Figura 21 - Atores do Sistema 60
Figura 22 - Modelo de Caso de Uso. 61
Figura 23 - Requisitos do Caso de Uso 04 e seus respectivos cenários. 61
Figura 24 - Cenários do Caso de Uso 04. 62
Figura 25 - Realização do Caso de Uso 01. 63
Figura 26 - Diagrama de Robustez. 64
Figura 27 - Diagrama de Sequencia do RCSU 01. 65
Figura 28 - Diagrama de pacotes das camadas do sistema. 66
Figura 29 - Diagrama de Classe *Abstract Factory* na visão global do sistema. 67
Figura 30 - Modelo de Classe de Domínio. 68

Figura 31 - Modelo de Dados. 69
Figura 32 - Visão de Componentes. 70
Figura 33 - Topologia do Servidor. 71
Figura 34 - Topologia do Cliente. 72
Figura 35 - Ambiente de desenvolvimento. 73
Figura 36 - Matriz de Rastreabilidade entre Requisitos Funcionais e Regras de Negócio. 75
Figura 37 - Matriz de Rastreabilidade entre Caso de Uso e Requisito Funcional. 76
Figura 38 - Matriz de Rastreabilidade entre Colaboração e Caso de Uso. 76
Figura 39 - Matriz de Rastreabilidade entre Colaboração e Objeto. 77
Figura 40 – Arquitetura da Camada de Controle ligando RCSU01 e CtrlComunicado. 77
Figura 41 - Matriz de Rastreabilidade entre Objeto e classe. 78
Figura 42 – Diagrama hierárquico do Requisito Funcional RF01 ao código fonte. 79
Figura 43 - Hierarquia do código fonte ao Requisito Funcional. 80
Figura 44 - Rastreabilidade do Requisito RF15 81

LISTA DE GRÁFICOS

Quadro 1 - Cenário atual de desenvolvimento de software. 18
Quadro 2 - Tipos de Rastreabilidade. 45

SUMÁRIO

1 INTRODUÇÃO **15**
1.1 PROBLEMA: COMO FACILITAR O GERENCIAMENTO E O RELACIONAMENTO ENTRE ARTEFATOS DA MODELAGEM COM O CÓDIGO FONTE? 16
1.2 OBJETIVOS 17
1.2.1 Objetivo Geral **17**
1.2.1.1 Objetivos Específicos 17
1.3 JUSTIFICATIVA 18
1.4 ESTRUTURA DO PROJETO 19
2 MÉTODOS **20**
2.1 CARACTERÍSTICAS DO TIPO DE PESQUISA 20
2.1.1 Pesquisa Bibliográfica **20**
2.2 ETAPAS METODOLÓGICAS 21
2.2.1 Revisão Bibliográfica **22**
2.2.2 Definir o processo de desenvolvimento **22**
2.2.3 Implementação **22**
2.2.4 Conclusão dos resultados obtidos **22**
2.3 PROPOSTA DE SOLUÇÃO 23
2.4 DELIMITAÇÃO 23
3 ENGENHARIA DE SOFTWARE **24**
3.1 DESENVOLVIMENTO DE SOFTWARE 25
3.1.1 **Ciclo de Vida do *Software*** **26**
3.1.2 Processo de desenvolvimento **28**
3.1.3 Ferramentas CASE **32**
3.2 ENGENHARIA DE REQUISITOS 34
3.2.1 Processo de Engenharia de Requisitos **36**
3.2.1.1 Produção de Requisitos 37
3.2.1.1.1 Elicitação *37*
3.2.1.1.2 Registro *38*
3.2.1.1.3 Verificação *39*

3.2.1.1.4 Validação *39*
3.2.1.2 Gerência de Requisitos 40
3.2.1.2.1 Controle de Mudanças *40*
3.2.1.2.2 Gerência de Configuração *41*
3.2.1.2.3 Rastreabilidade *42*
4 RASTREABILIDADE 44
4.1 MATRIZ DE RASTREABILIDADE 44
4.2 MÉTODOS DE RECUPERAÇÃO AUTOMÁTICA 46
4.3 MODELOS E CARACTERÍSTICAS DOS MÉTODOS DE RASTREABILIDADE ENTRE OS REQUISITOS E O CÓDIGO DE SOFTWARE 47
4.4 CENÁRIOS 48
4.5 GRANULARIDADE DA RASTREABILIDADE 49
5 DEFINIÇÃO DO PROCESSO DE SOFTWARE 51
5.1 FASES 52
5.1.1 Concepção ..... Erro! Indicador não definido.
5.1.2 Elaboração 53
5.1.3 Construção 55
5.1.4 Transição 56
6 MODELAGEM DO SISTEMA 57
6.1 VISÃO DE REQUISITOS 57
6.2 VISÃO DE CASO DE USO 59
6.2.1 Atores 60
6.2.2 Modelo de Caso de Uso 60
6.3 VISÃO DE ANÁLISE 62
6.3.1 Realização 63
6.4 VISÃO DE PROJETO 65
6.4.1 DAO 66
6.5 VISÃO DE DADOS 67
6.5.1 Modelo de Classe de Domínio 67
6.5.2 Modelo de Dados 68
6.6 VISÃO DE COMPONENTES 70
6.7 VISÃO DE DISTRIBUIÇÃO 70
6.7.1 Topologia do Servidor 71
6.7.2 Topologia do Cliente 71

6.7.3 Ambiente de Desenvolvimento 72
7 ANÁLISE DOS DADOS 74
7.1 MATRIZ DE RASTREABILIDADE 74
7.2 RASTREABILIDADE VERTICAL 78
7.3 ALTERAÇÃO DE REQUISITOS 80
8 CONCLUSÕES E TRABALHOS FUTUROS 82
8.1 CONCLUSÕES 82
8.2 TRABALHOS FUTUROS 83
REFERÊNCIAS 84
APÊNDICE 88
APÊNDICE A – VISÃO DO NEGÓCIO 89
APÊNDICE B – GLOSSÁRIO DE NEGÓCIO 92

1 INTRODUÇÃO

A engenharia de software é conseqüência do processo de engenharia de sistemas (PRESSMAN, 2006). Este, por sua vez, concentra-se não somente no software, mas em diversos elementos, analisando, projetando, organizando em um sistema para a transformação da informação ou do controle.

Para Sommerville (2005), a engenharia de sistema se divide em **padrões de processos** (antes do desenvolvimento do sistema), que podem incluir definições de especificações, processos de projetos e validação, e uma descrição dos documentos que devem ser gerados no curso desses processos e **padrões de produtos** (para software em desenvolvimento), incluindo padrões de documentos, como a estrutura do documento de requisitos, padrões de documentação e padrões de codificação.

Os padrões do processo se alinham com as normas propostas pela IEEE 98, que contempla a atividade levantamento de requisitos, os padrões de produtos se alinham com a idéia de padronização de código (FABRI, 2009).

No desenvolvimento de software, nada é constante, e sim mutável. As regras mudam a toda hora, manutenções são feitas em funcionalidades já entregues. (SOUZA, 2009).

Souza (2009) completa que é muito problemático quando uma determinada informação do negócio muda ou uma nova funcionalidade passa a existir, pois isso acaba acarretando várias alterações em cascata no desenvolvimento do software.

Para amenizar esse problema, é necessário ter uma maneira de mapear e manter registrado todos os requisitos e regras de negócio do cliente para que, no futuro, possa-se ter uma rastreabilidade de impacto no caso de alteração de um requisito ou de uma regra de negócio.

O termo rastreabilidade é usado para descrever referência ao grupo coletivo de requisitos baseados em relacionamento, fazendo uso de relacionamento sobre requisitos, projeto e implementação, visando a qualidade e estabelecendo mecanismos para avaliar o impacto de mudanças no sistema. (GENVIGIR E VIJAYKUMAR, 2008).

Rastreabilidade de requisitos é definida por Gotel e outros (apud LEAL, 2007) como sendo a habilidade de descrever e seguir a vida de um requisito nos dois sentidos, de avanço e retorno.

A IEEE (1997) define que todo requisito deve ser rastreado durante o processo de software. A rastreabilidade é essencial para ser capaz de determinar o conjunto completo de requisitos que podem ser afetados por uma alteração no código e documentos de projeto.

A analise de impacto pode ser facilitado caso a rastreabilidade entre requisitos e os demais produtos de trabalho possa ser identificada. (COSTA et al., 2009).

Segundo Costa e outros (2009), existem muitas dificuldades na implantação na pratica de uma estratégia para estabelecer a rastreabilidade dos requisitos. Para reduzir o esforço e estabelecer uma interligação algumas ferramentas CASE (*Computer-Aided Software Engineering*) dão suporte a essas necessidades como, por exemplo, a atualização dos artefatos no momento da criação.

1.1 PROBLEMA: Como facilitar o gerenciamento e o relacionamento entre artefatos da modelagem com o código fonte?

A área da engenharia de *software*, que trata dos requisitos de software, é conhecida como Engenharia de Requisitos. Esta envolve o processo de descoberta dos requisitos, de identificação dos envolvidos e suas necessidades e de documentação, de forma que seja útil para a análise, comunicação e a sua implementação. (NUSEIBEH, 2000).

O gerenciamento é uma atividade da engenharia de software que tem como objetivo controlar a agregação e a evolução dos requisitos do projeto, através das atividades de Rastreabilidade de Requisitos (GENVIGIR E VIJAYKUMAR, 2008).

O mesmo autor completa com o exemplo do Departamento de Defesa Norte-Americano que utiliza 4% de seu orçamento em Tecnologia de Informação (TI) com rastreabilidade. Complementa a afirmação, informando que o custo com rastreabilidade, para 2008, seria de 1,5 bilhões de dólares.

Atualmente, já existem diversos métodos especializados em rastreabilidade com o código de software, que podem ser agregados em categorias bases, por exemplo, métodos baseado em roteiros (*scenario*), métodos baseados em modelos, métodos baseados em recuperação de informação *(information retrieval*), métodos baseados em semântica (CLELAND-HUANG et al., apud LEAL, 2007).

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Implementar a rastreabilidade num processo de desenvolvimento de software através da ferramenta CASE *Enterprise Architect*, indentificando, no sistema, o impacto da alteração ou inclusão de novos requisitos funcionais.

1.2.1.1 Objetivos Específicos

Os objetivos específicos deste trabalho são:

a) identificar os locais de alteração com a alteração ou inclusão de um novo requisito funcional;
b) obter uma documentação do software interligada, dos requisitos ao código;
c) analisar quais documentações serão necessárias no processo de desenvolvimento e como devem ser feitas para rastrear;
d) identificar o que deve ser rastreado no sistema;
e) centralizar todos os processos de desenvolvimento em uma única ferramenta *CASE*;
f) implementação de um protótipo de software para a arquitetura apresentada.

1.3 JUSTIFICATIVA

Espíndola e outros (2004) estudos demonstram que uma grande quantidade de projetos de software são cancelados ou fracassam por não atenderem completamente as necessidades dos clientes e excederem o prazo e o orçamento estimados. Diversos trabalhos apontam deficiências nos requisitos dos sistemas como uma das principais causas de fracassos em projetos de software (SOMMERVILLE, 1997). Tais constatações têm levado alguns autores a considerar a Engenharia de Requisitos como uma das mais importantes disciplinas da Engenharia de Software (ESPINDOLA et al., 2004).

Diversos são os desafios encontrados na manutenção de sistemas. Dentre estes desafios, a realização da Engenharia de Requisitos destaca-se como sendo uma área particularmente afetada pelas dificuldades envolvidas em projetos de manutenção (ESPÍNDOLA et al., 2004).

O custo do desenvolvimento das atividades iniciais é baixo. Por outro lado, é nessa fase que grande parte dos defeitos são inseridos, portanto, o custo de correção é baixo. Entretanto, se não tratados, neste momento, pode aumentar bastante o custo do projeto, ou seja, na manutenção. Conforme destacada no Quadro 1.

	% do Custo de Desenvolvimento	% dos erros introduzidos	% dos erros encontrados	Custo relativo de correção
Análise de Requisitos	5	55	18	1
Projeto	25	30	10	1 – 1.5
Códificação e teste de unidade	50			
Teste	10	10	50	1 – 5
Validação e Documentação	10			
Manutenção		5	22	10 – 100

Quadro 1 - Cenário atual de desenvolvimento de software.

Fonte: ÁVILA, Ana L.; SPÍNDOLA, Rodrigo O.; Introdução a Engenharia de Requisitos. Rio de Janeiro: Engenharia de Software Magazine, 2007, n 01, p. 47.

Atualmente, é muito difícil gerenciar um conjunto de requisitos de tal forma que seja possível visualizar em que esses são atendidos (COSTA et al., 2009). Ter rastreabilidade e saber relacionar os requisitos é uma tarefa importante do engenheiro de software.

O uso adequado da rastreabilidade pode facilitar uma série de atividades do processo de desenvolvimento, podendo, também, auxiliar na melhoria da qualidade, tanto para o processo quanto para o produto de software. (GENVIGIR E VIJAYKUMAR, 2008)

Após uma modificação, necessita-se de novos testes em várias áreas do sistema, garantindo que a mudança foi implementada corretamente. Não menos importante, a alteração da documentação deve estar completamente condizente com o que foi implementado. (AQUINO, 2009)

Essa rastreabilidade é de total importância, pois com ela pode-se determinar, por exemplo, a complexidade da alteração, os artefatos (documentos) que necessitarão ser alterados e o prazo da alteração do software (SOUZA, 2009).

1.4 ESTRUTURA DO PROJETO

O projeto compõe-se da seguinte divisão:

Capítulo 1 – Neste capítulo, é apresentado o problema, os objetivos gerais e específicos, a justificativa e as delimitações do mesmo.

Capítulo 2 – Neste capítulo, são apresentados os conceitos ligados a Rastreabilidade de *Software*, metodologia de desenvolvimento com a ferramenta case *Enterprise Architect*.

Capítulo 3 – Definição do processo utilizado para a rastreabilidade.

Capítulo 4 – Documentação UML desenvolvida para auxiliar na rastreabilidade em conseqüência do processo definido.

Capítulo 5 – São apresentados os resultados que foram obtidos, apresentam-se as conclusões obtidas, juntamente, com sugestões de trabalhos futuros.

Referências – Neste capítulo, são apresentadas as referências utilizadas na realização deste trabalho.

2 MÉTODOS

Neste tópico, será apresentado o processo de desenvolvimento do projeto, destacando a forma para alcançar o objetivo deste trabalho.

2.1 CARACTERÍSTICAS DO TIPO DE PESQUISA

A classificação da pesquisa, em Pesquisa Bibliográfica, do ponto de vista dos procedimentos técnicos, é elaborada a partir de material já publicado, constituído, principalmente, de livros, de artigos de periódicos e, atualmente com material disponibilizado na internet. (GIL, 1991).

2.1.1 Pesquisa Bibliográfica

Este projeto tem a característica de uma pesquisa bibliográfica, pois, para a elaboração, foram realizadas pesquisas de fontes diversas de informações escritas para conhecer as diferentes contribuições científicas disponíveis e, assim, coletar dados gerais ou específicos a respeito de um assunto.

A pesquisa abrangeu publicações impressas e digitais, incluindo base de dados, em forma de livros, periódicos, resenhas, monografias, dicionários, enciclopédias, slides, dissertações, teses, apostilas, etc.

Assim, todo este estudo dá suporte para a identificação do problema, a definição dos objetivos, a elaboração da justificativa, a criação de um modelo e a elaboração de todo o projeto.

2.2 ETAPAS METODOLÓGICAS

Neste trabalho foram identificadas algumas etapas para execução do projeto: Revisão Bibliográfica, pesquisa de ferramentas Case, definição do processo de rastreabilidade e identificar os modelos a serem utilizados no processo de rastreabilidade, implementação e a conclusão sobre os resultados obtidos, conforme destacado na Figura 1.

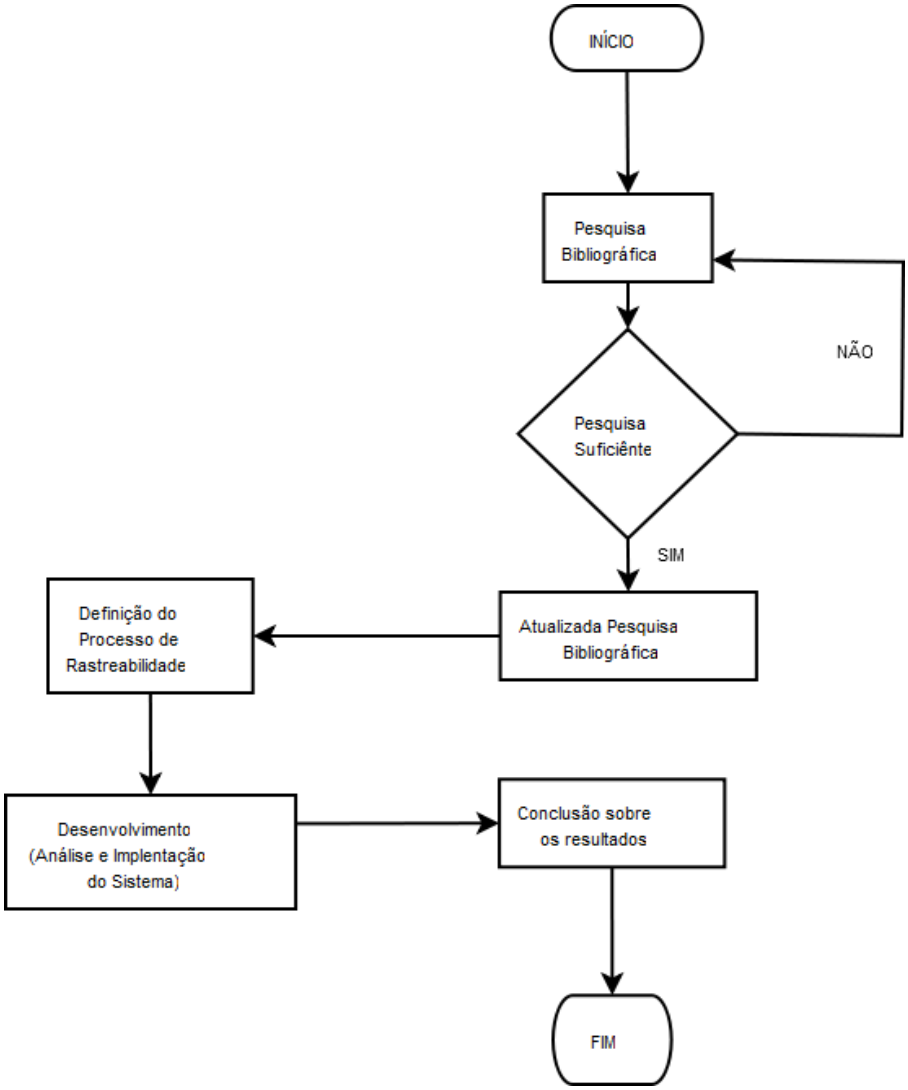


Figura 1 - Fluxograma das etapas metodológicas.
Fonte: Elaborado pelo Autor.

2.2.1 Revisão Bibliográfica

Na etapa de Revisão Bibliográfica são apresentados os conceitos abordados no trabalho, entre eles está a Engenharia de Software, Engenharia de Requisitos, Gerenciamento de Requisitos, Rastreabilidade, Ferramenta CASE e outros.

2.2.2 Definir o processo de desenvolvimento

Esta etapa apresenta uma proposta do processo de desenvolvimento, identificando os artefatos rastreados e que matrizes construídas.

2.2.3 Implementação

Após os conceitos teóricos e a definição do processo de desenvolvimento, um *software*, seguindo o processo definido na etapa anterior com a utilização da ferramenta CASE *Enterprise Architect*, será desenvolvido para apresentação da proposta do projeto, exibindo os artefatos gerados pela rastreabilidade.

2.2.4 Conclusão dos resultados obtidos

Na etapa de implementação, será possível identificar a rastreabilidade nos diferentes artefatos usados no desenvolvimento de softwares. Nessa etapa, serão apresentados os resultados identificados nas etapas anteriores.

2.3 PROPOSTA DE SOLUÇÃO

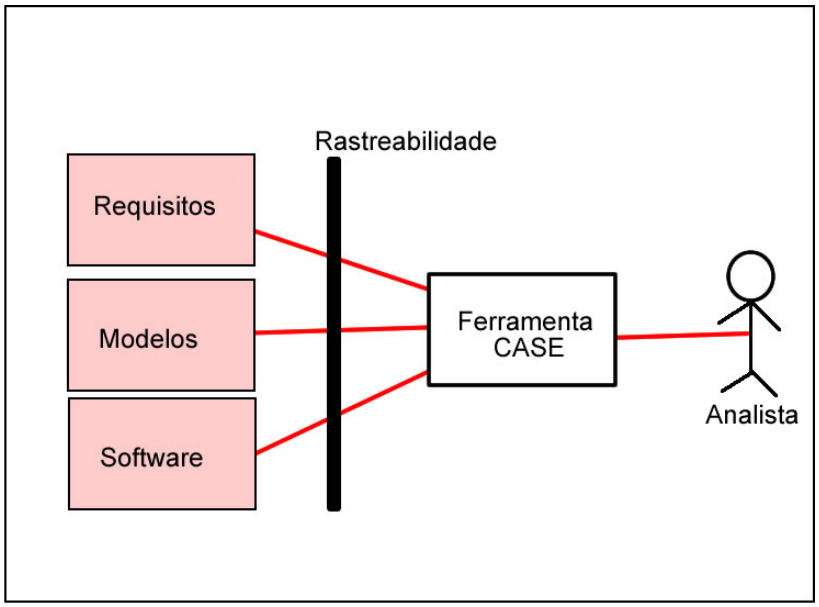


Figura 2 - Proposta da Solução.
Fonte: Elaborado pelo Autor.

2.4 DELIMITAÇÃO

Os tópicos, abaixo, não serão apresentados por não se tratar do escopo deste projeto, conforme segue:

a) rastreabilidade de requisitos não funcionais;
b) controle de revisão dos requisitos;
c) definição de métrica a partir da rastreabilidade, sobre o impacto gerado no sistema;
d) apresentação dos conceitos de Orientação Objeto, UML e Linguagem de Programação;
e) comparação entre ferramentas CASE.

3 ENGENHARIA DE SOFTWARE

A Engenharia de *Software* atinge os objetivos de "[...] produzir *software* com alta qualidade [...]" (PRESSMAN, 2006) e complementando, a afirmação, Paula (2007), e com alta produtividade, quanto é praticada por profissionais treinados e bem informados, utilizando tecnologias adequadas, dentro de processos que tirem proveito tanto da criatividade quando da racionalização do trabalho.

A Engenharia de *Software* é voltada para produção de *software* de qualidade, conforme Pressman (2006):

> A engenharia de software abrange um conjunto de três elementos fundamentais – métodos, ferramentas e procedimentos – que possibilitam ao gerente o controle do processo de desenvolvimento do *software* e oferece ao profissional uma base para a construção de software com alta qualidade produtivamente.

No objetivo de produção de softwares de qualidade, conforme citado anteriormente, o objetivo da engenharia de requisitos é melhorar a modelagem de sistemas e a capacidade de analisá-los, possibilitando maior entendimento de suas características antes da implementação (ROBINSON et al. 2003 apud SILVA, 2006).

