

UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIAS DA COMPUTAÇÃO
DISSERTAÇÃO DE MESTRADO

**GERENCIAMENTO DE REQUISITOS NO
AMBIENTE COCAR**

André Di Thommazo

São Carlos
Janeiro/2007

**Ficha catalográfica elaborada pelo DePT da
Biblioteca Comunitária da UFSCar**

D615gr

Di Thommazo, André.

Gerenciamento de requisitos no ambiente Cocar / André Di Thommazo. -- São Carlos : UFSCar, 2008.
149 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2007.

1. Engenharia de requisitos. 2. Gerenciamento de requisitos. 3. Rastreabilidade de requisitos. 4. Modelo de casos de uso.I. Título.

CDD: 005.1 (20^a)

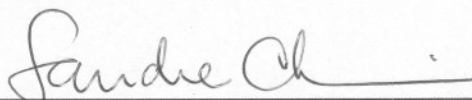
Universidade Federal de São Carlos
Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

“Gerenciamento de Requisitos no Ambiente COCAR”

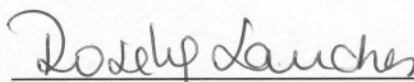
ANDRÉ DI THOMMAZO

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

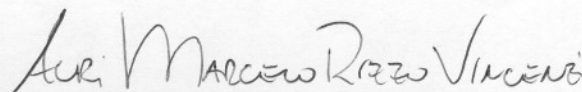
Membros da Banca:



Prof. Dra. Sandra Camargo P. Ferraz Fabbri
(Orientadora – DC/UFSCar)



Prof. Dra. Rosely Sanches
(ICMC/USP)



Prof. Dr. Auri Marcelo Rizzo Vincenzi
(UNISANTOS)

São Carlos
Janeiro/2007

DEDICATÓRIA

À minha família e namorada,
pela ajuda e compreensão de vocês.

AGRADECIMENTOS

Primeiramente, agradeço a Deus por permitir a realização deste trabalho.

A minha orientadora, Prof^ª. Dr^ª. Sandra C.P.F. Fabbri, pela confiança e constante ajuda na orientação, além da paciência e dedicação. A Prof^ª. Dr^ª Maria da Graça Brasil Rocha, pela colaboração com o trabalho desenvolvido.

Em especial, aos meus pais Zezo e Lena, que sempre apoiaram-me e incentivaram-me nessa caminhada. A minha irmã Luciana, que não poupou esforços para que esse trabalho fosse concluído, fazendo tudo aquilo que eu não podia fazer. Ao meu irmão Renato, pela motivação para a finalização desse trabalho. À minha namorada Aisla, pela força e paciência, além da compreensão nos inúmeros momentos de ausência que este trabalho exigiu-me. A ela e minha família, enfatizo aqui o agradecimento e o reconhecimento de que, sem sua ajuda e apoio, esse trabalho não estaria concluído.

Ao amigo e companheiro de mestrado Marcos, pela colaboração, ajuda e ricos momentos de discussão acerca desta dissertação. Pela motivação vinda do amigo Pedro. Pela ajuda do amigo e mestrando Bruno. Agradeço também à colaboração do amigo Mauricio Porto, que contribuiu no aspecto técnico da ferramenta desenvolvida.

Apesar de extensa a lista, destaco a ajuda dos amigos Abimael, Murilo, Rafael, Raphael e Lúcio, amigos do trabalho e do mestrado que colaboraram em tudo que fosse possível para a conclusão desta dissertação. Só Deus sabe o quanto me ajudaram...

Ao pessoal da Comunidade Hesed, pela amizade.

À comissão da PPG-CC e às secretárias Cristina e Miriam, pela ajuda e colaboração.

Enfim, a todos que fizeram e fazem parte da minha vida e que, de uma forma ou de outra ajudaram na construção deste trabalho.