Devido à natureza volátil dos requisitos, é necessário estar preparado para mudanças, no sentido da melhoria contínua garantindo a qualidade do software e atendendo as expectativas dos requisitantes, sofrendo alterações decorrentes de erros/omissões ou para atender novas necessidades (LEITE, 1997). O mesmo autor (2001) afirma: "Acreditamos que, ao atacar a definição e gerência de requisitos, estaremos colaborando de maneira fundamental para a qualidade geral do software".

Silva (2006) esclarece que a engenharia de requisitos tem abordado o princípio de separação de características, com o intuito de prover facilidades nos processos, como por exemplo, na rastreabilidade de software. Na Seção 2.1.1, foi resumido o processo de desenvolvimento do software, na seção 2.1.2 é apresentado a engenharia de requisitos e seus processos e na seção 2.1.2.3, os conceitos de gerenciamento de requisitos direcionando para o processo de rastreabilidade de *software*.

3.1 DESENVOLVIMENTO DE SOFTWARE

Software é um termo erroneamente associado a programas de computadores. *Software* não é apenas programa, mas também toda a documentação associada e os dados de configuração para auxiliar que este programa funcione corretamente. Sistema de Software são vários softwares separados, arquivos de configuração, documentação para descrever a estrutura e usuários (SOMMERVILLE, 2003).

Varias são as definições de engenharia de software, mas a definição proposta na primeira conferência sobre o assunto, NATO *Conference on Software Enginnering* em 1968, é utilizada até os dias de hoje (NAUR e RANDELL, 1968, apud PRESSMAN, 2006):

"Engenharia de software é a criação e a utilização de sólidos princípios de engenharia a fim de obter softwares econômicos que sejam confiáveis e que trabalhem eficientemente em maquinas reais".

Engenharia de software é responsável por todos os aspectos da produção de *software*, desde os estágios de especificação do sistema até a manutenção desse sistema, ou seja, depois que ele entrou em operação (SOMMERVILLE, 2003) .

"A engenharia de software é a aplicação sistemática, disciplinada, quantificável no desenvolvimento, operação e manutenção de software" (IEEE 610, 1990; tradução livre). Pressman (2006) completa o conceito com a afirmação, "[...] que, além de disciplinada, necessitamos adaptabilidade e agilidade".

A engenharia de *software* propõe modelos para o desenvolvimento de software, que são escolhidos pelo que melhor se adapte às necessidades e recursos. Os modelos, assim como os padrões de projetos (*design patterns*), são concretizados na prática, e não na teoria. Outros dizem que os modelos são descobertos, e não inventados (MORAGA, 2008).

A engenharia de software pode ser explicada com algumas características, por exemplo, dominar a complexidade: em geral, os problemas não podem ser pesquisados em sua totalidade. Uma técnica é forçar a dividir o problema em partes de tal forma que cada parte possa ser aproveitada, logo, a comunicação entre as partes é simples. (VLIET, 2007).

O alicerce da engenharia de software é o processo. (PRESSMAN, 2006). O processo é um conjunto de atividades e resultados que geram um produto de software. (SOMMERVILLE, 2003).

Algumas são as características para a utilização da Engenharia de *Software*, tais como: a eficiência com que o software é desenvolvido, o custo e tempo de desenvolvimento.

O custo total e o tempo de desenvolvimento de projetos de software são elevados. Isso também é válido para a manutenção de software. (AVILA e SPÍNDOLA, 2007).

Para que o software não se torne obsoleto, rapidamente, tem que evoluir com a realidade que está sendo modelado. Isto significa que os custos são incorridos após a entrega do software sistema e que temos que suportar essa evolução em mente durante o desenvolvimento.

Regular cooperação entre pessoas é uma parte integrante de programas desenvolvidos em larga escala. A disciplina é uma das chaves para a conclusão bem sucedida de um software projeto de desenvolvimento.

O objetivo da Engenharia de Software é produzir software de qualidade. Leite (2001) afirma:

> Produzir software de qualidade é uma meta básica da Engenharia de Software, que disponibiliza métodos, técnicas e ferramentas para este fim. Muito tem-se escrito sobre qualidade e seus vários adjetivos, no entanto, o fundamental é que o software seja confiável, isto é seja eficaz e siga os padrões exigidos pelo contexto onde irá atuar.

O cenário atual de desenvolvimento de software está distante do que foi idealizado pela Engenharia de Software, isto se deve a alguns motivos, entre eles, estão a não utilização dos fundamentos da Engenharia de Software no processo e/ou a má utilização desses fundamentos. (ÁVILA e SPÍNDOLA, 2007).

3.1.1 Ciclo de Vida do *Software*

Para Paula (2007), se um programa interessar a outras pessoas e, assim, passar a ter valor, aparecerão demandas para melhorar suas qualidades, aumentar suas funções, prolongar sua vida. Ela ainda destaca o seguinte ciclo de vida:

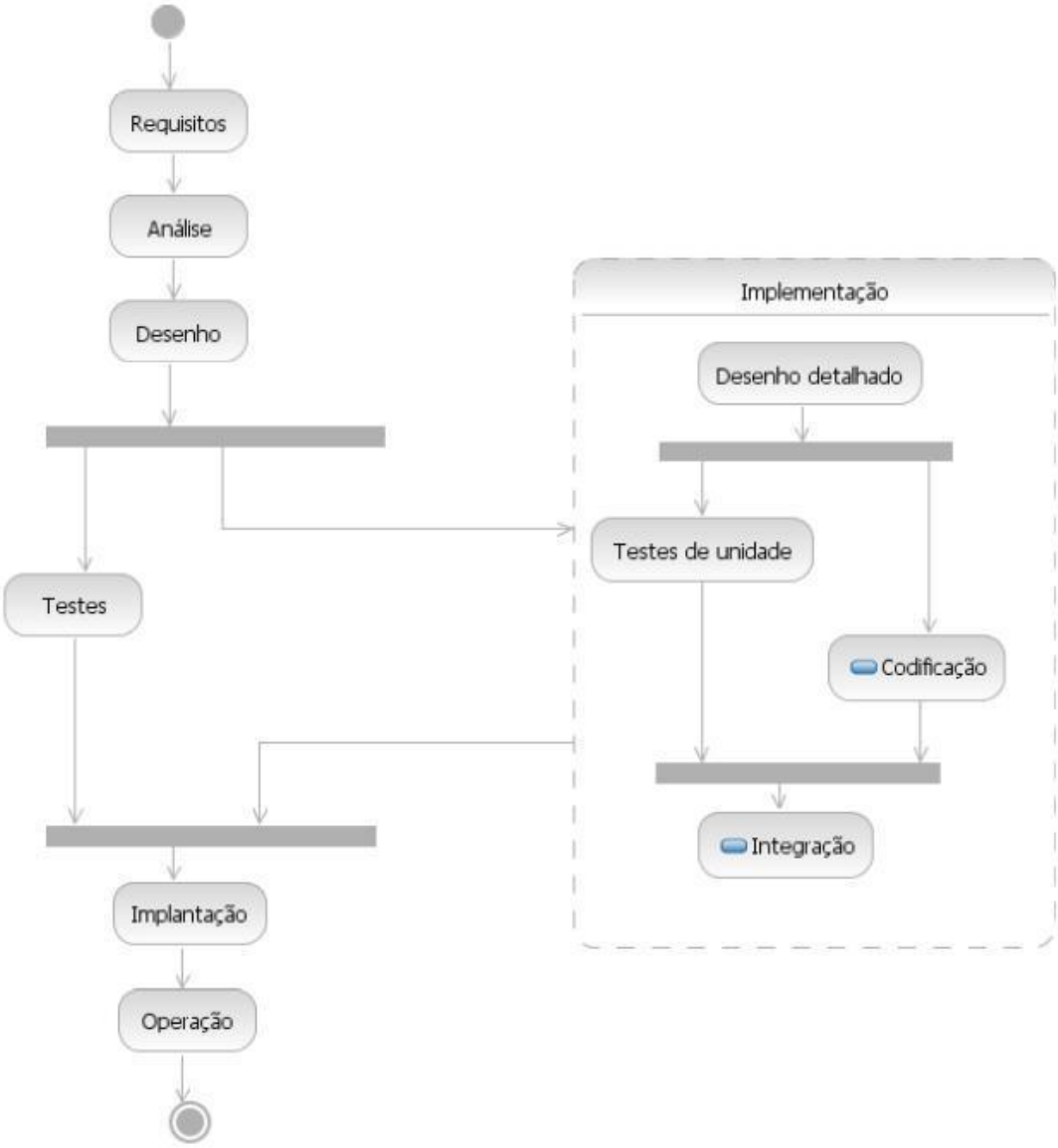


Figura 3 - Ciclo de vida do *software*.

Fonte: PAULA, Wilson; **Modelo UML do ciclo de vida do software**. Engenharia de Software *Magazine*: Rio de Janeiro: 2007. p. 06.

A Figura 3 é representada por um diagrama de atividade, que destaca o início do ciclo com o círculo preenchido e o final da atividade com o círculo com uma circunferência. As setas correspondem a transições entre as atividades.

3.1.2 Processo de desenvolvimento

Para Rational Software (2009), é um processo que define "Quem" está fazendo "O Que" e "Como" para atingir certa meta. Portanto, o processo de desenvolvimento de *software* é um conjunto de atividades e resultados associados que têm por objetivo produzir software eficiente, de alta qualidade, com baixa taxa de erros e que atenda às necessidades e expectativas do usuário de forma geral. (JACOBSON apud HUSITA, 2003).

Vidal (2003) destaca alguns métodos utilizados no processo de desenvolvimento, são eles: ASD, RUP, dX, PSP, DSDM, Scrum, XP, Iconix, Catalysis, TSP.

No escopo deste projeto, será utilizado o método RUP (*Rational Unified Process*). Seguem abaixo algumas definições de acordo com Kroll e Kruchten (2003):

> a) o RUP é uma maneira de desenvolvimento de software que é iterativa, centrada à arquitetura e guiada por casos de uso.
> b) o RUP é um processo de engenharia de software bem definido e bem estruturado. Ele define claramente quem é responsável pelo que, como as coisas devem ser feitas e quando fazê-las. O RUP também provê uma estrutura bem definida para o ciclo de vida de um projeto, articulando claramente os marcos essenciais e pontos de decisão;
> c) o RUP é também um produto de processo que oferece uma estrutura de processo customizável para a engenharia de software.

Para Boente e outros (2008), o RUP utiliza a linguagem de modelagem unificada para especificar, modelar e documentar artefatos. Por ser flexível e configurável, ele pode ser utilizado em projetos de pequeno, médio e grande porte. Na Figura 4, está apresentada a arquitetura geral do RUP.

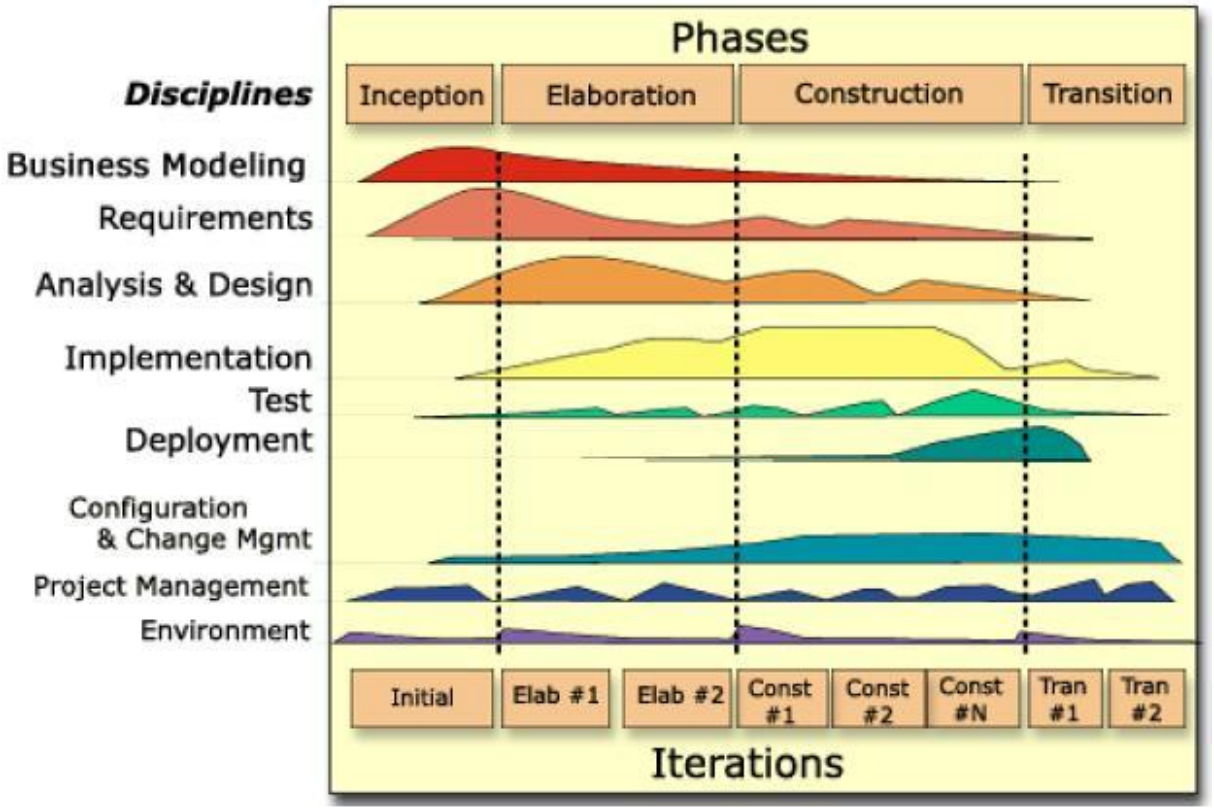


Figura 4 - Arquitetura Geral do RUP.

Fonte: BOENTE, Alfredo P; OLIVEIRA, Fabiano G.; ALVEZ, João N.; **RUP como Metodologia de Desenvolvimento de Software para Obtenção da Qualidade de Software**. In: SIMPÓSIO DE EXCELÊNCIA EM GESTÃO E TECNOLOGIA - SEGeT, 2008, Resende: Ed. AEDB, 2008, p. 10.

A arquitetura geral do RUP, apresentada na Figura 4, possui duas dimensões: Aspecto dinâmico do processo (expresso em termos de fases, disciplinas e marcos) e termos de componentes, disciplinas, atividades, fluxo de trabalho, artefato e papéis de processo. (BOENTE et al., 2008).

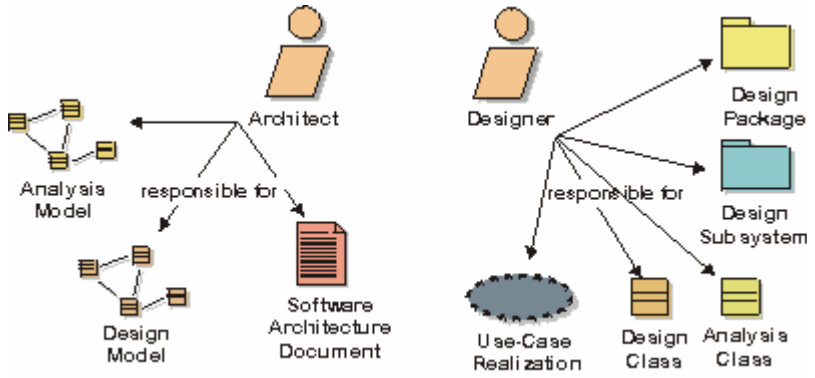


Figura 5 - Análise e Projeto.

Fonte: VIDAL, Raul; FARIA, João; AGUIAR, Ademar; GONÇALVES, Gil**; Processos de Desenvolvimento de Software**; Universidade do Porto Faculdade de Engenharia – FEUP/LEIC – Laboratório de Gestão de Projetos: 2003-04

A *Rational* destaca suas fases de planejamento, como: Iniciação, Elaboração, Construção e Transição.

a) iniciação: a meta dominante da fase de iniciação é atingir o consenso entre todos os envolvidos sobre os objetivos do ciclo de vida do projeto;

b) elaboração: a meta da fase de elaboração é criar a *baseline* para a arquitetura do sistema a fim de fornecer uma base estável para o esforço da fase de construção;

c) construção: a meta da fase de construção é esclarecer os requisitos restantes e concluir o desenvolvimento do sistema com base na arquitetura da *baseline;*

d) transição: o foco da Fase de Transição é assegurar que o software esteja disponível para seus usuários finais.

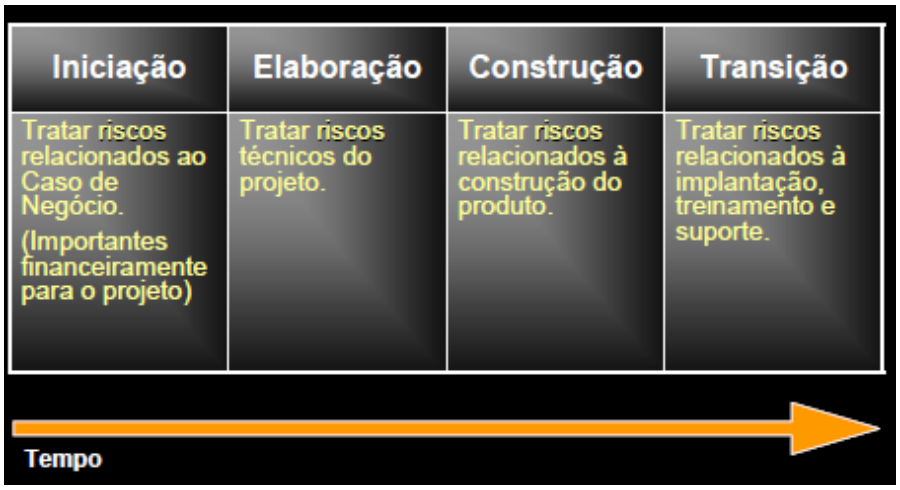


Figura 6 - Fases de Planejamento.
Fonte: ALENCAR, Ivan Z.; **Gerenciamento de Requisitos com Casos de Uso**: Melhores práticas da Engenharia de *Software*; *Rational Software*: 2006, p. 39.

A disciplina mostra todas as atividades que devem ser realizadas para produzir um determinado conjunto de artefatos. Seguem, abaixo, e com a descrição das principais finalidades:

a) modelagem de negócios: descreve como desenvolver uma visão da nova organização-alvo e, com base nessa visão, definir os processos, os papéis e as responsabilidades dessa organização em um modelo de casos de uso de negócios e em um modelo de objetos de negócios;

b) requisitos: dentre as principais metas dessa atividade estão: oferecer aos desenvolvedores do sistema uma compreensão melhor dos requisitos do sistema, definindo as fronteiras do sistema, fornecer uma base para planejar o conteúdo técnico das iterações, fornecer uma base para estimar o custo e o

tempo de desenvolvimento do sistema e definir uma interface de usuário para o sistema, focando as necessidades e metas dos usuários;

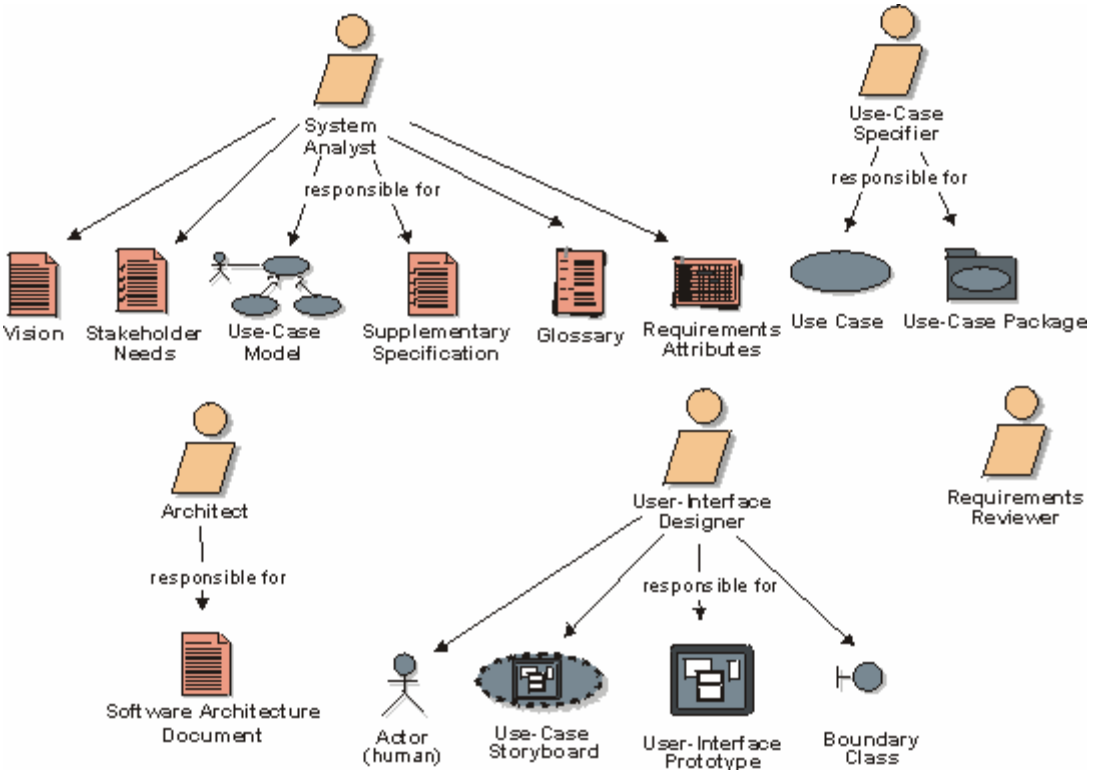


Figura 7 - Análise de Requisitos.

Fonte: VIDAL, Raul; FARIA, João; AGUIAR, Ademar; GONÇALVES, Gil**; Processos de Desenvolvimento de Software**; Universidade do Porto Faculdade de Engenharia – FEUP/LEIC – Laboratório de Gestão de Projetos: 2003-04

c) análise e design: esta disciplina transforma os requisitos em um design do sistema a ser criado, desenvolve uma arquitetura sofisticada para o sistema e adapta o design para que corresponda ao ambiente de implementação, projetando-o para fins de desempenho;

d) implementação: tem como finalidade a definição da organização do código em termos de subsistemas de implementação organizados em camadas, implementa classes e objetos em termos de componentes, testa os componentes desenvolvidos como unidades e integra os resultados produzidos por implementadores ao sistema executável;

e) teste: a disciplina teste atua em vários aspectos como uma provedora de serviços para as outras disciplinas. O teste enfatiza, principalmente, a avaliação da qualidade do produto;

f) implantação: descreve os modos de implantação de produto. São eles: a instalação personalizada, o produto em uma forma "compacta" e o acesso ao software por meio da Internet;

g) ambiente: concentra-se nas atividades necessárias à configuração do processo para um projeto. Ela descreve as atividades para o desenvolvimento das diretrizes de suporte de um projeto;

h) gerenciamento de projeto: é a arte de confrontar os objetivos da concorrência, gerenciar riscos e superar obstáculos para liberar com êxito um produto que atenda às necessidades dos clientes e dos usuários;

i) gerenciamento de configuração e mudança: envolve a identificação dos itens de configuração, a restrição de mudanças nesses itens, envolve a auditoria das mudanças feitas nesses itens e a definição e o gerenciamento das configurações desses itens.

3.1.3 Ferramentas CASE

Pressman (2006) define o conceito de CASE - *Computer-Aided Software Enginerring (*Engenharia de *Software* Auxiliada por Computador) - como a chegada do primeiro par de sapatos, que os engenheiros de software ganharam. Seguindo a analogia do sapato, considera que ainda não existem muitas variedades, não apresentam tanta sofisticação para quem acompanha a moda e não combina com outras peças que os desenvolvedores usam. Mas constituem uma peça essencial para os guarda-roupas do desenvolvedor de *software* e, ao passar do tempo, são mais adaptáveis, mais usáveis às necessidades individuais de profissionais.

O mesmo autor descreve a classificação de ferramentas CASE de acordo com a função, por seus papéis como instrumento para gerentes e pessoal técnico, pelo uso nas várias etapas do processo de engenharia de software, pela arquitetura do ambiente e ainda pelo custo e origem. No escopo deste trabalho, a ferramenta CASE auxiliará no gerenciamento do projeto e rastreamento de requisitos.

Alguma das finalidades do uso de ferramenta CASE, no gerenciamento de projeto, segundo Pressman (2006), são úteis estimativas de esforço, custo, duração de um

projeto de software, definir uma estrutura de divisão de trabalho, planejar uma programação viável de projeto e acompanhar projetos em base contínua. Além disso, pode ampliar o uso para compilar métricas, indicando a produtividade e a qualidade do produto.

Para os gerentes que desejam contratar desenvolvimento de software existem algumas ferramentas CASE para rastrear os requisitos, desde a origem da proposta até o trabalho do desenvolvedor de software, que implementarão o requisito no sistema. (PRESSMAN, 2006).

O mesmo autor identifica problemas para grandes sistemas, que não atendem os requisitos especificados pelo cliente. Logo, o objetivo das ferramentas de rastreamento de requisitos é oferecer uma abordagem sistemática ao isolamento dos requisitos, que se inicia com a especificação do cliente. O trabalho do desenvolvimento subsequente pode sofrer um referência cruzada com o banco de dados para que a conformidade aos requisitos seja mais provável.

> Ferramentas não são balas de prata. Eles não removem o fato de ter que decidir sobre uma estratégia de rastreabilidade, e não podem confirmar se os seus artefatos estão corretos e completos. Mas ferramentas de gerenciamento de requisitos, tais como o IBM *Rational* RequisitePro, permitem: Estabelecer e manter a rastreabilidade facilmente em todo o seu conjunto de requisitos definidos em uma maneira discreta; Aplicar a estratégia definida para a rastreabilidade; Apoio a propagação de uma mudança através da sua hierarquia de requisitos, identificando os requisitos que podem ser afetadas por essa mudança com base na rastreabilidade definida; Relatórios e consultas sobre a rastreabilidade. Estas características do RequisitePro, portanto, reduz o esforço manual e aumenta a confiabilidade da sua rastreabilidade. (BEHRENS, 2007; tradução livre)

Behrens (2007) destacou a importância da utilização de ferramentas, nesse processo de rastreabilidade de software, caracterizando, principalmente, a aplicação da estratégia, redução do esforço manual e o aumento da confiabilidade da rastreabilidade.

A ferramenta CASE utilizada neste trabalho será o *Enterprise Architect* desenvolvida pela *Sparx Systems*. A Ferramenta utiliza a notação UML - *Unified Modeling Language* - Modelo de Linguagem Unificado. Um exemplo desta ferramenta segue na Figura 8 - Modelo de gerenciamento de Requisitos no *Enterprise Architect*.

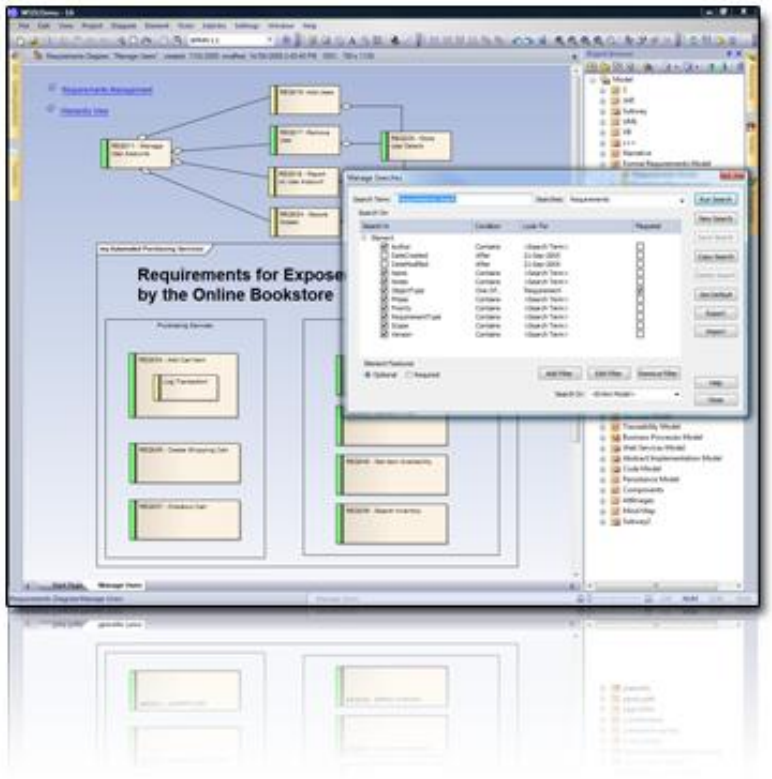


Figura 8 - Modelo de gerenciamento de Requisitos no *Enterprise Architect.*
Fonte: **SPARTX SYSTEMS**; 2009.

3.2 ENGENHARIA DE REQUISITOS

A Engenharia de Requisitos (ER) é área da Engenharia de Software que estuda o desenvolvimento e a manutenção de requisitos que o sistema deverá satisfazer e tem como objetivo identificar, comunicar e ajustar as expectativas do Cliente. (VLIET, 2007).

O mesmo autor completa dizendo: “A Engenharia de requisitos é um processo cíclico que envolve quatro atividades: Elicitação, especificação, validação e negociação”.

De acordo com IEEE (1997, tradução livre), a definição para requisito seria:

a) condição ou capacidade que um usuário necessita para resolver um problema ou alcançar um objetivo;
b) condição ou capacidade que deve ser atendida ou estar presente em um sistema ou componente de sistema para satisfazer um contrato, padrão, especificação ou outros documentos formalmente impostos;
c) representação documentada de uma condição ou capacidade como indicado em 1 ou 2.

Concluindo, requisitos são as necessidades detalhadas pelo cliente que deverão ser atendidas com o *software*. Ávila e Spíndola (2007) ressaltam a importância dessa definição, pois, mesmo definido pelo cliente, a capacidade pode não ser o que ele ou o negócio necessitam. Cabe, então, à equipe identificar as necessidades do negócio junto às informações fornecidas.

Os requisitos desempenham um papel central no desenvolvimento de sistemas e, portanto, precisam ser mantidos. Para alterar "o design de um produto", por exemplo, sem saber aonde refletirá a mudança, pode acarretar enormes problemas nas fases posteriores de desenvolvimento. Para tentar resolver e/ou diminuir este problema, entra a engenharia de requisitos, ligado fortemente com a gestão da mudança. (ÁVILA e SPÍNDOLA, 2007).

Aspectos de qualidade, quando tratados ainda que no processo de definição dos requisitos, seria a solução para os problemas identificados nas mudanças dos requisitos, que ficam escondidos, normalmente, até a etapa de programação. (LEITE, 2001).

Os requisitos de sistemas são, normalmente, classificados em funcionais e não funcionais. Os requisitos de *software*, segundo Sommerville (2005), podem ser expressos como requisitos de usuário e de sistema, e diferenciados por requisitos funcionais e requisitos não funcionais. Para ele, os requisitos de sistema estão classificados em:

a) requisitos funcionais: são declarações de funções que o sistema deve fornecer, em alguns casos, pode descrever o que o sistema não deve fazer;
b) requisitos não funcionais: são restrições sobre funções e serviços fornecidos pelo sistema;
c) requisitos de domínio: são requisitos que se originam do domínio de aplicação do sistema, podendo ser requisitos funcionais e não funcionais.

Engenharia de requisitos é o termo usado para descrever as atividades relacionadas à produção (levantamento, registro, validação e verificação) e gerência (controle de mudanças, gerência de configuração, rastreabilidade, gerência de qualidade dos requisitos) de requisitos. (ÁVILA e SPÍNDOLA, 2007).

Figura 9 - Engenharia de Requisitos.

Fonte: ÁVILA, Ana L.; SPÍNOLA, Rodrigo O.; **Introdução a Engenharia de Requisitos**. Rio de Janeiro: Engenharia de Software *Magazine*, 2007, n 01, p. 47.

A Figura 9 apresenta de forma clara e objetiva, conforme Ávila e Spíndola (2007), as atividades da Engenharia de Requisitos. A Atividade de Gerencia de Requisitos atua durante todo o processo de Produção de Requisitos.

3.2.1 Processo de Engenharia de Requisitos

É importante que a empresa tenha um processo de ER bem definido. Cada empresa deve criar sua metodologia de acordo com sua realidade. (ÁVILA e SPÍNDOLA, 2007). Segundo o padrão IEEE 830, devemos considerar alguns critérios de qualidade ao trabalharmos com requisitos: correção, ambiguidade, completude, consistência, verificabilidade e modificabilidade. (IEEE, 1997).

O processo genérico, que descreve a maioria dos processos, está nos próximos itens. A Figura 10 apresenta um modelo genérico de atividades que pode descrever a maioria dos processos de engenharia de requisitos. Esse modelo concentra as atividades na produção dos requisitos, mas o gerenciamento dos requisitos que está em paralelo, na figura, participa de todas as atividades de produção.

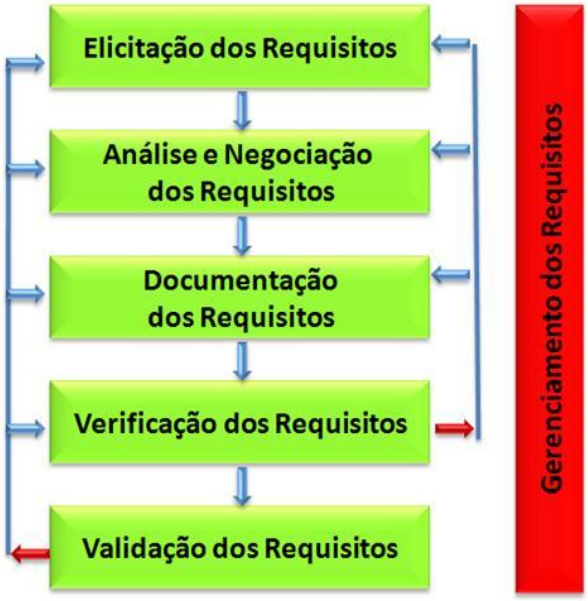


Figura 10 - Processo de Engenharia de Requisitos.

Fonte: ÁVILA, Ana L.; SPÍNDOLA, Rodrigo O.; **Introdução a Engenharia de Requisitos**. Rio de Janeiro: Engenharia de Software *Magazine*, 2007, n 01, p. 49.

3.2.1.1 Produção de Requisitos

A produção de requisitos é o processo de descobrir, analisar, documentar e verificar as funções e restrições do *software*. Nessa seção, apresentaremos as quatro atividades bases na produção de requisitos.

3.2.1.1.1 Elicitação

Para Espíndola e outros (2004), é nessa fase que são identificadas as expectativas e necessidades dos *stakeholders* (cliente) com relação ao *software* a ser desenvolvido. Levantamento de requisitos é o entendimento do problema, sendo que o analista não é um mero observador, ele pode ter que tomar uma posição em uma disputa de poder ou decidir entre as necessidades conflitantes. (ÁVILA e SPÍNDOLA, 2007; VLIET, 2007).

A cada fase do ciclo de vida do software, será produzido um documento contendo uma representação distinta do *software* a ser construído. Cada um desses documentos representa o *software* em um determinado nível de abstração. Com o passar do processo, o nível dos documentos é mais detalhado, sendo a última representação o código fonte do sistema.

Considerada a base para as demais atividades, a especificação de requisitos é fundamental para o sucesso do projeto. Uma especificação de requisitos bem elaborada é pré-requisito para um software de qualidade, mas não existe uma garantia disso. (ÁVILA e SPÍNDOLA, 2007).

Na caracterização dos requisitos funcionais do sistema, Sommerville (2005) observa que a especificação de requisitos deve ser completa e consistente. Para ele, completa, significa que todas as funções requeridas pelo usuário devem ser definidas. A consistência significa que o requisito não deve ter definições contraditórias.

Existem algumas técnicas que apóiam as atividades de levantamento de requisitos. Entre elas estão Entrevista e Prototipação.

O protótipo tem por objetivo explorar aspectos críticos dos requisitos de um produto, implementando de forma rápida um pequeno subconjunto de funcionalidades deste produto. (MORAES, 2007). A prototipagem auxilia na descoberta de requisitos ocultos, ou seja, requisitos que não são identificados em um primeiro momento pelo cliente ou analista.

A entrevista é o método mais simples e o mais tradicional de ser efetuado para esse processo, produzindo bons resultados na elaboração dos requisitos. Um ponto chave, em uma entrevista, seria a escolha do cliente ‘certo’ que, conhecendo o processo, identifique os reais problemas a serem resolvidos e, assim, facilitando o entendimento dos requisitos. Uma escolha ‘errada’ do cliente desencadeará uma série de soluções que não correspondem a sua real necessidade. Moraes (2007) sugere, para que não haja dispersão no foco da reunião, ter um plano de entrevista e definir o tempo para término.

3.2.1.1.2 Registro

Uma vez identificados e negociados, os requisitos devem ser documentados para que possam servir de base para o restante do processo de desenvolvimento.

Espíndola e outros (2004) definem que, nessa atividade, os requisitos são utilizados como base para análise dos requisitos (atividade anterior). A análise distribui os requisitos em categorias, explora as relações entre eles e classifica a importância de cada um dos requisitos de acordo com as necessidades dos *stakeholders*. Os requisitos são negociados para decidir quais devem ser aceitos, de forma a obter consenso.

Entre os muitos problemas que enfrentamos na documentação de requisitos, certamente, administrar o grande volume de informações, gerado pelo processo de requisitos, é um dos principais.

O registro dos requisitos, num documento próprio, facilita o controle de alterações de todos os envolvidos na manutenção dos requisitos, bem como a geração de versões do documento e a facilidade de acesso por todos os envolvidos.

3.2.1.1.3 Verificação

A verificação examina a especificação do software, de forma a assegurar que todos os requisitos foram definidos sem ambiguidades, inconsistências ou omissões, detectando e corrigindo possíveis problemas ainda durante a fase de definição dos requisitos. (ÁVILA e SPÍNDOLA, 2007).

Os mesmos autores acreditam que o custo da correção de defeitos aumenta na medida que o processo de desenvolvimento progride. Para eles, as revisões de artefatos de software têm se mostrado uma abordagem eficiente e de baixo custo para encontrar defeitos logo após terem sido introduzidos, reduzindo o retrabalho e melhorando a qualidade dos produtos.

“Além da análise de erros e omissões, o processo de definição de requisitos possibilita a análise sob diferentes perspectivas, tais como, viabilidade, custo, tempo, prioridades, reuso, completude, corretude, variabilidade, evolução, dentre outras.” (SILVA, 2006).

3.2.1.1.4 Validação

A validação examina a especificação do software, assegurando se todos os requisitos foram definidos sem ambigüidades, inconsistências ou omissões e que todos os erros foram detectados e corrigidos. (ESPÍNDOLA et al., 2004).

Existem semelhanças entre a atividade de análise e validação de requisitos, conforme apresenta Sommerville (2005), pois ambos se preocupam em descobrir problemas em requisitos. A diferença entre eles é que, na validação deve se elaborar um esboço completo do documento de requisitos e na análise trabalha com os requisitos incompletos.

Essa atividade significa aprovar junto ao cliente os requisitos que foram especificados. Embora aparentemente simples, essa atividade pode ser bastante dificultada pelo cliente ou mesmo por um processo de validação inadequado utilizado pela empresa. (ÁVILA e SPÍNDOLA, 2007).

3.2.1.2 Gerência de Requisitos

Segundo Leite (2001), o fator da falta de definição de escopo, o fato dos requisitos serem por natureza volátil, mudanças externas no ambiente, erros incorridos no processo de requisitos, entre outros, todos esses fatores fazem com que seja necessário alterar os requisitos. Ávila e Spíndola (2007) destacam que as alterações precisam ser conduzidas de forma ordenada para que não se perca controle sobre o prazo e o custo do desenvolvimento.

O sucesso da Gerência de Requisitos depende diretamente de quão bem definidos são os relacionamentos entre os requisitos e entre os outros conjuntos de artefatos gerados durante o processo de desenvolvimento (LETELIER, 2002 apud LEAL, 2009). Sendo esta tarefa de responsabilidade da Rastreabilidade de Software, completa Leal (2009).

Gerenciamento de requisitos é a atividade de administrar os requisitos ao longo do tempo de desenvolvimento do sistema. Os benefícios dessa atividade são percebidos no médio prazo, sendo que são necessários investimentos no curto prazo. A seguir serão apresentados as atividades durante o processo de gerenciamento.

3.2.1.2.1 Controle de Mudanças

Para Sommerville (2005), o gerenciamento de mudança deve ser aplicado em todas as mudanças dos requisitos. Ele considera como vantagem, na utilização de processo formal, que todas as mudanças serão tratadas de modo consistente e de modo controlado.

Conforme foi citado anteriormente, os requisitos são voláteis e, portanto, sofrem mudanças ao logo do tempo. Uma das maneiras bastante utilizadas para organizar estas mudanças é a "*baseline*" de requisitos que nos permitem diferenciar o que era o requisito original, o que foi introduzido e o que foi descartado. (ÁVILA e SPÍNDOLA, 2007).

Para Sommerville (2005), existem três estágios no controle de mudanças:

a) análise do problema e especificação da mudança: Nesse estagio é realizado a análise do problema ou da proposta de mudança, tendo como finalidade verificar a sua validade;
b) análise e custo de mudança: avaliação do efeito da proposta, verificando a facilidade de rastreamento e conhecimento do requisito. Nessa etapa é definido se será mudado o requisito ou não, tendo em vista, também, o custo da modificação;
c) implementação da mudança: alteração do documento de requisitos para que essas possam se acomodar sem muito esforço e facilitando novas alterações;

3.2.1.2.2 Gerência de Configuração

Durante o ciclo de vida do desenvolvimento, o software passa por uma série de modificações, desde a especificação dos requisitos até a implantação do sistema. A gerência de configuração de *software* existe no intuito de definir critérios que permitam realizar tais modificações, mantendo-se a consistência e a integridade do software com as especificações, minimizando problemas decorrentes ao processo de desenvolvimento, através de um controle sistemático sobre as modificações.

A criação de um plano de gerência de configuração consiste em estabelecer normas, ferramentas e *templates* que permitam gerenciar de maneira satisfatória os itens de configuração de um sistema, que são definidos por Pressman (2006), "cada um dos elementos de informação que são criados, durante o desenvolvimento de um produto de software, ou que

para esse desenvolvimento sejam necessários, que são identificados de maneira única e cuja evolução é passível de rastreamento".

Em cada fase do processo de desenvolvimento, um conjunto bem definido de itens de configuração deve ser estabelecido. A esse conjunto é dado o nome de *baselines*. Estas *baselines* servem como marco no processo de desenvolvimento, pois, ao final de cada fase, é estabelecida uma nova *baseline* e, portanto, todos os itens de configuração estão sob total controle de seus desenvolvedores.

3.2.1.2.3 Rastreabilidade

No centro da atividade de gerenciamento de requisitos, está a rastreabilidade. Para Spanoudakis e Zisman (apud LEAL, 2009), rastreabilidade é a capacidade de relacionar artefatos criados durante o processo de desenvolvimento, *stakeholders* que contribuíram com a elaboração de um artefato e decisões e informações que contribuirão para a construção de um artefato.

Para alguns autores, como Gotel e Finkelstein (1995 apud CRUZ et al. 2006), a rastreabilidade é tida como complexa de implantar e cara de ser mantida.

> A dificuldade envolvida com a rastreabilidade está no grande volume de informações gerado. As informações do processo de requisitos devem ser catalogadas e associadas aos outros elementos de forma que possam ser referenciadas através dos diversos itens de informação registrados. É um trabalho extenso que, sem o auxílio de ferramentas adequadas, muito provavelmente não poderá ser feito. (AVILA e SPÍNDOLA, 2007).

Entretanto, os mesmos autores identificam que a ausência da rastreabilidade faz com que as economias de curto prazo sejam logo suplantadas pelas despesas no longo prazo, verificadas com superação de custo e prazo nos projetos conduzidos.

Alguns dos benefícios gerados pela rastreabilidade são destacados como análise de riscos e impactos sobre modificações/integrações de possíveis conflitos entre requisitos, aumento da reutilização a partir da identificação de componentes, testes contínuos sobre os requisitos e os links entre os mesmos. (CLELAND-HUANG, 2006; EGYED, GRUENBACHER, 2002; KELLEHER, SIMONSSON, 2006; SPANOUDAKIS, ZISMAN; 2005 apud LEAL 2009).

Segundo De Lucia e Oliveto (2008 apud LEAL, 2009), a razão mais clara para utilizar a rastreabilidade é o apoio à análise de impacto que é a identificação do impacto que uma mudança de requisito gerará.

4 RASTREABILIDADE

O processo de desenvolvimento de software é bastante complexo e dinâmico, exigindo vários documentos ou artefatos. A cada etapa do desenvolvimento, novos documentos (artefatos) são criados, podendo ser alterados diversas vezes. Geralmente, uma modificação pode causar impacto em vários outros documentos. (CRUZ et al., 2006).

Espíndola e outros (2004) definem o termo de rastreabilidade como a capacidade de recuperar, rapidamente, todos os artefatos (digramas, código, folha de teste, etc.) gerados a partir de um requisito.

"A rastreabilidade auxilia na avaliação do impacto no projeto que eventuais modificações em artefatos possam causar e, também, contribui para verificar se todas as partes envolvidas no projeto têm suas necessidades satisfeitas". (CRUZ et al., 2006).

Em contraponto, aos benefícios identificados, a rastreabilidade é tida como problemática por fatores de custo, cronograma e frequentes atualizações dos sistemas em desenvolvimento. (MILLER, 1991).

4.1 MATRIZ DE RASTREABILIDADE

Na rastreabilidade, usualmente é confeccionado, em conjunto com a especificação de requisitos, um artefato chamado matriz de rastreabilidade, que tem como objetivo mapear os rastros dos requisitos descritos na especificação. (SOMMERVILLE, 2005).

Os rastros dos requisitos podem ser de dois tipos:

- pré-rastreabilidade: os rastros (artefatos: plano de negócio, atas de reunião com o usuário) que fundamentaram a criação do requisito;
- pós-rastreabilidade: os rastros (artefatos: código, documentos) que se formaram a partir do requisito criado.