EPIGRAFE

Para todos os seres humanos constitui quase um dever pensar que o que já se tiver realizado
é sempre pouco em comparação com o que resta por fazer.
João XXIII

SUMÁRIO

Lista de Figuras	iii
Lista de Tabelas	iv
Resumo	v
Abstract.....	vi
Introdução	1
1.1. Contexto	3
1.2 Motivação e Objetivos.....	5
1.3 Organização do Trabalho.....	6
Engenharia de Requisitos	8
2.1 - Considerações Iniciais.....	8
2.2 - O que é Requisito de Software.....	8
2.3 - Processo de Engenharia de Requisitos.....	10
2.4. Modelagem de Requisitos – Modelo de Casos de Uso	12
2.4.1 - Diagrama de Caso de Uso	13
2.4.2 - Especificação de Casos de Uso	16
2.5 - Considerações Finais.....	19
Gerenciamento de Requisitos – Aspectos de Rastreabilidade.....	20
3.1 – Considerações Iniciais	20
3.2 – Rastreabilidade	20
3.3 - Armazenamento de Requisitos.....	22
3.4 - Atividades de Gerenciamento de Requisitos.....	24
3.5 - Problemas da gerência de requisitos	32
3.6 - Ferramentas de gerenciamento de requisitos	33
3.7 - Considerações Finais.....	37
Trabalhos Relacionados.....	39
4.1 - Considerações Iniciais.....	39
4.2 – Apresentação dos Artigos.....	39
- Considerações Finais.....	68
O ambiente COCAR	71
5.1 – Considerações Iniciais	71
5.2 – Funcionalidades do ambiente COCAR	72
5.3 - Aspectos de Implementação do ambiente COCAR	75
5.3.1 - Processo de Desenvolvimento.....	76
5.3.2 - Projeto	79
5.3.3 - Implementação	83
5.4 - Requisitos Funcionais do ambiente COCAR – Gerenciamento de Requisitos e Aspectos de Rastreabilidade.....	83
5.4.1 – Geração da Matriz de Rastreabilidade de Requisitos	84
5.4.2 – Criação do Indicador de Estabilidade.....	85
5.4.3 – Rastreabilidade entre Documento de Requisitos e Modelo de Casos de Uso ..	86
5.4.4 – Consultas sobre os requisitos.....	87
5.5 - Considerações Finais.....	88
Exemplo de uso do ambiente COCAR.....	89
6.1 – Considerações Iniciais	89

6.2 - Apresentação do ambiente COCAR	90
6.3 – Criação de Sistema	92
6.4 - Cadastramento de Requisitos	93
6.5 – Criação do Modelo de Casos de Uso.....	97
6.5.1 - <i>Actor-Goal Reading Technique</i> (AGRT)	98
6.5.2 - <i>Use Case Reading Technique</i> (UCRT)	102
6.6 – Aspectos de Rastreabilidade.....	106
6.6.1 - Geração da Matriz de Rastreabilidade de Requisitos.....	106
6.6.2 – Criação dos Indicadores de Estabilidade	109
6.6.3 – Rastreabilidade entre Documento de Requisitos e Modelo de Casos de Uso	110
6.6.4 - Pesquisas sobre os Requisitos	112
6.7 - Considerações Finais.....	115
Conclusões.....	116
7.1 – Contribuições e limitações deste trabalho	118
7.2 – Lições aprendidas	119
7.3 – Trabalhos futuros.....	120
Referências Bibliográficas	122
Apêndice I.....	135
Apêndice II.....	142