Modelar a rastreabilidade em um projeto envolve considerar quais associações e quais informações de rastreabilidade devem ser controladas. (CRUZ et al., 2006).

Segundo Backes (2009), as matrizes de rastreabilidade são geralmente utilizadas para exibir os relacionamentos entre elicitação de requisitos. Elos de rastreabilidade são geralmente estabelecidos pelo relacionamento Fletcher e Cleland-Huang (2006).

Para Marquioni (2004 apud SANTOS et al. 2004), foram definidos alguns tipos de rastreabilidade, conforme o Quadro 2:

Tipo de rastreabilidade	Descrição
Requisitos – Fontes	Link do requisito às pessoas ou documentos que especificaram o requisito.
Requisitos – Razão	Link do requisito com uma descrição de "porque" foi especificado. Pode ser uma destilação de informações de várias fontes.
Requisitos – Requisitos	Link do requisito com outros requisitos que sejam, de alguma maneira, dependentes dele. Deve ser um link de "mão dupla" ("depende" e "é dependente de").
Requisitos – Arquitetura	Link do requisito com os subsistemas onde estes requisitos estão implementados. Particularmente importante se os subsistemas estiverem sendo desenvolvidos por subcontratados diferentes.
Requisitos – *Design*	Link do requisito com hardware ou componentes de software específicos no sistema que são usados para implementar o requisito.
Requisitos – Interface	Link do requisito com as interfaces de sistemas externos que serão usados na provisão dos requisitos.

Quadro 2 - Tipos de Rastreabilidade.

Fonte: SANTOS, Fernando dos; VARGAS, Karly S.; ABREU, Christian R. C.; In: Rastreabilidade de Requisitos através da web, 2004, Blumenau, p. 03.

As Matrizes de rastreabilidade são geralmente utilizadas para exibir os relacionamentos entre elicitação de requisitos e a representação desses requisitos em um método particular da engenharia de requisitos de *software*. (BACKES, 2009). A Figura 11 é um exemplo de matriz de rastreabilidade, criada através da ferramenta CASE *Enterprise Architect*.

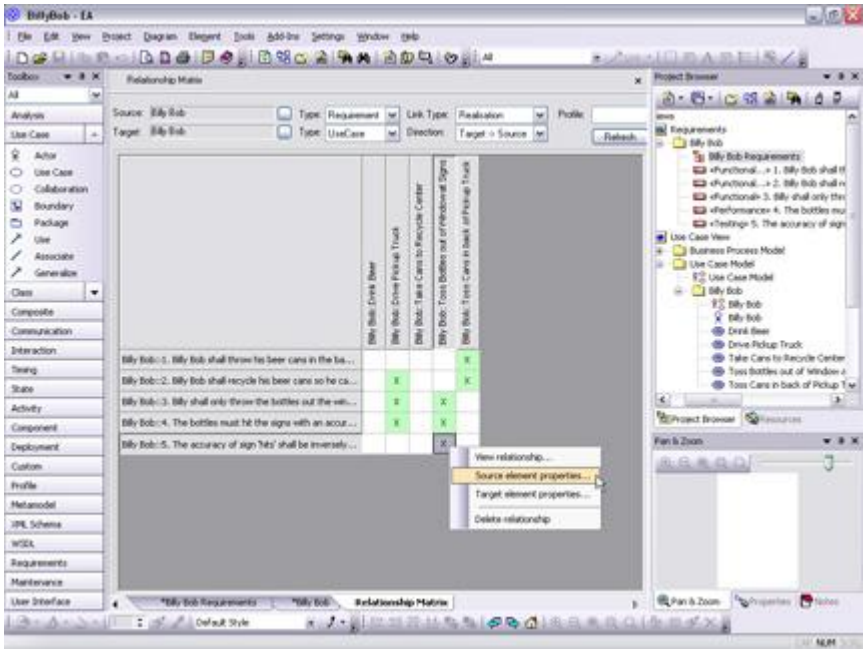


Figura 11 - Matriz de Rastreabilidade.
Fonte: **SPARTX SYSTEMS**; 2009.

4.2 MÉTODOS DE RECUPERAÇÃO AUTOMÁTICA

Técnicas de geração automática de elos, entre requisitos expressos textualmente e modelos posteriores, como código fonte, vem emergindo como alternativa para resolver o problema da cara e complexa manutenção da rastreabilidade. (SPANOUDAKIS, 2002; EGYED, et al 2006; LORMANS, DEURSEN, 2006; ANTONIOL et al. 2000; apud BACKES, 2009).

A recuperação de elos funciona na seguinte seqüência (PENTA et al. 2002, apud BACKES, 2009):

a) processamento de requisitos: cada requisito é processado e um dicionário de palavras é montado, com base na frequência de cada uma. Essa fase, em geral, é semi-automática, pois não é possível retirar a ambiguidade da semântica contextual, nem lidar com discursos metafóricos (eg. “deve-se ’matar’ o processo...”);

b) processamento do código: para cada classe, uma lista de identificadores, contendo suas ocorrências é extraída. Aqui, a intervenção humana é necessária para informar o significado de palavras curtas, com menos de três letras;

c) recuperação do mapa de rastreabilidade: é realizada através de um classificador.

Entretanto, o mesmo autor destaca o fato de que não existe processo de recuperação da rastreabilidade completamente automático. Todos incluem um componente humano, pois as decisões (de aceitação ou rejeição) sobre elos recuperados devem ser tomadas pelo usuário. Para Cleland-Huang (2006, tradução livre), os métodos de recuperação automática estão sendo cada vez mais reconhecidos pela indústria como uma solução potencial para o problema da rastreabilidade.

4.3 MODELOS E CARACTERÍSTICAS DOS MÉTODOS DE RASTREABILIDADE ENTRE OS REQUISITOS E O CÓDIGO DE SOFTWARE

Alguns conceitos são relevantes para o entendimento deste processo, identificados e conceituado por Leal (2007):

a) **Rastreabilidade de requisitos**: habilidade de descrever e seguir a vida de um requisito nos dois sentidos, de avanço e retorno.

b) **Ligação de Rastreabilidade** (*trace link*): é o elemento que liga dois itens rastreáveis.

c) **Unidade Computacional**: uma parte executável de um sistema, por exemplo, instruções (como acesso a variáveis globais), blocos de código básicos, rotinas, classes, unidades de compilação, módulos ou subsistemas.

d) **Corte Funcional** (*Slice*): é o conjunto de Unidades Computacionais que podem afetar um conjunto determinado de variáveis. Os algoritmos de corte podem ser classificados como algoritmo para Cortar estático (*static slicing*), que usa apenas informações estáticas do código para se obter o corte; e o algoritmo de Cortar dinâmico (*dynamic slicing),* que usa também valores de variáveis que podem influenciar o corte.

e) **Item de Rastreabilidade** (*traceability item*): qualquer item textual ou item de modelo que precisa ser explicitamente rastreado para outro item textual ou item de modelo, para manter a trilha de dependência entre eles.

f) **Item de código de rastreabilidade** (ICR): Quando o item de rastreabilidade for uma Unidade Computacional ou um Corte Funcional, pode ser designado genericamente por item de código de rastreabilidade ou ICR.

4.4 CENÁRIOS

Cenários vêm sendo utilizados como um meio alternativo e, por vezes, complementar, para expressar requisitos e o comportamento do sistema através das diversas fases de desenvolvimento. Cada cenário pode assumir representações e semânticas diferentes entre cada fase, e estas podem ser relacionadas. (BACKES, 2009).

Naslavsky et. al.(2005, apud BACKES, 2009) identificam diversos tipos de cenários utilizados em diferentes fases do desenvolvimento:

a) **Cenários de Requisitos**: são utilizados para descrever requisitos após uma fase preliminar de levantamento. É o tipo de cenário mais próximo da compreensão dos *stakeholders*, ao contrário dos utilizados em fases posteriores. Os cenários de requisitos descrevem sequências de eventos que ocorrem sobre o sistema proposto. Um exemplo desse tipo de cenário são Use Cases;
b) **Cenários de Análise**: adicionam mais detalhes e uma perspectiva da parte interna do sistema aos cenários de requisitos. São identificadas entidades de alto nível, que farão parte do sistema. Da mesma forma, eventos, que antes eram representados por meio de linguagem natural, assumem a forma de interações entre os conceitos do sistema. Um exemplo desse tipo de cenário são diagramas de sequência da UML;
c) **Cenários de Arquitetura**: acrescentam mais detalhes aos anteriores. Nessa fase, são tomadas decisões sobre quais conceitos tornar-se-ão componentes da arquitetura do sistema. Um exemplo desse tipo de cenário é um diagrama de sequência da UML, ou um gráfico para sequenciamento de mensagens, em que agentes que trocam mensagens são componentes arquiteturais e conectores;
d) **Cenários de Projeto**: são cenários descritos em maiores detalhes do que os de arquitetura. Os conceitos usados nesse caso são partes de objetos de

componentes arquiteturais, exibindo, assim, o funcionamento interno dos componentes;

e) **Cenários de Código**: descrevem o fluxo de eventos a partir do nível de implementação do sistema, que são, na verdade, informações de rastreamento em tempo de execução, correspondendo a comandos executados ou invocações de métodos. Assim, cenários de requisitos, análise, arquitetura e de projeto, quando executados, resultam em algum cenário de código;
f) **Cenários de Teste**: aparecem em cada fase do ciclo de desenvolvimento e são utilizados para testar a implementação do sistema ou outros artefatos. Os cenários definidos, acima, para cada fase do ciclo de vida, podem ser utilizados como cenários de teste. Assim, cenários, em cada nível de abstração, são utilizados para testar se o comportamento de um cenário de nível inferior está de acordo com o comportamento esperado, descrito pelo de nível superior. Já que um cenário de nível superior pode levar a múltiplos cenários de nível inferior, é necessário testar a sua conformidade.

Existem diversas formas de definir o rastreamento entre cenários. Utilizar a mesma linguagem para descrever todos os tipos de cenários facilita o estabelecimento e manutenção do relacionamento entre eles. (BACKES, 2009).

4.5 GRANULARIDADE DA RASTREABILIDADE

Leal e outros (2009) afirmam que um fator importante na definição dos itens de rastreabilidade é o nível de automatização. Se a rastreabilidade for manual, as informações deverão ter maior granularidade para viabilizar a Rastreabilidade. Quanto maior a automatização, maior a possibilidade de diminuição da granularidade para que se ganhe em riqueza e precisão das informações.

O uso excessivo da rastreabilidade pode resultar em um emaranhado de links de difícil manutenção e utilização. Ou seja, não faz sentido rastrear todos os requisitos com a mesma granularidade. (DOMGES e POHL, 1998, apud BACKES, 2009).

Para Backes (2009), os links de rastreabilidade devem ser tratados de forma diferente, pois não são todos de igual importância. Deve-se considerando a classificação dos artefatos, através da identificação com alguma medida de valor e então identificar o grau de relevância para o projeto.

5 DEFINIÇÃO DO PROCESSO DE SOFTWARE

Segundo Vidal e outros (2003), o processo de desenvolvimento de *software* tem como objetivo garantir a qualidade do produto desenvolvido indo a encontro com a "boa produtividade" de desenvolvimento. Ainda completa que tem como componentes os papeis, artefatos, atividades, técnicas, praticas e ferramentas.

O processo de *software* descreve quem faz o que, como e quando; define as atividades da equipe e quais artefatos devem ser desenvolvidos. (DATASUS, 2009).

Esta metodologia apresenta vários conceitos que podem e devem ser utilizados em processos de desenvolvimento. Autores, como Vidal (2003) e Datasus (2009), destacam alguns deles:

- modelo genérico de processo da Rational que fornece uma abordagem disciplinada para produzir e gerenciar software;
- iterativo e incremental;
- representado através de UML;
- guiado por casos de uso e cenários típicos de utilização e atividade que conduz todo o processo de desenvolvimento, desde a análise de requisitos ao teste do sistema final;
- a definição inicial de uma arquitetura de software robusta, que posteriormente facilitará o desenvolvimento em paralelo, a sua reutilização e manutenção;
- planejado com foco na eliminação de riscos;

Complementando os conceitos de RUP, Datasus (2009) destaca práticas de sucesso na indústria de software: desenvolvimento iterativo e incremental, gerência de requisitos, gerência de configuração, gerência de mudanças, participação ativa dos usuários e melhoria contínua do processo de desenvolvimento.

O processo de desenvolvimento de software é baseado no RUP através da customização, visando simplicidade e objetividade. Com o objetivo de viabilizar a utilização do RUP, de acordo com a nossa realidade no desenvolvimento deste *software*, foi necessária a customização que então gerou um novo processo de desenvolvimento.

O processo se difere do todo, que é o RUP, principalmente nos artefatos e atividades elaboradas. De certa forma, as atividades foram enxugadas para que o processo não

se torne muito grande, complexo e inviável dentro do projeto. A seguir o detalhamento das fases do novo processo.

5.1 FASES

A partir de uma perspectiva de gerenciamento, o ciclo de vida de *software* é dividido em quatro fases seqüenciais, cada uma concluída por um marco principal, ou seja, cada fase é, basicamente, um intervalo de tempo entre dois marcos principais.

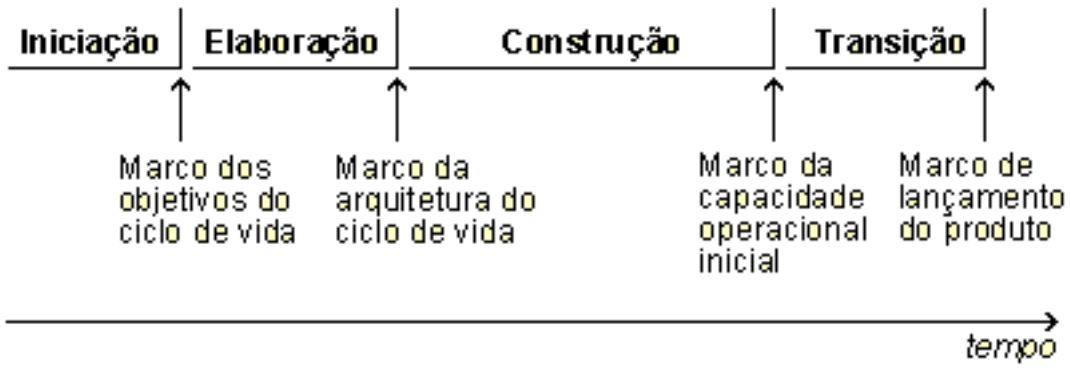


Figura 12 – As fases e os marcos de um projeto.
Fonte: **RATIONAL SOFTWARE**; *Rationa Unified Process*: 2002; Português v. 2002.05.00;

5.1.1 Iniciação

O objetivo da fase de Iniciação é o estabelecimento de um acordo formal entre a equipe de desenvolvimento e usuários do projeto, do escopo do produto a ser desenvolvido.

O marco dessa fase será o escopo aprovado. Os artefatos produzidos são Documento de Visão do Negócio e Glossário.

Na Visão do Negócio, estará o objetivo de alto nível para o desenvolvimento do projeto, especificando principais questionamentos relacionados. Será desenvolvido pelo

gerente de projeto, aprovado pelo cliente e lido por todos os envolvidos, com o sistema, visando o entendimento do escopo do sistema.

O artefato Glossário define os termos específicos ao domínio do sistema e do projeto, facilitando a comunicação entre todas as partes. Nessa fase, o glossário será desenvolvido de forma inicial e completado nas próximas fases.

Após o escopo do produto definido e os artefatos criados, Conforme apresentado na Figura 13, a fase estará concluída e o processo dará sequência na fase de Elaboração.

Figura 13 - Fase de Iniciação.
Fonte: Elaborado pelo Autor.

5.1.2 Elaboração

O objetivo da fase de Elaboração é o estabelecimento e a validação de uma arquitetura de hardware e de software que suporte de forma adequada os requisitos funcionais e não-funcionais do sistema. (DATASUS, 2009).

O marco desta fase é a Arquitetura Validada, e os artefatos produzidos são: Matriz de Requisitos, Modelo de Casos de Uso, Documento de Arquitetura, Diagrama de Classes, Modelo de Banco de Dados, Versão do Sistema (arquitetura testada e validada) e Glossário.

- matriz de Requisitos: ou matriz de rastreabilidade demonstra a intersecção entre requisitos e regra de negócio de forma inicial;
- modelo de Casos de Uso: demonstra a visão funcional do sistema através de seus atores e casos de uso;
- documento de Arquitetura: especificações de estrutura de hardware, software do sistema que foram adotados pela equipe de desenvolvimento;
- diagrama de Pacotes: para organizar as classes de acordo com sua responsabilidade;
- diagrama de classes: apresenta uma visão estática do sistema, projetado pela realização dos casos de uso e de seus relacionamentos;
- modelo de banco de dados: apresenta o modelo do banco do sistema através do diagrama entidade-relacionamento, modelo relacional e dicionário de dados;
- versão do sistema: representa o código fonte implementado ao longo do desenvolvimento;

Os papeis envolvidos são Analista de Sistemas, Projetista de Banco de Dados e o Programador, que são responsáveis por algumas atividades, conforme detalhando na Figura 14. Após a conclusão de todas as atividades, através do gerenciamento dos artefatos esta fase estará concluída e iniciando a fase de Construção.

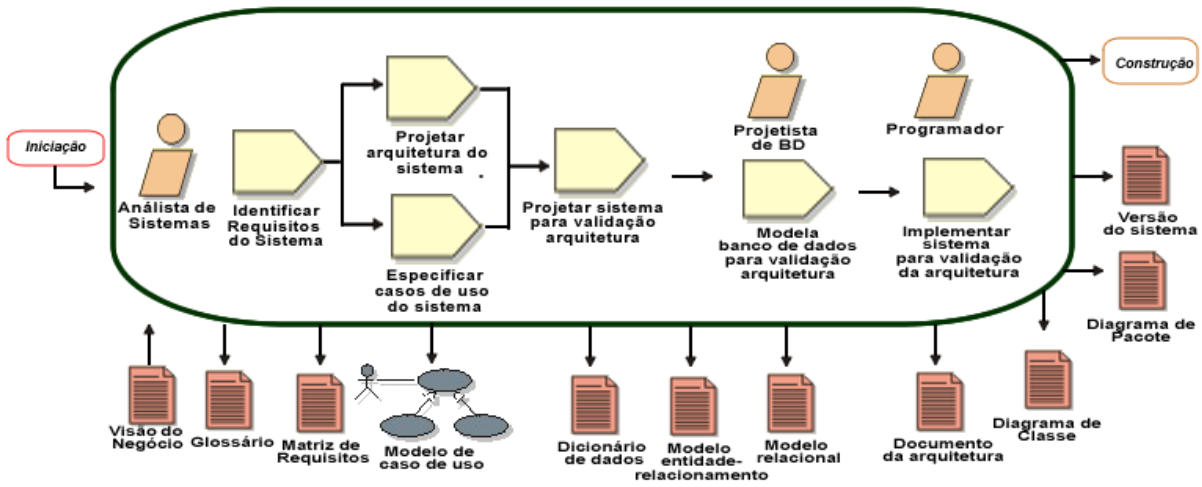


Figura 14 - Fase de Elaboração.
Fonte: Elaborado pelo Autor.

5.1.3 Construção

Na fase de construção, em que serão refinados os requisitos e concluido o desenvolvimento do sistema. Para a Rational (2009), nesta fase que da ênfase no gerenciamento de recursos e controle de operações para otimizar custos, programações e qualidade. O desenvolvimento propriamente dito, em termos de código-fonte e componentes de *software*.

O objetivo dessa fase é o desenvolvimento de uma versão operacional, gerando iterações que são testadas, contendo os cenários de caso de uso e implementados até o momento.

Artefatos Produzidos: Diagrama de Classes, Modelo de Banco de Dados, Versão do Sistema. Marco desta fase é Versão Operacional do Sistema.

- Pacote de Distribuição: versão funcional do sistema, com a liberação de do sistema em uma base de testes.

Executadas todas as atividades exibidas na Figura 15, logo a implementação do sistema esta concluído e dará inicio na fase de Transição.

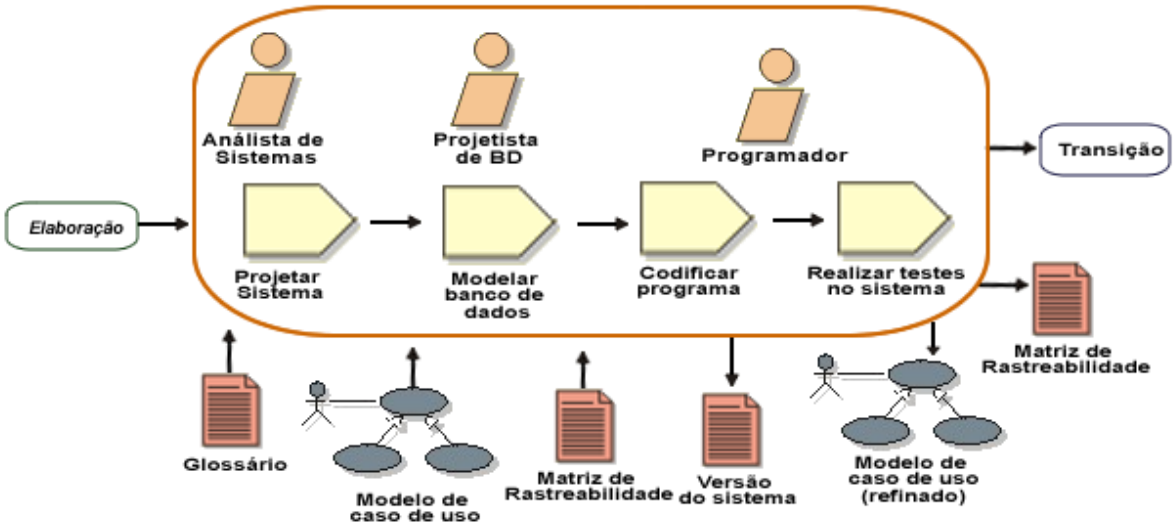


Figura 15 - Fase de Construção.
Fonte: Elaborado pelo Autor.

5.1.4 Transição

O foco da Fase de Transição é assegurar que o software esteja disponível para seus usuários finais. A Fase de Transição pode atravessar várias iterações de ajustes pequenos com base no feedback do usuário. Nesse momento do ciclo de vida, o feedback do usuário deve priorizar o ajuste fino do produto, a configuração, a instalação e os problemas de usabilidade. (RATIONAL, 2009)

O Marco da fase é o Release do Produto e é nesse momento, que identifica se os objetivos foram atendidos, e se outro ciclo de desenvolvimento deve ser iniciado. Com a apresentação do sistema, logo efetuando o ajusto fino do produto: validando, configuração, instalação e quaisquer ajustes necessários para implantação do sistema.

Os artefatos produzidos são: Versão do Sistema (homologada) e documentação de Usuário.

- documentação de usuário: consiste na documentação elaborada com o foco no usuário final do sistema.

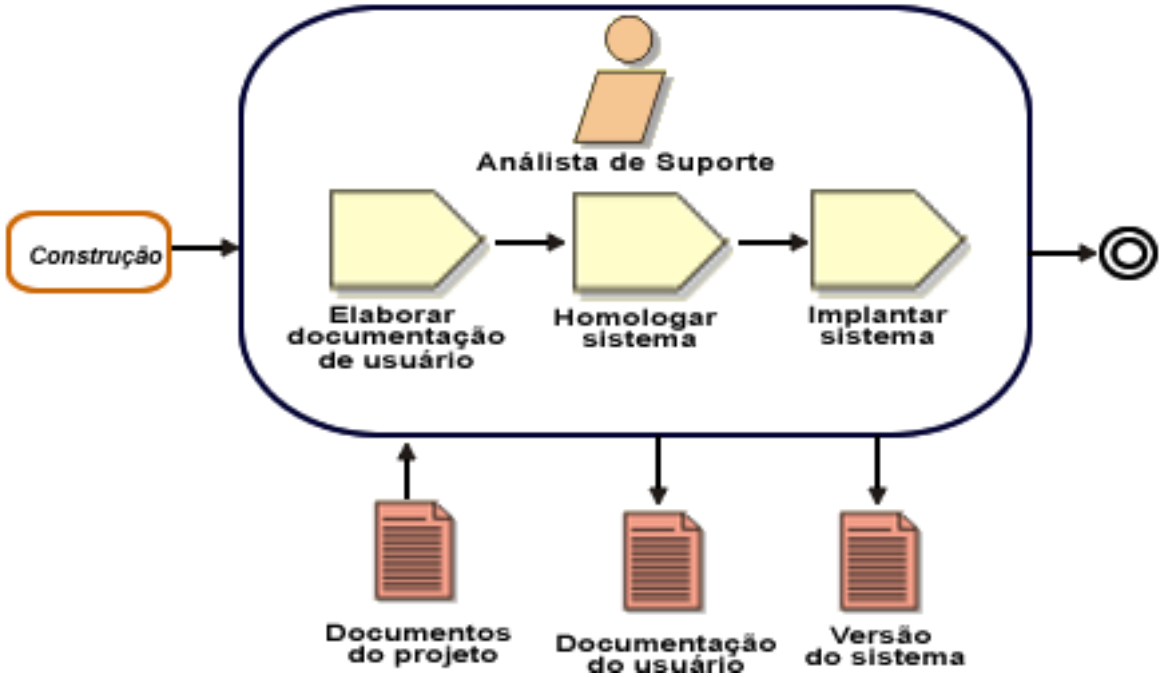


Figura 16 - Fase de Transição.
Fonte: Elaborado pelo Autor.

6 MODELAGEM DO SISTEMA

O sistema protótipo, HEMOCOM, tem a finalidade de gerenciar o envio de comunicados internos e externos para os colaboradores do Hemosc. O principal objetivo é atingir o maior numero de pessoas em um menor tempo e agilizar o procedimento que, atualmente, acontece de forma manual.

Neste tópico, será apresentado o processo de análise e documentação do sistema, elaborado durante as fases de desenvolvimento do *software*, através da ferramenta CASE *Enterprise Architect*.

No ciclo de iniciação foram gerados os artefatos Glossário de Negócio e Visão de Negócio. Ambos são apresentados nos apêndices A e B, apenas para entendimento do sistema protótipo.

Os tópicos a seguir, representam a real ordem de execução no processo de desenvolvimento, embora alguns deles sejam revistos em mais de uma fase, sendo essa a sequência de desenvolvimento.

6.1 VISÃO DE REQUISITOS

Na fase de Elaboração foi iniciado a Visão de Requisitos, Figura 17, essa visão esta separada entre Requisitos Funcionais, Requisitos Não Funcionais e Regras de Negócio. Os requisitos apresentam uma evolução durante, permanecendo durante as fases de Elaboração e Construção.

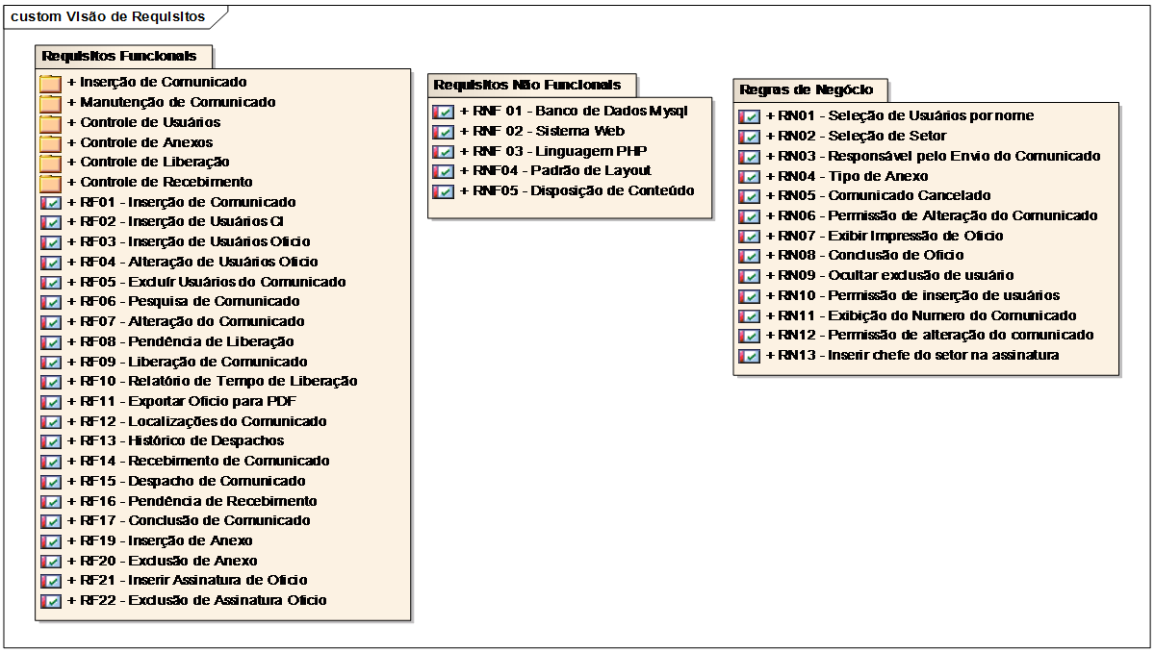


Figura 17 - Visão de Requisitos.
Fonte: Elaborado pelo Autor.

Com o objetivo de facilitar a visualização dos Requisitos Funcionais, foram separados em pequenos ‘módulos’, como por exemplo, Figura 18 - Módulo de Inserção de Comunicado e Figura 19 - Módulo de Controle de Usuários.

No módulo de Inserção de Comunicado destaca os Requisitos Funcionais e Regras de Negócios necessários para o Requisito Funcional RF01 – Inserção de Comunicado. Este requisito, por sua vez, realiza a regra RN03 e o RF08 – Pendência de Liberação esta associada ao requisito RF01, pois ao inserir um comunicado uma pendência de liberação será gerada.

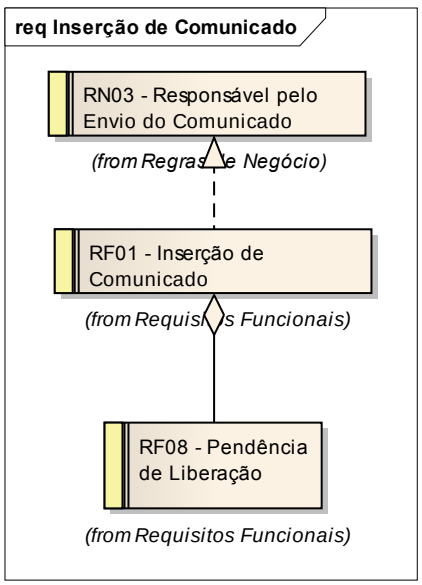


Figura 18 - Módulo de Inserção de Comunicado.

Fonte: Elaborado pelo Autor.

O módulo Controle de Usuários especifica os Requisitos Funcionais e Regras de Negócio, que estão envolvidos com operações de usuários do comunicado. Por exemplo, RF15 – Despacho de Comunicado, processo que insere novos usuários no comunicado após a sua liberação, e realizando as regras RN01 e RN02, similar ao RF02 – Inserção de Usuários CI.

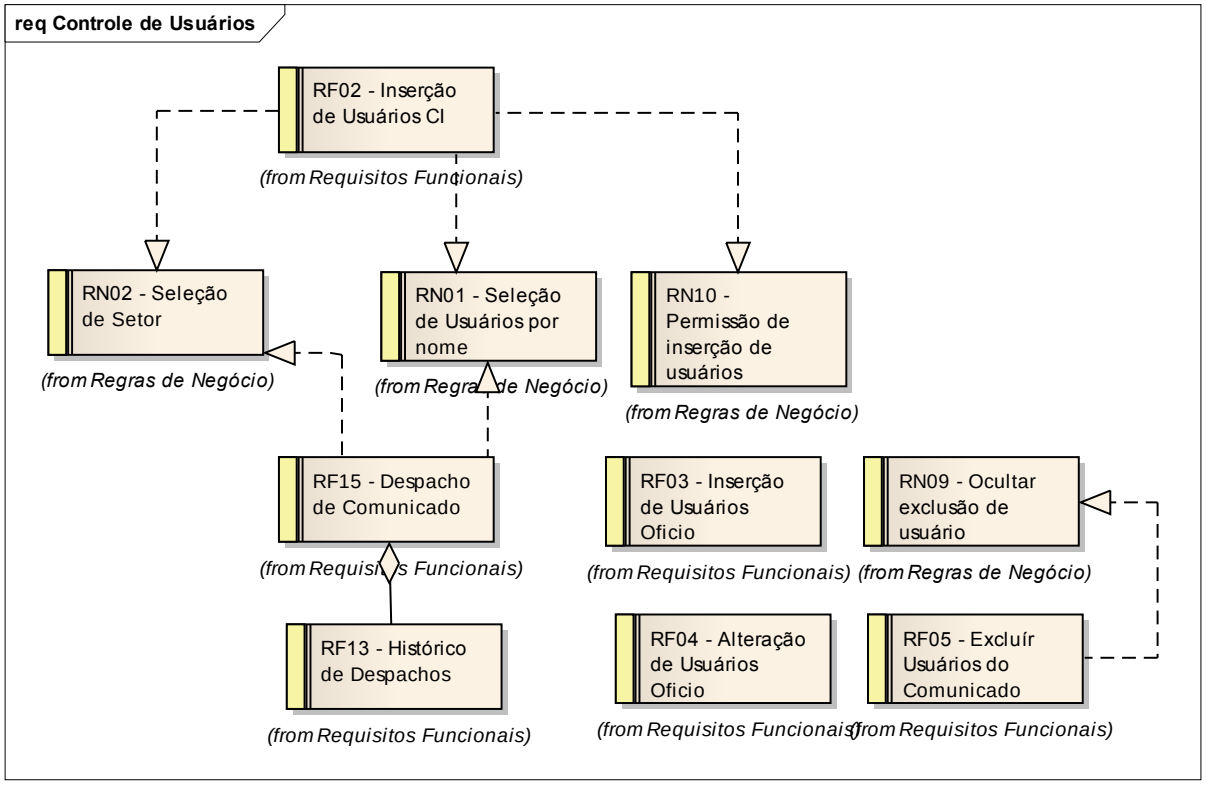


Figura 19 - Módulo de Controle de Usuários.
Fonte: Elaborado pelo Autor.

6.2 VISÃO DE CASO DE USO

A visão de caso de uso apresenta uma visão global sobre interações dos atores e sistema. Separados em dois pacotes esta as descrições dos atores e o modelo de caso de uso.

6.2.1 Atores

Os atores envolvidos são todos Usuários Comuns, que exercem responsabilidades básicas. Por exemplo, um usuário comum pode inserir um comunicado e ao inserir o comunicado o Usuário Comum passa a ser um novo Ator o Responsável do Comunicado, logo para detalhar estes relacionamentos foram destacados como herança entre atores.

O Assessor da chefia é o ator que substitui o Chefe do Setor, então necessita ter as mesmas responsabilidades que ele, entretanto também é um Usuário Comum que pode tanto inserir um comunicado, como receber comunicado entre outras funções, detalhadas no próximo tópico.

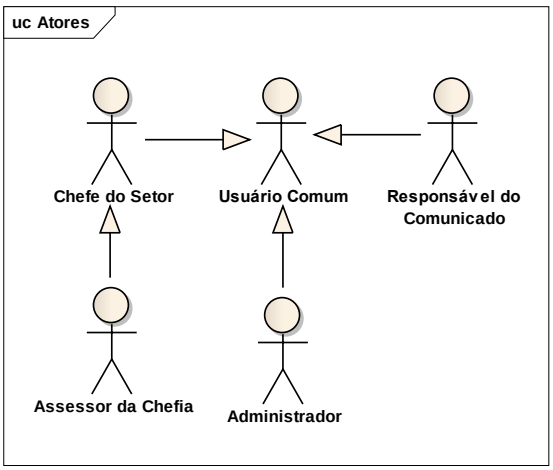


Figura 20 - Atores do Sistema
Fonte: Elaborado pelo Autor.

6.2.2 Modelo de Caso de Uso

O Modelo de Caso de Uso apresenta uma visão geral do sistema, Figura 21. No CSU01 são inseridos os comunicados e acionados os CSU02, CSU03 e CSU04, sendo obrigatória a execução dos CSU02 e CSU04. Uma das operações indispensável para o comunicado é a liberação efetuada pelo chefe do setor (realizado no CSU04) e o envio aos usuários do Hemosc (realizado no CSU02).

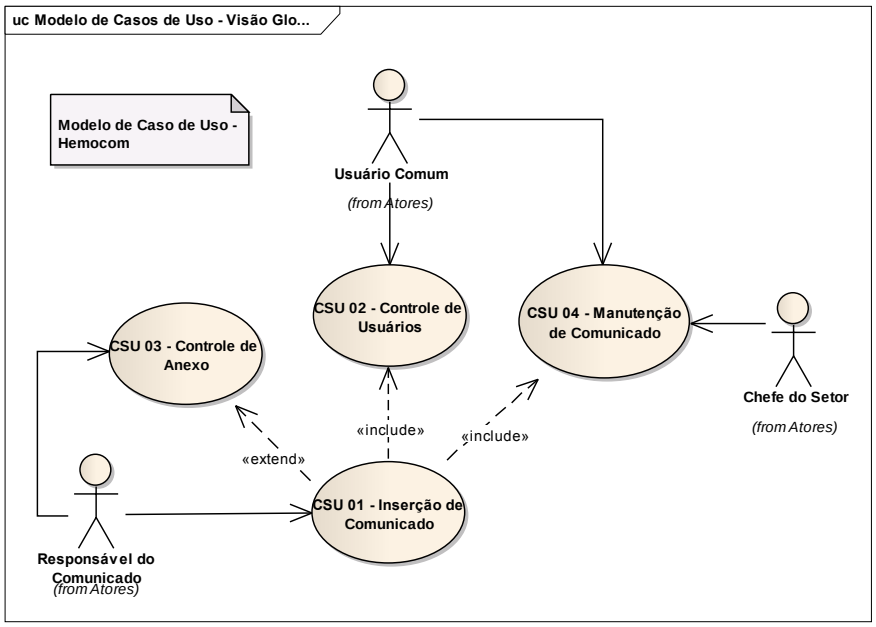


Figura 21 - Modelo de Caso de Uso.
Fonte: Elaborado pelo Autor.

A definição dos Casos de Uso inicia ainda na fase de elaboração, mas seu refinamento acontece posteriormente, no ciclo de Elaboração, e nessa fase que é definida a ligação entre os requisitos e caso de uso, Figura 22, apresenta todos os requisitos que o Caso de Uso deve implementar.

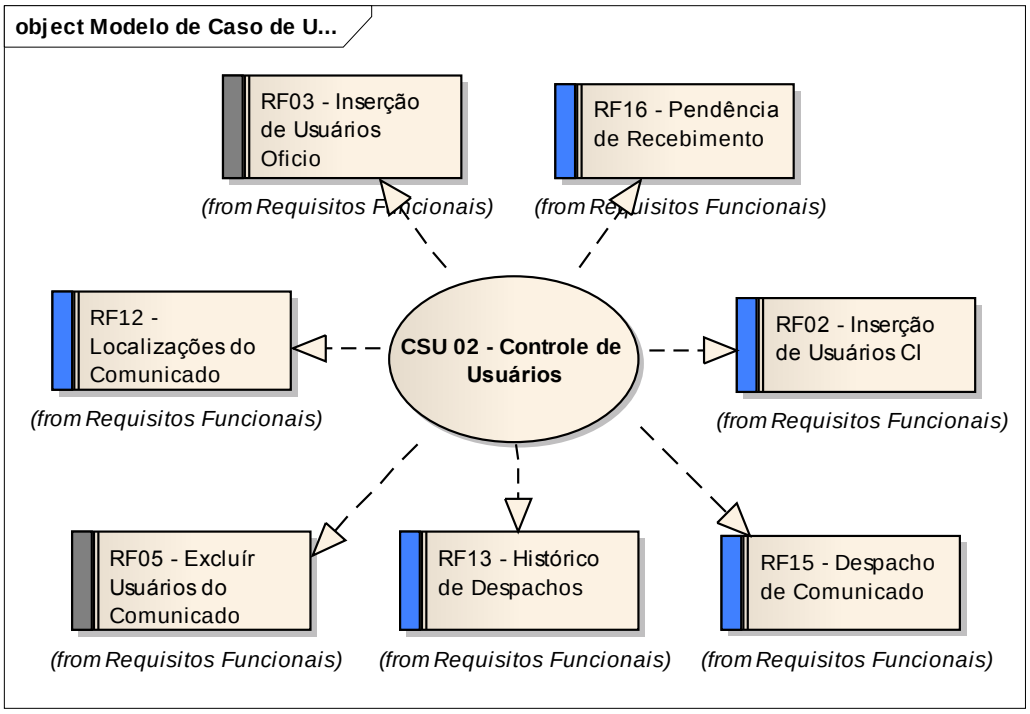


Figura 22 - Requisitos do Caso de Uso 04 e seus respectivos cenários.
Fonte: Elaborado pelo Autor.

Ainda na fase de Elaboração devem ser definidos os cenários de implementação dos requisitos, ou seja, como devem acontecer exatamente no sistema, assim como na Figura 23.

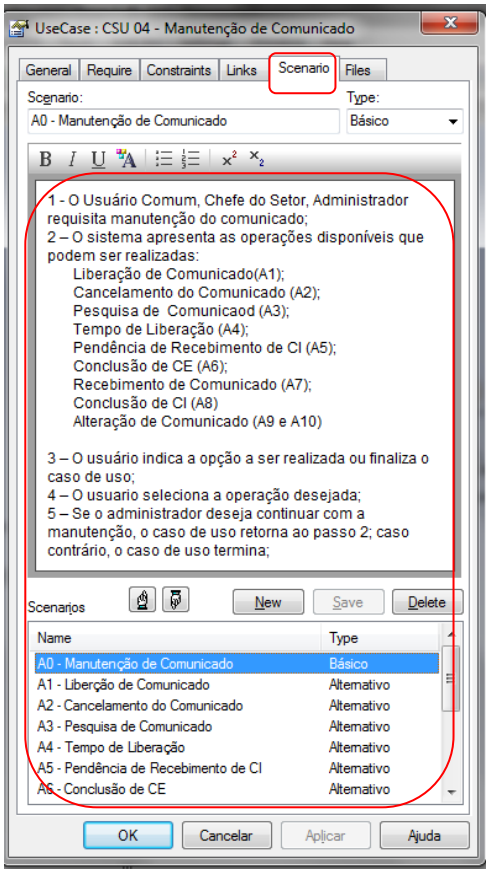


Figura 23 - Cenários do Caso de Uso 04.
Fonte: Elaborado pelo Autor.

6.3 VISÃO DE ANÁLISE

Na visão de análise todos os Casos de Uso serão analisados para buscar as classes que colaboram para atender os requisitos.

6.3.1 Realização

Na realização esta destacada como acontece à implementação dos Casos de Uso. Por exemplo, na Figura 24 que apresenta a realização do Caso de Uso CSU01, sendo efetuada pelo RCSU 01.

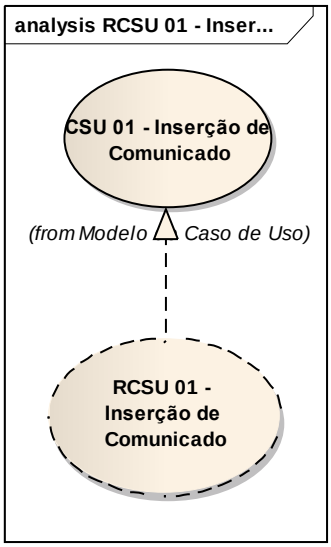


Figura 24 - Realização do Caso de Uso 01.
Fonte: Elaborado pelo Autor.

Com o objeto de atingir o mais baixo nível do sistema, através da realização dos casos de Uso, exibido na Figura 25 - Diagrama de Robustez e Figura 26 - Diagrama de Sequencia do RCSU 01 chegando até o código conte.

O diagrama de Robustez apresenta as interfaces utilizadas na realização da Inserção do Comunicado, implementado pelo caso de uso 01. Logo o arquivo ‘ins_comn’ corresponde a interface de tela, o controle é o arquivo ‘CtrlComunicado’ e o objeto é ‘Comunicado’.

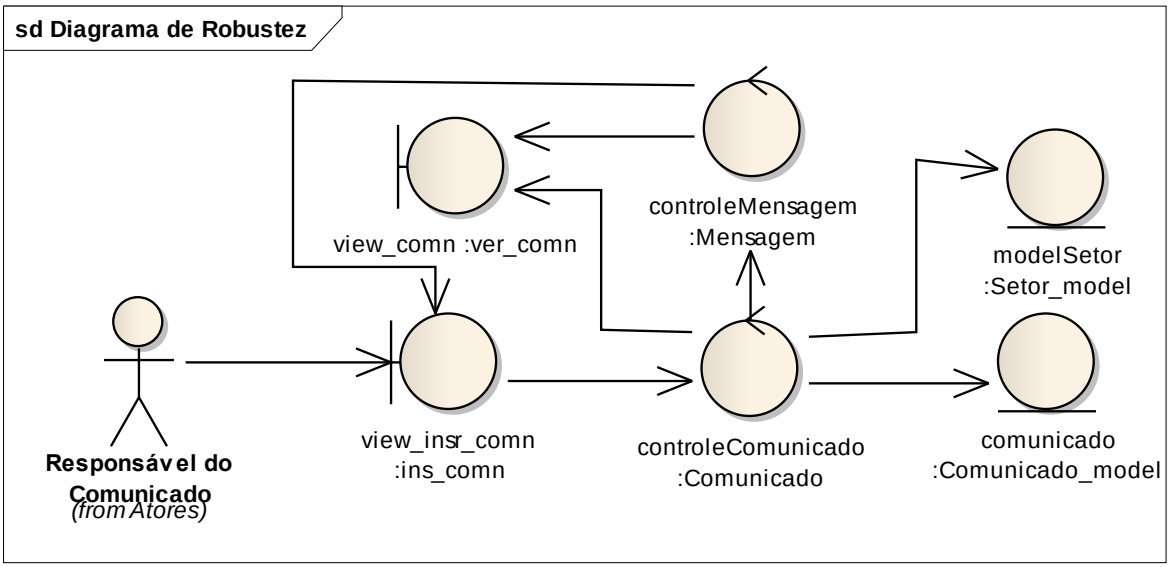


Figura 25 - Diagrama de Robustez.
Fonte: Elaborado pelo Autor.