Lista de Figuras

Figura 1 - Interação entre Atores e Casos de Uso	14
Figura 2 - Exemplo de associação com o estereótipo <<extend>>	15
Figura 3 - Exemplo de associação com o estereótipo <<include>>	16
Figura 4 - Exemplo de um Diagrama de Caso de Uso.....	18
Figura 5 - Especificação do Caso de Uso “Alugar Carro”	18
Figura 6 - Relação entre as boas práticas de Engenharia de Requisitos e o sucesso do produto [adaptado de Davis & Zowghi, 2005]	45
Figura 7 - Diagrama de Classe UML do Padrão DAO [Java, 2006]	80
Figura 8 - Diagrama de Classe UML do Padrão DAO [Java, 2006].....	81
Figura 9 - Diagrama de Classes UML do Padrão <i>Transfer Object</i> [Java, 2006].....	82
Figura 10 - Diagrama de Classes UML da funcionalidade Cadastrar Sistema	82
Figura 11 - Prototipação Evolucionária [Sommerville, 2003].....	83
Figura 12 - Interface de ajuda do sistema COCAR.....	91
Figura 13 - Escolha de um sistema ou a criação de um novo sistema.....	92
Figura 14 - Selecionar um sistema já cadastrado como ativo.....	92
Figura 15 - Cadastramento de um sistema no ambiente COCAR	93
Figura 16 - Cadastramento de Requisitos no ambiente COCAR	95
Figura 17 - Cadastramento de Entradas no ambiente COCAR	96
Figura 18 - Cadastramento de Atores (primário e secundário).....	96
Figura 19 - Cadastramento de Solicitante de Requisitos	97
Figura 20 - Cadastramento de Gerente de Requisito	97
Figura 21 - Processo de marcação de atores, funções e restrições	99
Figura 22 - Determinação dos atores e objetivos.....	100
Figura 23 - Exemplo de redundância intra-atores	101
Figura 24 - Exemplo do FAO gerado pela aplicação da AGRT no ambiente COCAR.....	102
Figura 25 - Primeira Etapa: Formulário de Casos de Uso Preliminares.....	103
Figura 26 - Interface para a especificação dos Casos de Uso e Janela com a janela dos requisitos relacionados e janela de especificação de curso alternativo	105
Figura 27 - Exemplo de Matriz de Rastreabilidade de Requisitos	107
Figura 28 - Exemplo de Matriz de Rastreabilidade de Requisitos	108
Figura 29 - Exemplo do Indicador de Estabilidade	110
Figura 30 - Exemplo de visualização da rastreabilidade entre requisito e Caso de Uso	111
Figura 31 - Exemplo de indicação de alterações de em requisitos	112
Figura 32 - Possibilidades de pesquisas sobre os requisitos.....	113
Figura 33 - Consulta sobre os requisitos de determinado Gerente.....	114
Figura 34 - Consulta sobre os requisitos de determinado Solicitante	114

Lista de Tabelas

Tabela 1 - Histórico sobre a situação de projetos de software [Standish Group,2004b; 1999].....	2
Tabela 2 – Principais fatores de sucesso de um projeto de software [Standish Group, 1994]	2
Tabela 3 - Exemplo de Matriz de Rastreabilidade de Requisitos [Sommerville, 2003]	27
Tabela 4 - Consenso de “boas” práticas de engenharia de requisitos [Davis & Zowghi, 2005].....	43
Tabela 5 – Comparação das pesquisas sobre rastreamento gerenciamento de requisitos [Toranzo et al., 2002]	54

Resumo

Este trabalho teve como objetivo a implementação de uma versão inicial de um ambiente de apoio ao desenvolvimento de software, baseado no Modelo de Casos de Uso, denominado COCAR. A concepção e as funcionalidades desse ambiente são frutos de alguns trabalhos de mestrado. No contexto deste trabalho o objetivo principal de estudo foi o gerenciamento de requisitos, embora, em termos de implementação, funcionalidades relacionadas a outros trabalhos anteriores foram também contempladas. Esse ambiente apóia algumas atividades do desenvolvimento de software, procurando aumentar a qualidade ao longo desse processo. Sabe-se que uma das principais características relacionadas à qualidade de um produto de software é que este atenda aos requisitos do usuário. Na Engenharia de Requisitos, a comunicação entre usuário e desenvolvedor é registrada no Documento de Requisitos, sendo que, a partir deste, os requisitos podem ser modelados, por exemplo, por meio de Modelos de Casos de Uso. Apesar dos esforços em levantarem-se as necessidades junto ao usuário, sabe-se que os requisitos solicitados serão alterados durante o processo de desenvolvimento. Um ponto fundamental para que a qualidade seja mantida é que essas alterações ocorram de forma controlada e previsível. A capacidade de descrever e acompanhar a vida de um requisito dentro do processo de desenvolvimento de software é chamada rastreabilidade de requisitos. Como o documento de requisitos representa o elo entre o usuário e os desenvolvedores, caso a rastreabilidade seja obtida a partir desse ponto, a qualidade do processo de desenvolvimento tende a ser melhor e a permitir um controle desde suas fases iniciais. A contribuição deste trabalho para a implementação do ambiente COCAR incluiu as funcionalidades de registro dos requisitos de um sistema, a geração do Modelo de Casos de Uso e o gerenciamento de requisitos, sobretudo no que diz respeito aos aspectos de rastreabilidade, entre o Documento de Requisitos e o Modelo de Casos de Uso, oferecendo métricas de rastreabilidade propostas na literatura e possibilidade de determinação do relacionamento entre os requisitos. Com base nas informações oferecidas pela ferramenta, pode-se perceber que elas são de grande valia para o planejamento e acompanhamento do desenvolvimento de um sistema, o que pode ter uma grande contribuição prática para a melhoria da qualidade do processo e do produto.