O diagrama de sequência aprofunda, um pouco mais, o processo de “Inserir Comunicado” exibindo as mensagens trocadas entre as diferentes interfaces.

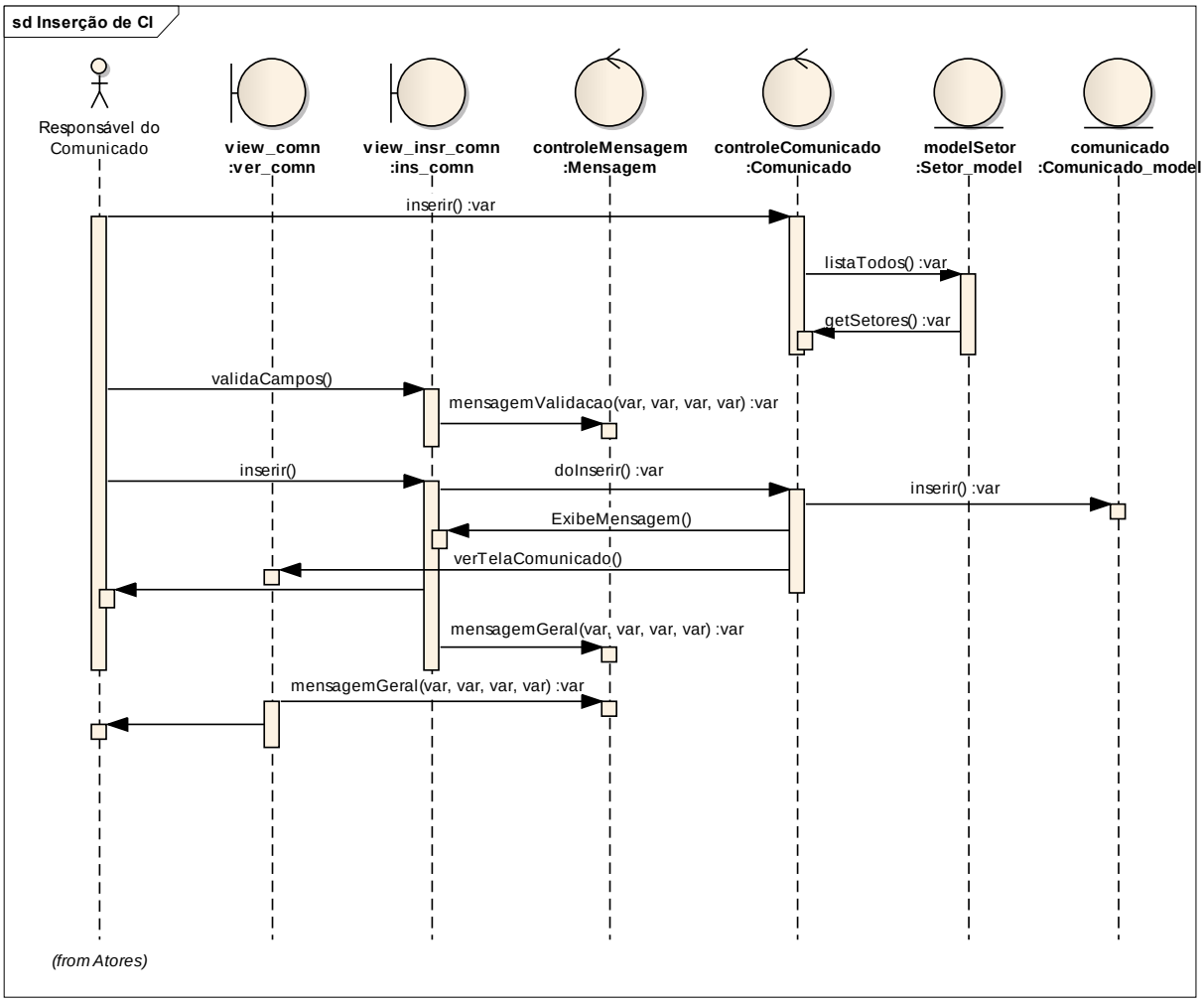


Figura 26 - Diagrama de Sequencia do RCSU 01.
Fonte: Elaborado pelo Autor.

6.4 VISÃO DE PROJETO

Na visão de projeto é apresentado à organização do sistema a fim de implementação, por exemplo, a Figura 27 detalha o sistema dividido em 3 (três) camadas, o *Model*, *View* e *Controller*.

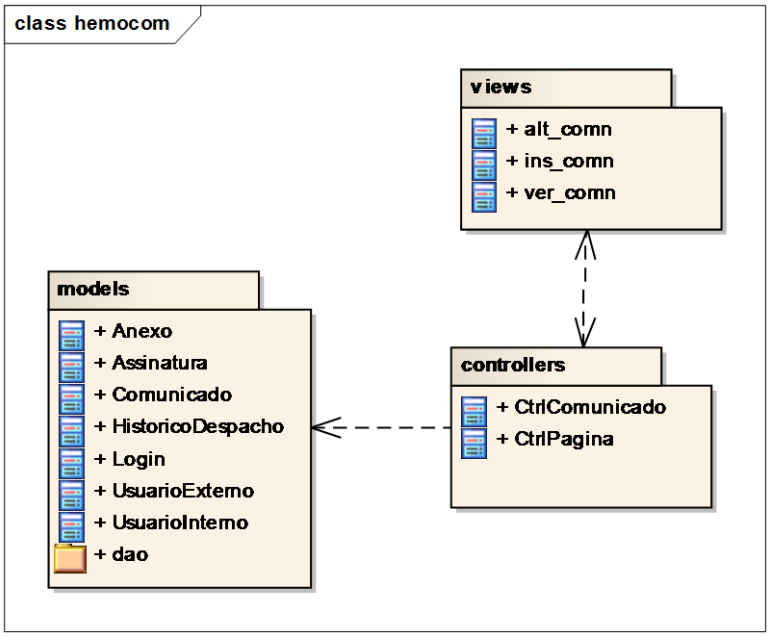


Figura 27 - Diagrama de pacotes das camadas do sistema.
Fonte: Elaborado pelo Autor.

6.4.1 DAO

Neste tópico é apresentado à implementação do *Design Pattern Abstract Factory*, conforme a Figura 28, camada que separa as regras de negócio das regras do banco.

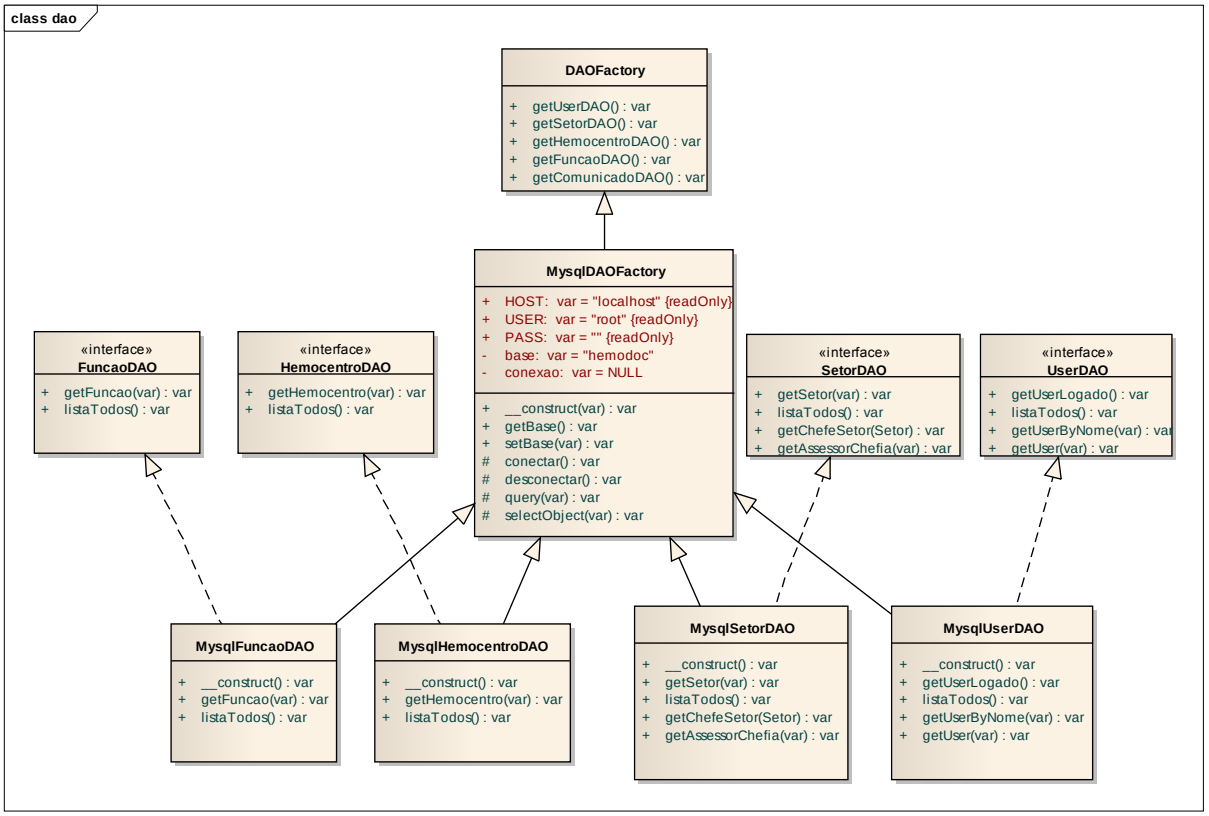


Figura 28 - Diagrama de Classe *Abstract Factory* na visão global do sistema.
Fonte: Elaborado pelo Autor.

6.5 VISÃO DE DADOS

A visão de dados é iniciada na etapa de elaboração e concluída na fase de construção, sendo de responsabilidade do Projetista do Banco de dados.

6.5.1 Modelo de Classe de Domínio

Neste diagrama, Entidade Relacionamento, é demonstrado a exibição dos modelos de banco utilizando a notação alternativa UML. De uma forma mais clara, seus detalhamentos exibem as responsabilidades de cada tabela através de suas ligações e multiplicidade.

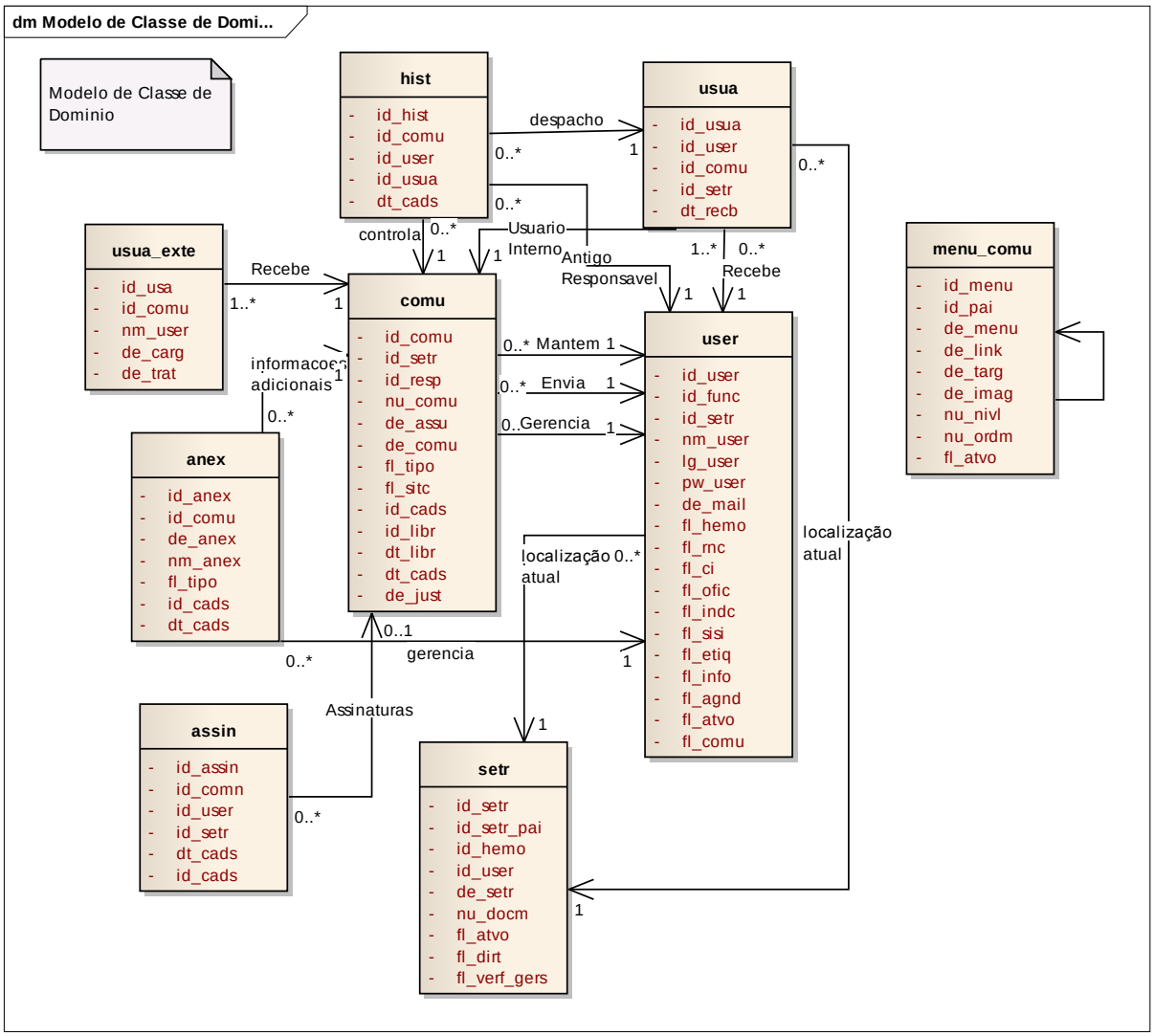


Figura 29 - Modelo de Classe de Domínio.
Fonte: Elaborado pelo Autor.

6.5.2 Modelo de Dados

O diagrama que define de uma forma mais especifica o banco de dados é o modelo relacional, conforme Figura 30, sua criação facilita a geração das tabelas, principalmente, com ajuda da ferramenta Case que gera toda a estrutura a partir do modelo criado.

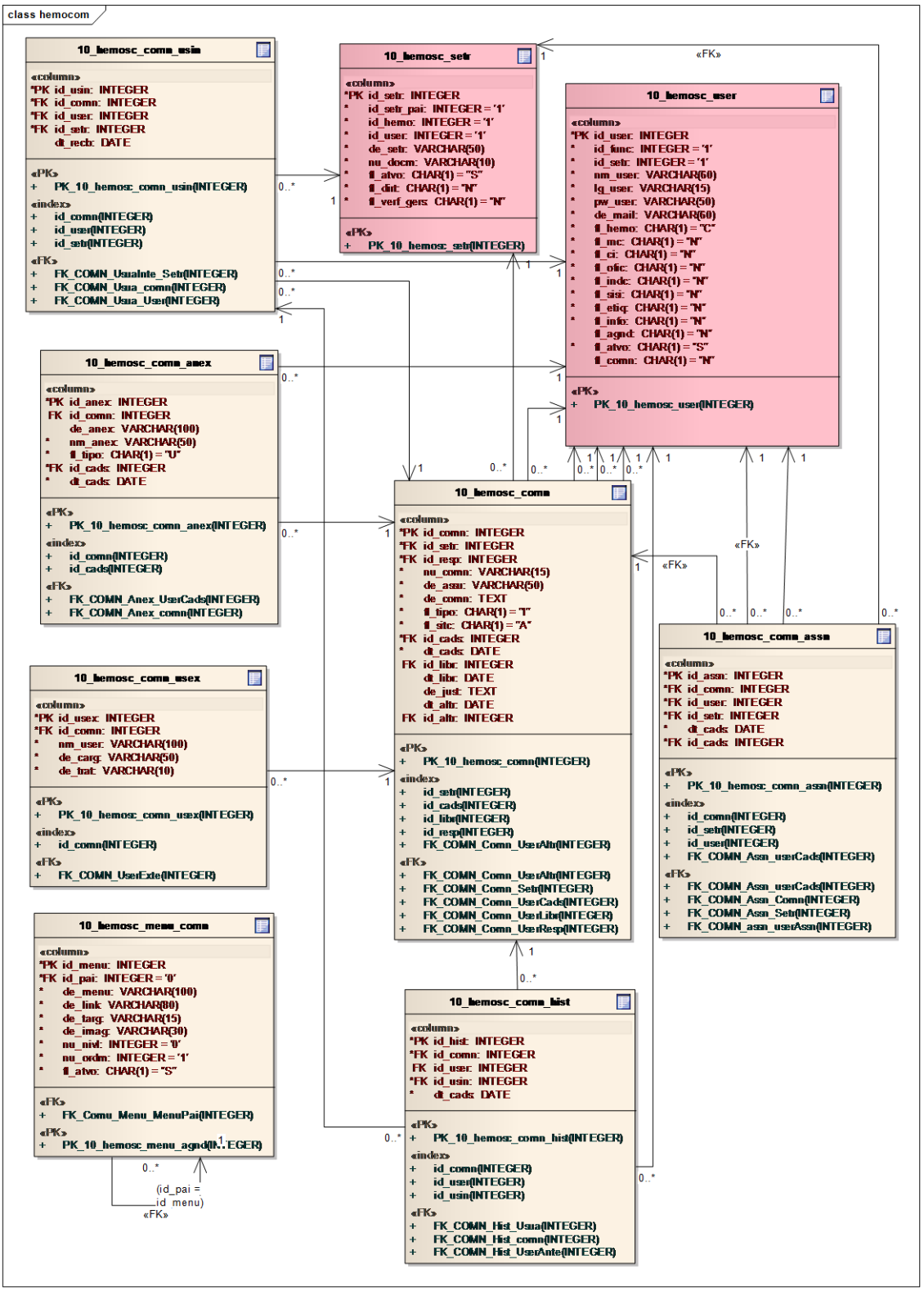


Figura 30 - Modelo de Dados.
Fonte: Elaborado pelo Autor.

6.6 VISÃO DE COMPONENTES

A visão de componentes, Figura 31, foi iniciada na fase de concepção com a descrição dos componentes externos e *software* e finalizada na fase de Construção com a definição dos Componentes do Sistema.

Os Componentes do Sistema são sistemas e/ou aplicações externas utilizados no desenvolvimento do sistema, o Jquery *framework* javascript utilizado na parte de animação das telas e validação de formulários e fPDF utilizado na geração de arquivos pdf.

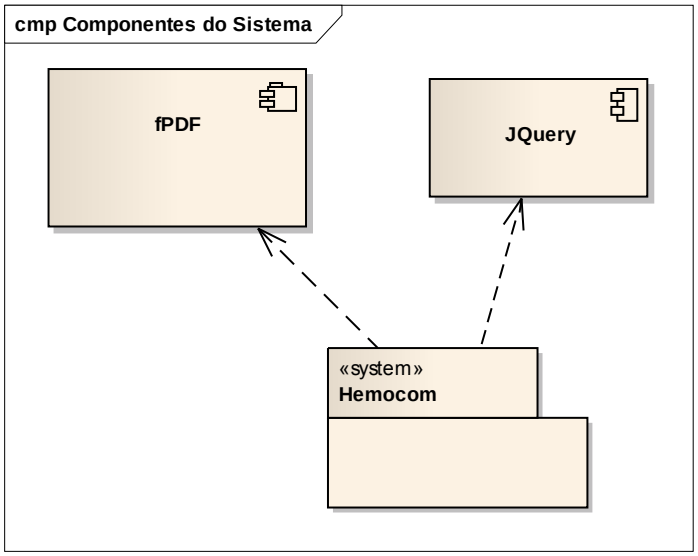


Figura 31 - Visão de Componentes.
Fonte: Elaborado pelo Autor.

Nos componentes externos estão as definições de database, servidor e web browsers utilizadas no projeto, serão mais detalhados na visão de distribuição, próximo tópico, assim como os softwares.

6.7 VISÃO DE DISTRIBUIÇÃO

Para melhor compreensão e uma definição completa, na visão de distribuição, foram elaborados diagramas que apresentam os diferentes ambientes que envolvem o processo de desenvolvimento desse projeto.

6.7.1 Topologia do Servidor

A topologia do servidor, Figura 32, apresenta os detalhes do ambiente de hardware e software que competem ao servidor, localizado dentro do Hemosc. Sendo este o ambiente real de produção de todos os sistemas desenvolvidos dentro da Intranet.

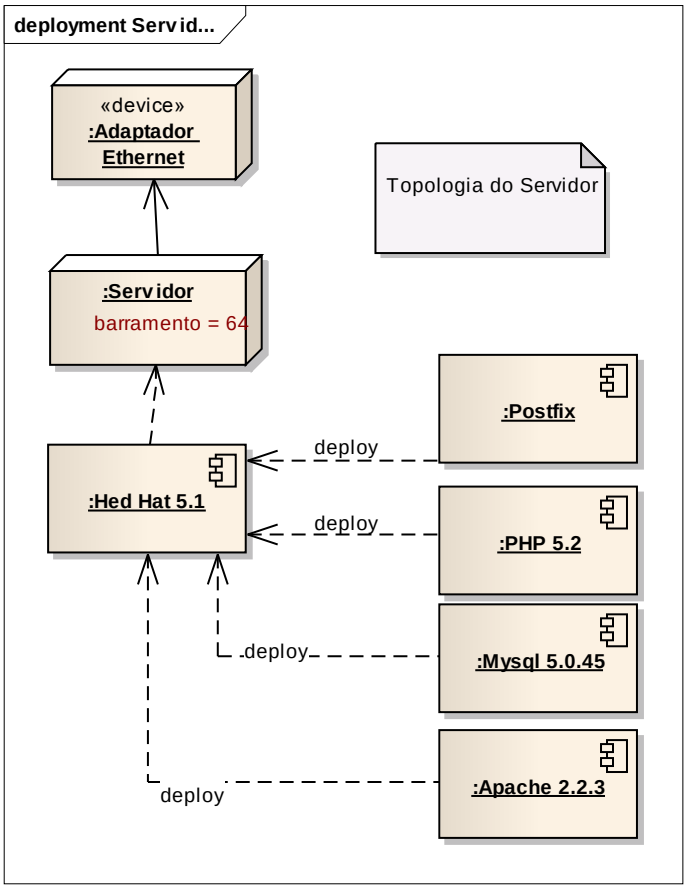


Figura 32 - Topologia do Servidor.
Fonte: Elaborado pelo Autor.

6.7.2 Topologia do Cliente

No ambiente do cliente nos dá uma idéia da capacidade de hardware e software disponibilizado na utilização do software, repassando informações de compatibilidade de softwares, como por exemplo, que por normal interna define o uso obrigatório do navegador Mozila Firefox.

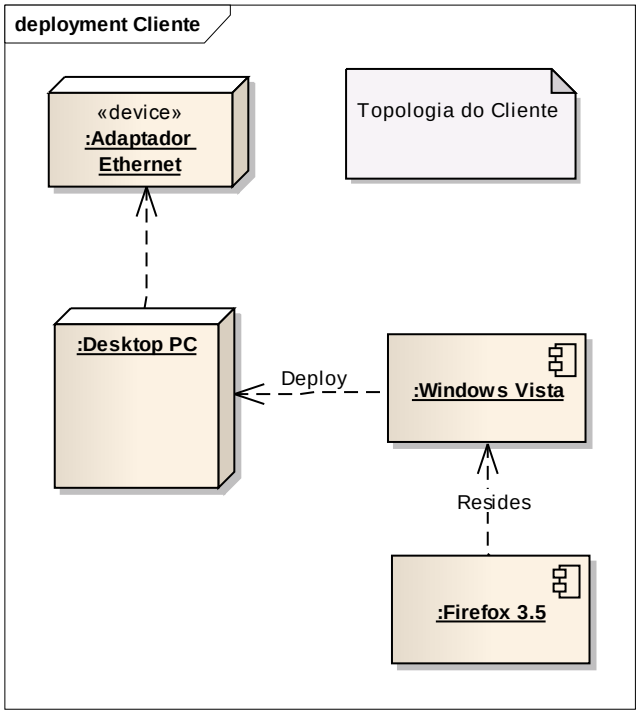


Figura 33 - Topologia do Cliente.
Fonte: Elaborado pelo Autor.

6.7.3 Ambiente de Desenvolvimento

O ambiente de desenvolvimento foi definido para chegar o mais próximo possível do ambiente do cliente e do servidor. Na Figura 34 estão detalhadas as ferramentas para desenvolvimento, softwares e hardwares utilizados.

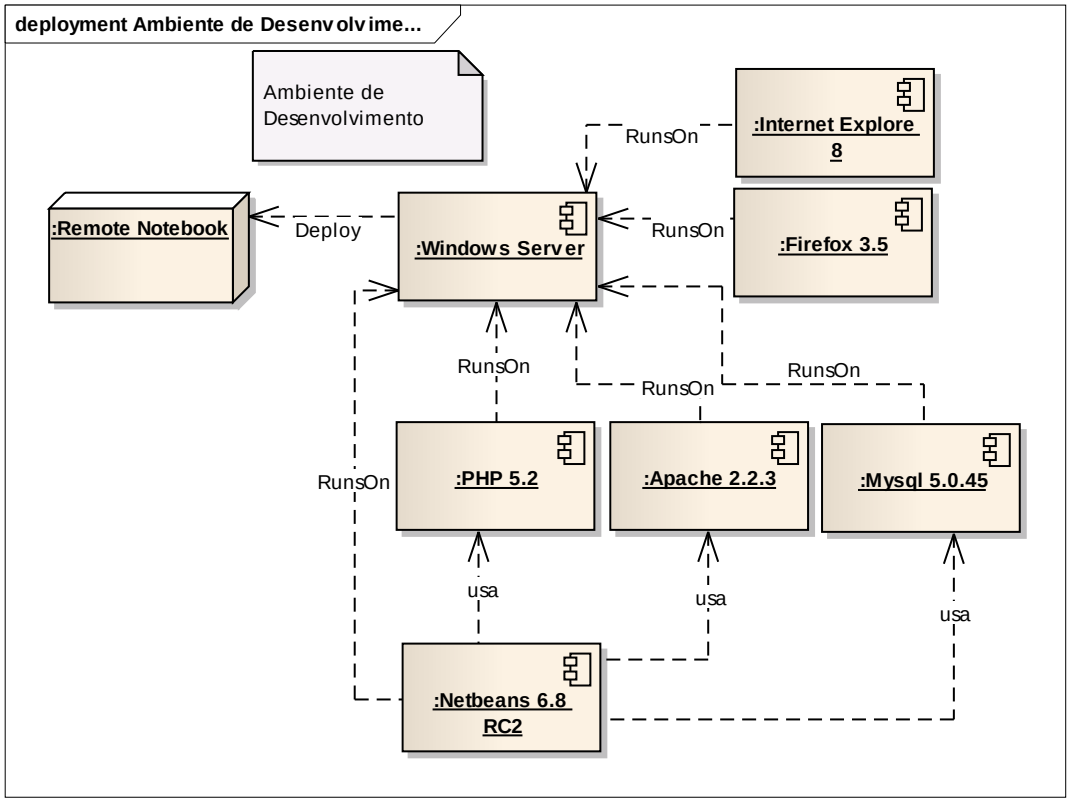


Figura 34 - Ambiente de desenvolvimento.
Fonte: Elaborado pelo Autor.

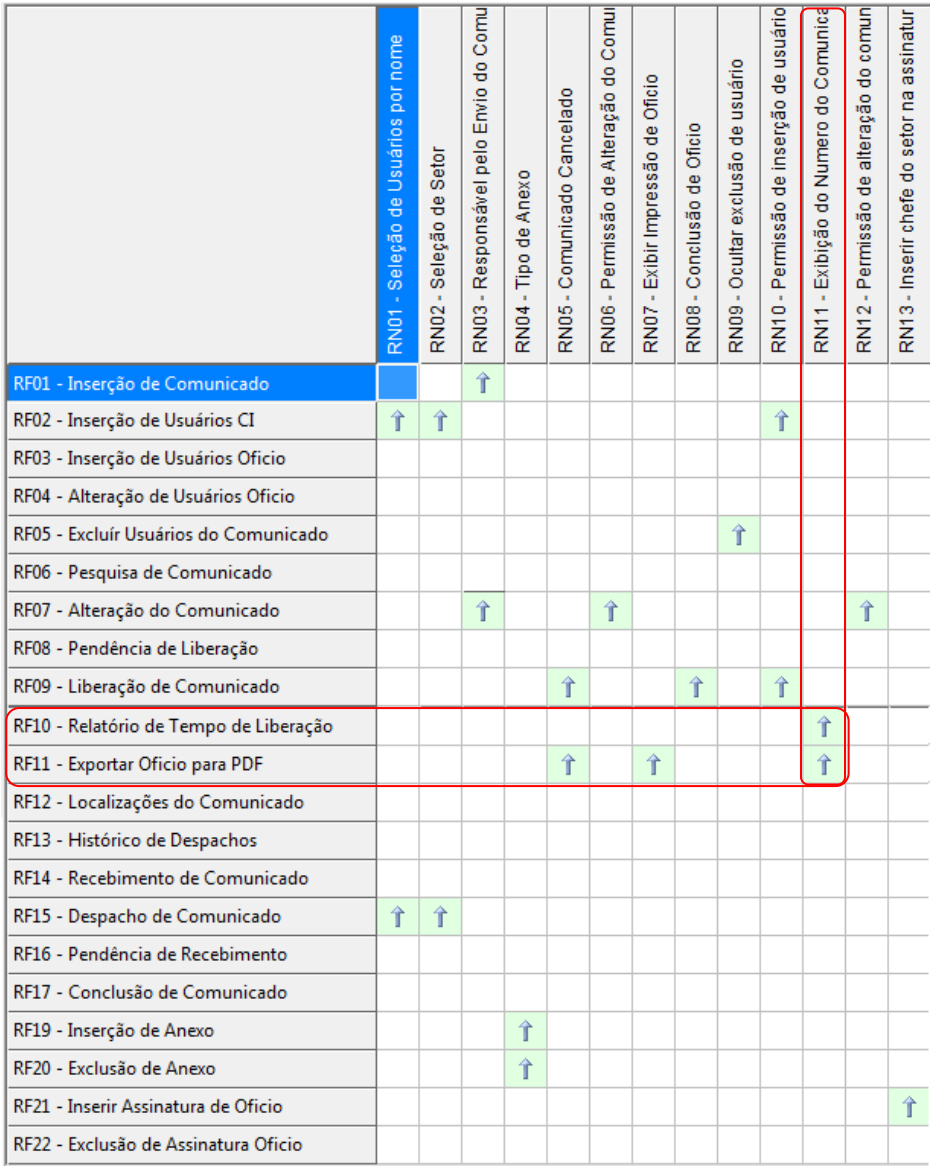
7 ANÁLISE DOS DADOS

Conforme apresentado nos tópicos anteriores a rastreabilidade deve ser iniciada ainda no início do projeto, e na Engenharia de Requisitos pertence à gerência de requisitos, logo esta ativa durante todos seus ciclos, do levantamento dos requisitos a verificação e posteriormente na manutenção de requisitos.

Através das ligações entre os artefatos, aos poucos, a rastreabilidade do software foi gerada paralelamente com a modelagem do sistema. No escopo deste projeto foram escolhidos alguns cenários para rastrear, o cenário de requisitos, cenário de análise e cenário de código fonte. Nos tópicos seguintes estarão detalhados na forma de matrizes de rastreabilidade.

7.1 MATRIZ DE RASTREABILIDADE

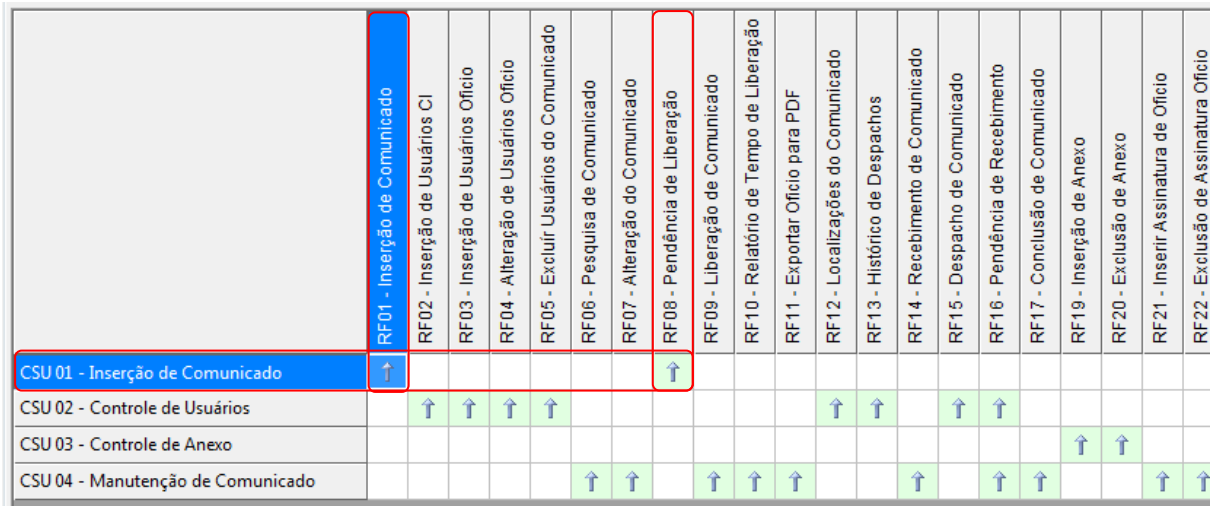
A ligação do relacionamento da ‘realização’ efetuada entre o Requisito Funcional e Regra de negócio gerou a matriz de rastreabilidade, conforme a Figura 35. Por exemplo, tanto o requisito funcional RF10 quanto o RF11 realizam a Regra de Negócio RN11 - Exibição do numero do comunicado, esta realização foi criada durante a análise de requisitos, sendo aprovado na verificação do requisito.



	RN01 - Seleção de Usuários por nome	RN02 - Seleção de Setor	RN03 - Responsável pelo Envio do Comu	RN04 - Tipo de Anexo	RN05 - Comunicado Cancelado	RN06 - Permissão de Alteração do Comu	RN07 - Exibir Impressão de Ofício	RN08 - Conclusão de Ofício	RN09 - Ocultar exclusão de usuário	RN10 - Permissão de inserção de usuário	RN11 - Exibição do Numero do Comunicad	RN12 - Permissão de alteração do comun	RN13 - Inserir chefe do setor na assinatur
RF01 - Inserção de Comunicado			↑										
RF02 - Inserção de Usuários CI	↑	↑								↑			
RF03 - Inserção de Usuários Oficio													
RF04 - Alteração de Usuários Oficio													
RF05 - Excluír Usuários do Comunicado									↑				
RF06 - Pesquisa de Comunicado													
RF07 - Alteração do Comunicado			↑			↑						↑	
RF08 - Pendência de Liberação													
RF09 - Liberação de Comunicado					↑			↑		↑			
RF10 - Relatório de Tempo de Liberação											↑		
RF11 - Exportar Oficio para PDF					↑		↑				↑		
RF12 - Localizações do Comunicado													
RF13 - Histórico de Despachos													
RF14 - Recebimento de Comunicado													
RF15 - Despacho de Comunicado	↑	↑											
RF16 - Pendência de Recebimento													
RF17 - Conclusão de Comunicado													
RF19 - Inserção de Anexo				↑									
RF20 - Exclusão de Anexo				↑									
RF21 - Inserir Assinatura de Oficio													↑
RF22 - Exclusão de Assinatura Oficio													

Figura 35 - Matriz de Rastreabilidade entre Requisitos Funcionais e Regras de Negócio.
Fonte: Elaborado pelo Autor.

A matriz de rastreabilidade gerada a partir da implementação dos requisitos funcionais pelo caso de uso segue na Figura 36. Se analisado o caso de uso CSU 01 - Inserção de Comunicado, pode-se perceber a realização com os requisitos RF01 e RF08. Tanto esta matriz quanto a anterior correspondem ao cenário de requisitos.



	RF01 - Inserção de Comunicado	RF02 - Inserção de Usuários CI	RF03 - Inserção de Usuários Ofício	RF04 - Alteração de Usuários Ofício	RF05 - Excluir Usuários do Comunicado	RF06 - Pesquisa de Comunicado	RF07 - Alteração do Comunicado	RF08 - Pendência de Liberação	RF09 - Liberação de Comunicado	RF10 - Relatório de Tempo de Liberação	RF11 - Exportar Ofício para PDF	RF12 - Localizações do Comunicado	RF13 - Histórico de Despachos	RF14 - Recebimento de Comunicado	RF15 - Despacho de Comunicado	RF16 - Pendência de Recebimento	RF17 - Conclusão de Comunicado	RF19 - Inserção de Anexo	RF20 - Exclusão de Anexo	RF21 - Inserir Assinatura de Ofício	RF22 - Exclusão de Assinatura Ofício
CSU 01 - Inserção de Comunicado	↑							↑													
CSU 02 - Controle de Usuários		↑	↑	↑	↑							↑	↑		↑	↑					
CSU 03 - Controle de Anexo																		↑	↑		
CSU 04 - Manutenção de Comunicado						↑	↑		↑	↑	↑			↑		↑	↑			↑	↑

Figura 36 - Matriz de Rastreabilidade entre Caso de Uso e Requisito Funcional.
Fonte: Elaborado pelo Autor.

A matriz de rastreabilidade da Figura 37 representa a realização dos casos de uso, entrando na visão de analise e, portanto os diagramas as seguir correspondem ao cenário de análise.

	CSU 01 - Inserção de Comunicado	CSU 02 - Controle de Usuários	CSU 03 - Controle de Anexo	CSU 04 - Manutenção de Comunicado
RCSU 01 - Inserção de Comunicado	↑			
RCSU 02 - Controle de Usuários		↑		
RCSU 03 - Controle de Anexo			↑	
RCSU 04 - Liberação do Comunicado				↑

Figura 37 - Matriz de Rastreabilidade entre Colaboração e Caso de Uso.
Fonte: Elaborado pelo Autor.

A matriz criada do relacionamento dos diagramas de robustez e realização do caso de uso exibe a colaboração e seus objetos.

	RCSU 01 - Inserção de Comunicado	RCSU 02 - Controle de Usuários	RCSU 03 - Controle de Anexo	RCSU 04 - Liberação do Comunicado
comunicado:Comunicado_model	↑			
controleComunicado:Comunicado	↑			
controleMensagem	↑			
modelSetor:Setor_model	↑			
view_comn:ver_comn	↑			
view_insr_comn:ins_comn	↑			

Figura 38 - Matriz de Rastreabilidade entre Colaboração e Objeto.
Fonte: Elaborado pelo Autor.

O ultimo nível que a rastreabilidade apresenta seria da realização do caso de uso com o código fonte, apresentado na Figura 38, criado a partir do diagrama de sequência para a RCSU 01, apresentados no tópico de Modelagem do Sistema.

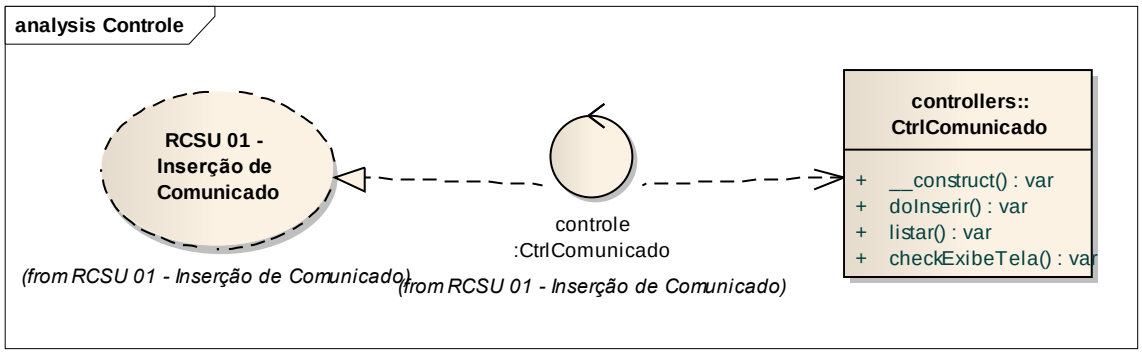


Figura 39 – Arquitetura da Camada de Controle ligando RCSU01 e CtrlComunicado.
Fonte: Elaborado pelo Autor.

O relacionamento apresentado anteriormente gerou a matriz de rastreabilidade apresentado na Figura 40, que exibe o *trace link* entre Objeto e as classes do sistema, ou seja, o código fonte.

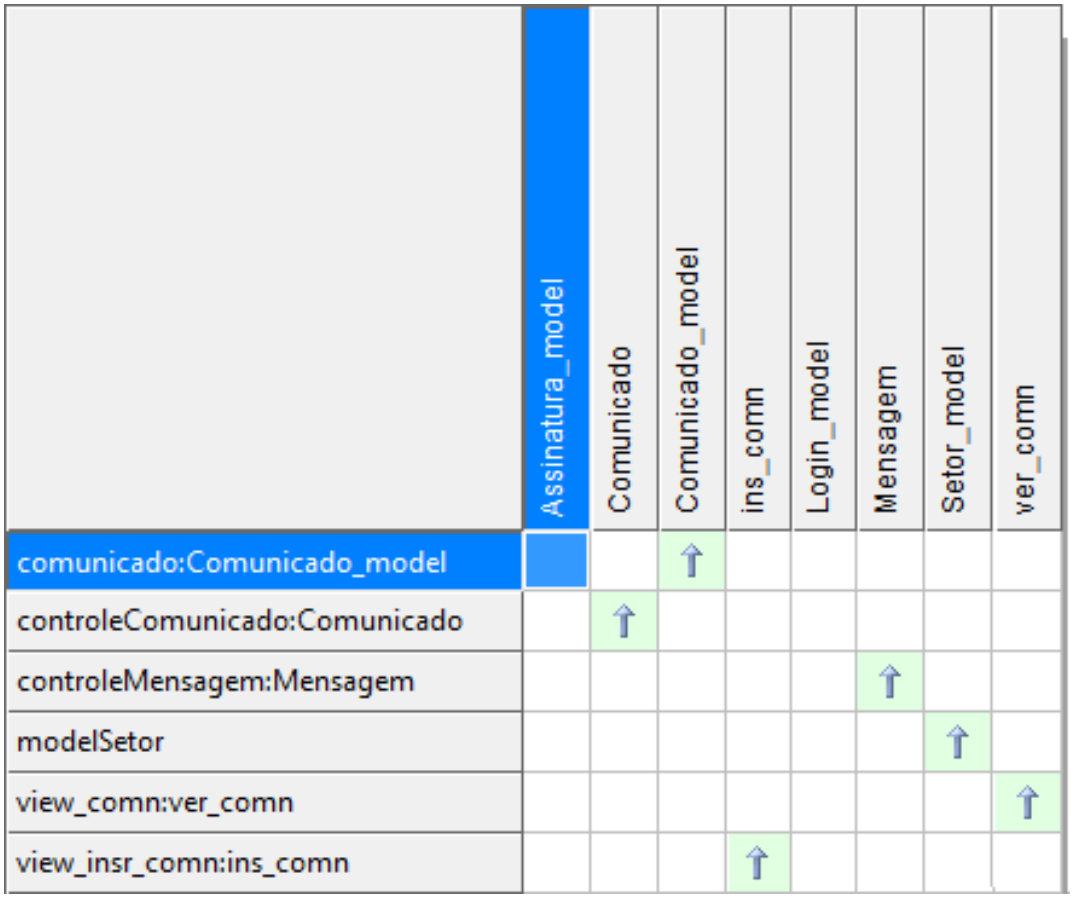


Figura 40 - Matriz de Rastreabilidade entre Objeto e classe.
Fonte: Elaborado pelo Autor.

O cenário de código, descreve o nível de implementação do sistema, portanto quanto executado os cenários de requisitos e análise resultam no cenário de código, nível atingimos com a figura anterior.

7.2 RASTREABILIDADE VERTICAL

A rastreabilidade vertical exibe todos os links do requisito ao código fonte em ambos os sentidos. No *Enterprise Architect* essa visão vertical de rastreabilidade é exibido através da Hierarquia.

Selecionando o Requisito Funcional RF01 – Inserção de Comunicado, conforme exibido na Figura 41, teríamos os locais impactados por alteração do RF01. Entre eles estão o CSU 01, em seguida RCSU 01, os objetos Comunicado, Controle e

Frm_cadastro_comunicado e finalmente chegando ao código fonte, na classe CtrlComunicado, tela ins_comn e model Comunicado.

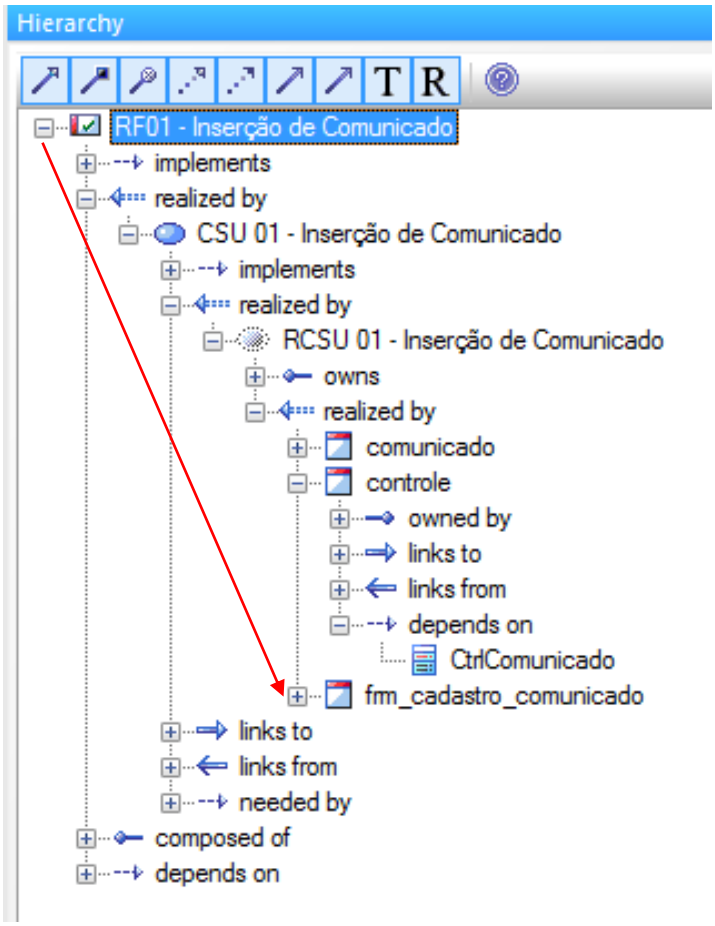


Figura 41 – Diagrama hierárquico do Requisito Funcional RF01 ao código fonte.
Fonte: Elaborado pelo Autor.