Abstract

The objective of this work was to implement an initial version of a development support environment, based on the Use Case Model, named COCAR. The conception and the features of this environment are the result of several master papers. In the scope of this work, the main subject under study has been requirement management, although concerning its implementation, functionalities related to other previous works have also been contemplated. This environment prop up a few software development tasks, aiming at leveling up quality throughout the process. It is well know that one of the principal characteristics related to software product quality is that the product should meet the user requirements. In requirement engineering, the communication between the user and the developer is stated in the Requirement Document and based on it, the requeriment can be modelled, for instance, by means of the Use Case Model. Despite the efforts invested in gathering requirements with the user, it is know that they are very likely to be ammended during the development process.

A fundamental factor for the maintainance of the overall quality is that such modifications to the requirements occur in a monitored and foreseeable way. The ability to describe and follow a requirement life-cycle within the software development process is denominated requirement traceability. As the requirement document represents what binds the user and the developers, if traceability is obtained from that point onwards, the quality of the development process tend to increase and this permits monitoring since its earliest phases. This paper contribution to the implementation of the environment COCAR, includes features such as: registering the requirement of a system, generating of a Use Case Model, requirement management (mainly what concerns traceability between the Requirement Document and the Use Case Model), providing traceability metrics found in academic literature and the possibility of determining the relationship between the requirements. Based on the data provided by the tool, it is easy to perceive that they are highly relevant to any system development planning or following up, which can be an important pragmatic contribution to the improvement of software development and software products.

Capítulo 1

Introdução

O uso de software no cotidiano das pessoas vem aumentando a cada dia e pode-se dizer que já faz parte de praticamente qualquer atividade humana. A tendência de utilização da informática nos mais diversos setores produtivos é muito grande, principalmente levando-se em consideração o advento de novas tecnologias como a computação móvel e as redes sem fio. Dessa forma, a penetração da informática está atingindo todos os públicos, gerando uma demanda por software excessivamente grande. Por outro lado, a indústria do software enfrenta desafios de gerar produtos que atinjam às expectativas dos clientes, obedecendo prazos, custos e critérios de qualidade.

Apesar desse desafio de satisfazer níveis cada vez mais rígidos de qualidade, uma vez que o software é usado em situações em que não pode falhar, como sistemas de aviação ou controle metroviário, a indústria de software ainda não consegue atender a essa expectativa do mercado, principalmente de forma mais generalizada, para outros tipos de software, contornando os desafios e exigências por qualidade solicitados no mercado.

Pesquisas do instituto Standish Group (2004b) mostram que no ano de 2004, a quantidade de projetos que foi finalizada com sucesso, ou seja, respeitando os prazos estabelecidos, o orçamento e, sobretudo, atendendo às expectativas dos clientes em todas as funcionalidades e características do software, é de apenas 29%. Apesar de ser um percentual baixo, observando-se o histórico publicado pelo mesmo instituto, vê-se que esse número, apesar de tímido – principalmente considerando-se o tamanho do mercado de software – apresenta uma melhora com relação aos anos anteriores. A Tabela 1, com dados do mesmo instituto, ilustra essa realidade.

Para a situação de o projeto ser classificado como Falha, é necessário que o mesmo tenha sido abortado antes de concluído ou que tenha sido entregue ao usuário, mas nunca utilizado por ele.

No caso do projeto ser definido como Problemático, é preciso que ele não tenha respeitado prazo, custo, e/ou que tenha sido entregue sem alguma característica solicitada pelo cliente.

Tabela 1 - Histórico sobre a situação de projetos de software [Standish Group, 2004b; 1999]

Ano	Situação dos projetos de softwares		
	Sucesso	Falha	Problemático
2004	29 %	18 %	53 %
1998	26 %	28 %	46 %
1996	27 %	40 %	33 %
1994	16 %	31 %	53 %

A pesquisa realizada pelo mesmo instituto em 1994 foi mais específica e buscava determinar quais fatores eram os mais importantes para definir o sucesso ou não de um projeto de software. Dessa forma, ela procurou determinar o motivo pelo qual um projeto extrapola custo, prazo ou é entregue aos clientes com alguma de suas funcionalidades inconsistentes. Os dados da pesquisa estão resumidos na Tabela 2.