No sentido contrário, ou seja, do código fonte aos requisitos foi selecionado a tela de inserção do comunicado (ins_comn) a visão será a mesma, com a única diferença que o *trace link* se dá pela implementação (*implements*) e não realização como na hierarquia anterior.

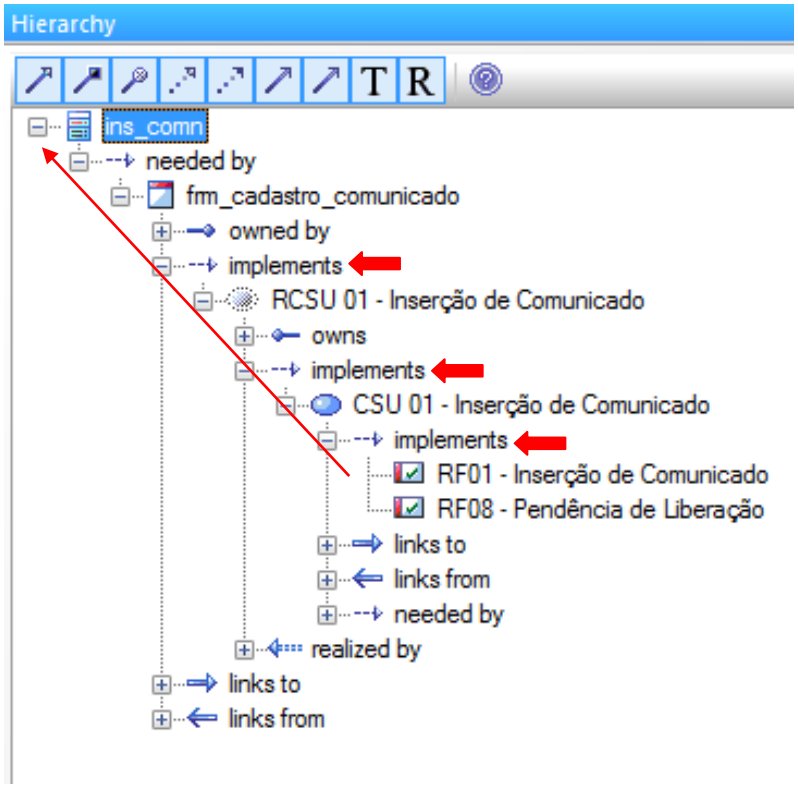


Figura 42 - Hierarquia do código fonte ao Requisito Funcional.
Fonte: Elaborado pelo Autor.

7.3 ALTERAÇÃO DE REQUISITOS

Durante o desenvolvimento do projeto ocorrem duas situações de alteração de requisitos que cabem neste processo destacar a criação de dois novos requisitos e a incerteza quanto ao requisito Despacho de Comunicado.

Na primeira situação durante a fase verificação dos requisitos o cliente identificou a necessidade de cadastrar quem assinaria os ofícios (um dos tipos de comunicado), portanto dois novos requisitos inserir e excluir assinatura nos comunicados do tipo ‘Ofício’.

A fase de desenvolvimento atual era a elaboração e o processo de Engenharia de Requisitos validação e, portanto os únicos artefatos criados até aquele momento eram dos requisitos e caso de uso.

Para identificar quais alterações seriam necessárias, foi selecionado o caso de uso que implementaria os novos requisitos, sendo possível ter a visão geral sobre o impacto destas inserções. Como estava ainda nas fases iniciais o impacto sobre as alterações foi muito pequeno, entretanto foi possível identificar com maior facilidade as alterações necessárias.

A segunda situação do Requisito Funcional – Despacho do Comunicado é uma situação um pouco mais delicada, pois este processo ainda não se apresenta de forma clara e definitiva para o cliente. Portanto, ficou definido que sua implementação se dará da forma mais simples e posteriormente durante a utilização do sistema (já em execução) será identificado qual a sequência correta do requisito.

Analisando esta situação, possivelmente haverá a necessidade de alterações após a aprovação do sistema. Com a Figura 43 podemos ter uma idéia sobre o impacto que o requisito tem na fase atual que o projeto se encontra, por exemplo, no CSU 02, RCSU02 e o RF13. Em se tratando de um requisito volátil, definimos não aprofundar no seu detalhado e, portanto o nível de código não foi especificado até este momento.

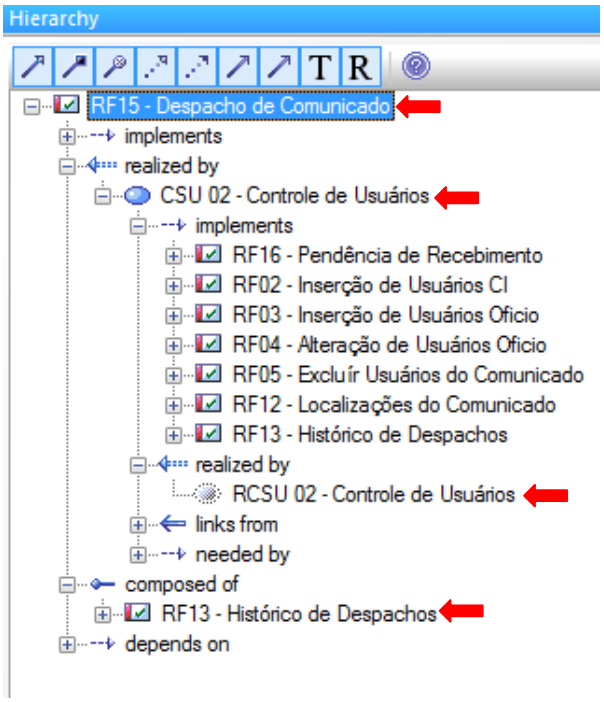


Figura 43 - Rastreabilidade do Requisito RF15
Fonte: Elaborado pelo Autor.

8 CONCLUSÕES E TRABALHOS FUTUROS

8.1 CONCLUSÕES

A utilização da rastreabilidade no processo de desenvolvimento de software foi possível identificar o impacto no sistema por mudanças de requisitos, obtendo uma documentação de software interligada, dos requisitos ao código fonte.

Diferentemente do que muitos autores destacam a rastreabilidade é o processo simples de ser implementado e gerenciado desde que sua utilização seja do início do projeto e com a utilização de uma ferramenta CASE. Por exemplo, a ferramenta *Enterprise Architect*, que se apresentou a ferramenta adequada para auxiliar na rastreabilidade.

A simplicidade na implementação da rastreabilidade esta fortemente acoplada à execução da modelagem do sistema, uma vez que a modelagem seja feita e **corretamente**, a rastreabilidade passa a ser a intersecção dos elos (relacionamentos) na modelagem.

Com o *Enterprise Architect*, foi possível centralizar todos os processos de desenvolvimento em uma única ferramenta *CASE*, na modelagem do sistema e entre outras operações, como por exemplo, a geração de código fonte a partir dos modelos desenvolvidos e estrutura do banco de dados.

Os cenários definidos para constar na documentação da rastreabilidade são os cenários de requisitos, os cenários de análise e os cenários de código, apresentados na forma de matrizes de rastreabilidade. Matrizes formadas através dos relacionamentos entre os cenários, possível somente por os cenários utilizarem a mesma linguagem, UML, para sua definição.

Com o desenvolvimento do sistema protótipo foi possível validar a arquitetura proposta e, portanto, identificar com maior facilidade o impacto no sistema em situações de modificação de requisitos, através da visibilidade proporcionada pela rastreabilidade.

8.2 TRABALHOS FUTUROS

As sugestões para trabalhos futuros estão a seguir:

- definir um processo de rastreabilidade para outras metodologias de desenvolvimento, como por exemplo, Scrum ou Iconix;
- verificar se o uso de rastreabilidade efetivamente aumenta a qualidade do processo de desenvolvimento e o software realizado.

REFERÊNCIAS

ALENCAR, Ivan; **Gerenciamento de Requisitos com Casos de Uso**: Melhores práticas da Engenharia de *Software*; *Rational Software*: 2006.

AQUINO, Rodrigo P.; Processo Unificado Integrado ao Desenvolvimento Web; **Engenharia de *software Magazine***: Rio de Janeiro, n. 01, p. 28 – 37, 2009. Disponível em: <http://www.devmedia.com.br>. Acesso em ago. 2009.

ÁVILA, Ana L; SPÍNDOLA, Rodrigo O.; Introdução a Engenharia de Requisitos; **Engenharia de *software Magazine***; Rio de Janeiro, n. 1, p. 46 – 52, 2007. Disponível em: <http://www.devmedia.com.br>. Acesso em jul. 2009.

BACKES, Jerônimo; Rastreabilidade: Conceitos Iniciais; **Engenharia de *software Magazine***: Rio de Janeiro, n. 13, p. 24 – 34, 2009. Disponível em: < http://www.devmedia.com.br>. Acesso em ago. 2009.

BEHRENS, Thomas; *Never 'Without a trace': Practical advice on implementing traceability;* **IBM Global Services**: feb. 2007. Disponível em: <http://www.ibm.com >. Acesso em out. 2009.

BOENTE, Alfredo P; OLIVEIRA, Fabiano G.; ALVEZ, João N.; **RUP como Metodologia de Desenvolvimento de Software para Obtenção da Qualidade de Software**. In: Simpósio de Excelência em Gestão e Tecnologia - SEGeT, 2008, Resende: Ed. AEDB, 2008.

CLELAND-HUANG, J.; ***Just Enough Requirements Traceability***. In: *Annual International Computer Software And Applications Conference, Compsac,* 30., 2006, Chicago, Illinois. Los Alamitos, CA: IEEE *Computer Society*, 2006.

COSTA, Marcelo; KALINOWSKI, Marcos; SPÍNDOLA, Rodrigo O.; Soluções Concretas para Problemas Práticos da Engenharia de Requisitos; **Engenharia de *Software Magazine***: Rio de Janeiro, n. 13, p. 08–15, 2009. Disponível em: <www.devmedia.com.br>. Acesso em ago. 2009.

CRUZ, Jorge L; JINO, Mário; CRESPO, Adalberto N.; ARGOLO, Miguel; **Suporte automatizado à rastreabilidade em um processo de teste de software baseado em documentação**. Simpósio Brasileiro de Qualidade de Software; Vila Velha: 2006.

DATASUS; **PDS-DATASUS**: Processo de desenvolvimento de Software; Ministério da Saúde, 2009. Disponível em: <http://pds.datasus.gov.br/PDS >. Acesso em mar. 2010.

ESPÍNDOLA, Rodrigo S.; MAJDENBAUM, Azriel; AUDY, Jorge L. N.; **Uma Análise Crítica dos Desafios para Engenharia de Requisitos em Manutenção de Software;** Pontifícia Universidade Católica do Rio Grande do Sul: Programa de Pós-Graduação em Ciência da Computação; 2004.

FABRI, José A.; **A Rastreabilidade dos Requisitos como importante fator da garantia de qualidade**: 2009; Disponível em: <http://engenhariasoftware.wordpress.com>. Acesso em: ago. 2009

FISHER, Alan S. **CASE**: utilização de ferramentas para desenvolvimento de software. Rio de Janeiro: Campus, 1990. 264 p.

GENVIGIR, E., VIJAYKUMAR, N.. Uma Proposta de Modelagem para a Generalização de Elos de Rastreabilidade. **Revista de Informática Teórica e Aplicada**, América do Norte, 15, dec. 2008. Disponível em: <http://www.seer.ufrgs.br/index.php/rita/article/view/rita_v15_n2_p181-202/4491>. Acesso em: 24 Apr. 2010.

GIL, Antônio Carlos**. Como elaborar projetos de pesquisa.** São Paulo: Atlas, 1991.

HULL, Elizabeth; JACKSON, Ken; DICK, Jeremy; ***Requirements Engineering***. Springer, ed 2. 2005.

HUSITA, Elisa H. **Processo de Desenvolvimento (UP/RUP)**. Disciplina de Engenharia de Software na Universidade Estadual de Maringá, Centro de Tecnologia Departamento de Informática; Maringá: 2003.

IEEE; **830-1998**: *IEEE Recommended Practice for Software Requirements Specications;* 1997. Disponível em: <http://standards.ieee.org/>. Acesso em: ago. 2009.

KROLL, P. E KRUCHTEN, Philippe; "***The Rational Unified Process made easy:*** *a practitioner's guide to the RUP*"; 2003.

LEAL, Pedro L. R. Jr; **Requisitos de Métodos de Rastreabilidade entre os Requisitos e o Código de Software;** Departamento de Ciência da Computação – Universidade Federal de Minas Gerais: Pampulha, 2007.

LEAL, Marcelio D´O; OLIVEIRA, Sandro R. B; SOUZA, Cleidson R. B; In 2ª edição do Workshop *on Requirements Engineering* – WER, 2009, Valparaíso-Chile; **Uma Avaliação sobre Rastreabilidade de Software no Contexto do MPS.BR;**

LEITE, Júlio. C. S. P. et al. *Enhancing a requirements baseline with scenarios*. ***Proc. Of the Third IEEE International Symposium on Requirements Engineering (RE'97)***, IEEE *Computer Society Press*, 1997. p. 44-53.

LEITE, Júlio. C. S. P. **Gerenciando a Qualidade de Software com Base em Requistos;** 2001. Disponível em < http://www.inf.puc-rio.br/~julio>. Acesso ago. 2009

MILLER, Thomas R. ***Requirements Engineering and Analysis - Workshop Proceedings***: *Requirements Engineering Project***;** *Software Engineering Institute*: Carnegie Mellon University; Pittsburgh, Pennsylvania: 1991

MORAES, Janaína; Introdução a abordagens de identificação de Requisitos; **Engenharia de *Software Magazine***: Rio de Janeiro, n. 2, p. 54–58, 2007. Disponível em: <www.devmedia.com.br>. Acesso em ago. 2009.

MORAGA, Alejandro F.; **Engenharia de Software**; Disponível em: <www.moraga.com.br/73/engenharia-de-software>. Acesso em ago. 2009.

PAULA, Wilson P. Filho; **Alguns Fundamentos da Engenharia de *Software***; Engenharia de *Software Magazine*; Rio de Janeiro, n. 1, p. 04–09, 2007. Disponível em: <www.devmedia.com.br>. Acesso em jul. 2009.

PRESSMAN, Roger S. **Engenharia de Software**. 6. ed. São Paulo: Ed. McGrawHill, 2006.

RATIONAL SOFTWARE; *Rationa Unified Process*: 2002; Português v. 2002.05.00; Disponível em: <http://www.wthreex.com/rup/portugues/>; Acesso em out. 2009.

SANTOS, Fernando dos; VARGAS, Karly S.; ABREU, Christian R. C.; **Rastreabilidade de Requisitos através da web;** In: SEMINÁRIO DE COMPUTAÇÃO Universidade Regional de Blumenau - FURB, XIII, 2004, Blumenau. Blumenau: p. 02-04

SILVA, Lyrene F. **Uma Estratégia Orientada a Aspectos para Modelagem de Requisitos**. 2006. 222f. Tese (Doutorado em Ciências) – Curso de Pós-graduação Informática do Departamento de Informática, Pontifícia Universidade Católica, Rio de Janeiro, 2006.

SOMMERVILLE, Ian; SAWYER, Peter; ***Requirements Engineering:*** *a good practice guide*; New York: John Wiley & Sons Ltd, 1997.

SOMMERVILLE, Ian; **Engenharia de Software**; Person Education do Brasil: 7ª ed. 2005.

SOMMERVILLE, Ian; **Engenharia de Software**; Person Education do Brasil: 6ª ed. 2003.

SOUZA, Vinicius L.; **Desenvolvimento de *Software* Dirigido por Caso de Uso**: Conceituando e entendendo Caso de Uso; Engenharia de *software Magazine*; Rio de Janeiro, n. 02, p. 36 – 40, 2009. Disponível em: <www.devmedia.com.br>. Acesso em ago. 2009.

SPARTX SYSTEMS; Disponível em: <www.sparxsystems.com.au/>. Acesso em: out. 2009

VIDAL, Raul; FARIA, João; AGUIAR, Ademar; GONÇALVES, Gil; **Processos de Desenvolvimento de *Software;*** Universidade do Porto Faculdade de Engenharia – FEUP/LEIC – Laboratório de Gestão de Projetos: 2003-04;

VLIET, Hans V.; ***Software Engineering: Principles and Practic*;** Wiley: 2007.

APÊNDICE

APÊNDICE A – Visão do Negócio

HEMOCOM

Visão do Negócio

Versão 1.5

Histórico da Revisão

Data	Versão	Descrição	Autor
19/abr/2010	1.0	Reunião Inicial	Adriana
23/abr/2010	1.5	Revisão com o Hemosc	Adriana

INTRODUÇÃO

O sistema tem a finalidade de gerenciar o envio de comunicados internos e externos para os colaboradores do Hemosc. O principal objetivo é atingir o maior numero de pessoas em um menor tempo, em conseqüência, agilizando o procedimento que, atualmente, acontece de forma manual.

DESCRIÇÃO DO PROBLEMA

O problema de	Controle de comunicação (colaboradores e externo) e impressão descontrolada de arquivos.
Afeta	O gerenciamento e envio de comunicação dentro e fora do HEMOSC.
cujo impacto é	Grande esforço para documentar as comunicações, custo nas impressões de documentos e organização.
uma boa solução seria	O Envio dos comunicados de forma digital e com acesso através da Intranet.

SENTENÇA DE POSIÇÃO DO PRODUTO

Para	Os Colaboradores do Hemosc
Que	Recebem e enviam comunicados internos e externos diariamente
O Hemocom	é um sistema de gerenciamento de processos
Que	Diminui o esforço no envio e recebimento de comunicados e desvincula a impressão dos comunicados através do acesso pela web.
Ao contrário de	Do processo efetuado atualmente.
Nosso produto	Gerencia as comunicações de acordo com o procedimento ideal proposto pelo Hemosc.

DESCRIÇÕES DOS ENVOLVIDOS E DOS USUÁRIOS

Os usuários são colaboradores do Hemosc, e entre eles estão chefe do setor, assessor da chefia, secretariado da direção e colaboradores comuns.

RESUMO DOS ENVOLVIDOS

Nome	Descrição	Responsabilidades
Diretorias (Gerencia Técnica, Gerencia Administrativa e Direção Geral)	Controlam e gerenciam os processos do Hemosc	Gerenciamento dos comunicados internos (CI) e externos

RESUMO DOS USUÁRIOS

Nome	Descrição	Responsabilidades	Envolvido
Secretaria	Administrador	- gerenciamento do sistema - elabora relatórios - coordenação das operações - inserção de comunicados	Representante das Direções
Colaborador	Acesso básico ao sistema	- inserção de comunicados - recebimento de comunicados	Na maioria, acessará o sistema para visualizar os comunicados
Chefe do Setor	Gerentes	- liberação - despacho - inserção de comunicados	Responsável pelo setor ao qual ele pertence
Assessor da Chefia	Gerentes	- liberação - despacho - inserção de comunicados	Atua como substituto do chefe do setor

AMBIENTE DO USUÁRIO

O Hemocom se trata de um sistema de comunicados, logo, tem o objetivo de atingir 100% dos colaboradores do Hemosc.

No mínimo, uma tarefa envolve cerca de 3 (três) pessoas e no máximo todos ao mesmo tempo. Portanto, a duração de um ciclo de tarefa se dará mediante a quantidade de pessoas envolvidas e a quantidade de encaminhamento (despacho) que se dará no decorrer do processo.

Para cada usuário comum nas entradas diárias terá um tempo muito baixo de utilização, pois será a confirmação do recebimento do comunicado. Já para usuários com permissão de gerenciamento como inserção de comunicado e liberações o tempo pode variar proporcionalmente a quantidade de usuários envolvidos.

O sistema é acessado através da Intranet do Hemosc, por qualquer Hemocentro e/ou locais com acesso à internet.

PRINCIPAIS NECESSIDADES DOS USUÁRIOS OU DOS ENVOLVIDOS

Necessidade	Prioridade	Preocupações	Solução Atual	Soluções Propostas
Leitura do comunicado	Alta	Recebimento em tempo o comunicado	Recebimento de comunicado impresso.	Comunicado eletrônico, com acesso através da Intranet.
Controle de comunicado	Alto	Confirmação de leitura dos comunicados	Assinatura de protocolo de recebimento	Confirmação através de um clique
Localização do comunicado	Alto	Saber o local aonde demora mais para dar retorno do comunicado	Entrar em contato com todos os envolvidos no comunicado	Visualizar automaticamente no sistema a atual situação do comunicado.
Padrão de documento	Média	Garantir o padrão dos documentos	Disponibilizar documentos de modelo na intranet	Gerar automaticamente os comunicados sem permissão para alterar o modelo obrigatório
Impressão dos comunicados	Média	Redução em 100% das impressões descontroladas de CI’s	Impresso e entregue em mãos o documento	Acesso a qualquer momento através da Intranet

APÊNDICE B – Glossário de Negócio

HEMOCOM
Glossário de Negócios
Versão 2.0

Histórico da Revisão

Data	Versão	Descrição	Autor
20/04/2010	1.0	Conceitos iniciais do sistema	Adriana Zenck
23/04/2010	2.0	Revisado com o Hemosc	Adriana Zenck

Definições

Assessor da Chefia
Colaborador que responde pelo Chefe do Setor, se necessário.

Chefe do Setor
Colaborador que responde por todo o setor.

CI
Comunicação Interna do Hemosc.

Oficio
As comunicações externas são no Hemosc utilizando o modelo de ofícios.

CPQ
Coordenadoria de Planejamento e Qualidade. A coordenação responsável pelo planejamento deste sistema. O CPQ pertence ao Hemocentro Coordenador em Florianópolis.

CQ
Coordenadoria de Qualidade. Pertencentes nos Hemocentros Regionais.

Despacho
Dar encaminhamento de um comunicado para responsabilidade de outro(s) colaborador(es).

Hemosc
Centro de Hematologia e Hemoterapia de Santa Catarina

Hemocentro Regional
Unidades auxiliares do Hemocentro Coordenador.

Hemorrede
Regiões coordenadas pelo Hemosc, atualmente, o Hemosc coordena a Hemorrede Pública do Estado de Santa Catarina, sendo os hemocentros localizados em Lages, Joaçaba, Chapecó, Criciúma, Joinville e Blumenau.

Secretaria
Representante das Diretorias e estão divididas em Gerencia técnica, Gerencia Administrativa e Direção Geral.