Analisando-se a tabela, conclui-se que um fator crítico de sucesso para os projetos de software é a habilidade do processo em lidar com usuários e entender melhor as suas necessidades, além de estar apto e preparado para as mudanças nos requisitos do usuário. Os três primeiros fatores, que estão associados diretamente com Requisitos, somam, aproximadamente, 37%, o que representa uma porção significativa de motivos que podem levar ao insucesso de um produto de software. Embora essa última pesquisa, retratada na Tabela 2, tenha sido realizada no ano de 1994, acredita-se que o panorama não tenha alterado significativamente. Salem (2006) afirma que maioria dos erros dos software são frutos de erros na fase de levantamento de requisitos e durante o acompanhamento da evolução deles durante todo o processo de desenvolvimento de software.

Tabela 2 – Principais fatores de sucesso de um projeto de software [Standish Group, 1994]

Fatores que tornam um Projeto Crítico	% de Respostas
---------------------------------------	----------------

1. Falta de Especificação do Usuário	12.8%
2. Requisitos Incompletos	12.3%
3. Mudança nos Requisitos	11.8%
4. Falta de Apoio Executivo	7.5%
5. Tecnologia Imatura	7.0%
6. Falta de Recursos	6.4%
7. Expectativas previstas são irreais	5.9%
8. Objetivos obscuros	5.3%
9. Tempo irreal	4.3%
10. Tecnologia nova	3.7%
11. Outros	23.0%

Tal deficiência em lidar com requisitos, muitas vezes em um ambiente de desenvolvimento desordenado, que privilegia a qualidade de seus produtos à qualidade de seus processos, tem como conseqüência um software sem qualidade e que, possivelmente, aumentará as estatísticas dos projetos com problemas em prazo, custo e não atendimento das necessidades dos clientes.

1.1. Contexto

Dada a importância da fase de Engenharia de Requisitos para o ciclo de desenvolvimento de software, como foi mostrado nos dados apresentados anteriormente, alguns trabalhos relacionados a esse tema têm sido orientados na linha de pesquisa que este mestrado está inserido.

Um desses trabalhos foi o mestrado de Belgamo (2004), que propôs uma técnica de leitura para dar apoio à construção de Modelos de Casos de Uso (Diagrama e Especificação dos Casos de Uso), de tal forma que, à medida que esses modelos são construídos, faz-se uma revisão do

Documento de Requisitos. Essa técnica é denominada TUCCA¹ – *Technique for Use Case Construction and Construction-based Requirements Analysis*.

A partir dos resultados de alguns estudos experimentais já realizados e publicados por Belgamo e Fabbri (2005), pode-se dizer que a TUCCA contribui substancialmente para a construção de Modelos de Casos de Uso mais padronizados, de tal forma que a experiência e a subjetividade do projetista não tenham tanta interferência nessa construção. Os Modelos de Casos de Uso gerados nos estudos experimentais realizados por Belgamo e Fabbri foram também avaliados por uma técnica proposta por Anda e Sjoberg (2002) e, de acordo com essa técnica, os modelos satisfazem os requisitos de qualidade que um bom modelo deve apresentar [Belgamo et al., 2005].

Outro trabalho que acenou na mesma direção, com foco na Engenharia de Requisitos, foi o mestrado de Kawai (2005). Nesse trabalho, foram propostas diretrizes para a descrição de requisitos funcionais com base em um *template* que organiza as informações relacionadas aos requisitos em campos distintos. Esse formato de especificação facilita a aplicação da técnica TUCCA.

Assim, considerando os seguintes pontos: i) a relevância da fase de engenharia de requisitos no contexto do desenvolvimento de software; ii) a técnica de modelagem de requisitos por meio de casos de uso é bastante utilizada na prática e é independente de paradigma de desenvolvimento; iii) a técnica TUCCA mostrou ser bastante efetiva na elaboração de modelos de casos de uso e na detecção de defeitos no documento de requisitos; e iv) vários passos da técnica TUCCA são algorítmicos e podiam ser automatizados; decidiu-se, no grupo de pesquisa, desenvolver um ambiente que automatizasse a técnica. Esse ambiente, denominado COCAR, teria como funcionalidade básica o suporte à geração do modelo de casos de uso, por meio da TUCCA, pois, a partir do momento em que se tenha esse modelo, várias outras funcionalidades poderiam ser implementadas, de forma a melhorar a qualidade do desenvolvimento de software. Exemplos dessas funcionalidades são: rastreabilidade entre requisitos e casos de uso, geração de pontos de casos de uso, geração de casos de teste com base em casos de uso, etc.

¹ Essa técnica era referenciada anteriormente por GUCCRA - Guidelines for Use Case Construction and Requirements Analysis

Particularmente, em relação à rastreabilidade de requisitos, Salem (2006) afirma que essa atividade ajuda a garantir que o projeto das especificações seja verificado apropriadamente e os requisitos funcionais também sejam validados. Segundo o autor, as vantagens da rastreabilidade são ainda mais evidentes quando o sistema passa por mudanças e o uso de técnicas formais - como a TUCCA e as diretrizes propostas por Kawai (2005) – favorece o estabelecimento da rastreabilidade. Munson e Nguyen (2005) afirmam que as práticas de rastreabilidade irão melhorar apenas quando forem apoiadas por ferramentas que reduzam o esforço requerido para mantê-las. Considerando esse cenário, o ambiente COCAR torna-se um facilitador para alguns aspectos de rastreabilidade possam ser implementados, facilitando o processo de gerenciamento de requisitos.

Além dos aspectos de rastreabilidade, paralelamente, mais dois trabalhos de mestrado estão sendo orientados e farão parte do ambiente COCAR, sendo que um deles tem por objetivo a geração dos Pontos de Casos de Uso e o outro, a determinação de casos de teste baseados nos Casos de Uso gerados com o apoio da TUCCA. Pretende-se, dessa forma, que o ambiente seja capaz de agregar o maior número de etapas do processo de desenvolvimento de software, tendo como objetivo facilitar e melhorar a qualidade das atividades que devem ser realizadas. Para tanto, um dos principais pontos desse ambiente é que ele seja extensível, permitindo que novos módulos sejam incorporados no futuro.

1.2 Motivação e Objetivos

Considerando a efetividade da TUCCA no que diz respeito tanto à padronização na construção de modelos de casos de uso como no que diz respeito à ajuda na detecção de defeitos no documento de requisitos à medida que o modelo é construído, decidiu-se implementar um ambiente que desse suporte a várias atividades do desenvolvimento de software.

Com base no contexto exposto anteriormente, a principal motivação deste trabalho é contribuir para o desenvolvimento do ambiente COCAR, cujo objetivo é dar suporte a várias atividades do desenvolvimento de software, com base nos casos de uso que representam os requisitos de um determinado sistema. O desenvolvimento desse ambiente tem sido um esforço conjunto desse

grupo de pesquisa e do trabalho de alguns mestrados e iniciações científicas. Embora a concepção desse ambiente tenha surgido na época do trabalho de Belgamo (2004), sua implementação teve início a partir deste e de outro mestrado realizado simultaneamente a este [Martins, 2007].

Considerando-se as diversas atividades que podem ser realizadas a partir do momento em que se tenha o modelo de casos de uso de um sistema, o objetivo principal deste trabalho é dar suporte ao gerenciamento de requisitos, tendo como alvo principal os casos de uso e o documento de requisitos. A agilidade em receber solicitações de novos requisitos ou alterar os requisitos existentes, visa ao aumento da qualidade do processo de desenvolvimento de software, à medida que busca a mudança de forma racional, analisando o impacto da mesma, dando suporte para a tomada de decisão dos gerentes de projeto e armazenando o histórico das alterações.

Dessa forma, o objetivo deste trabalho é dar início ao desenvolvimento do ambiente COCAR, compartilhando a implementação da TUCCA [Belgamo, 2004] e das diretrizes de especificação de requisitos propostas por Kawai (2005) com outro aluno de mestrado [Martins, 2007] e implementar, individualmente, os aspectos de gerenciamento de requisitos. Essa implementação deve prezar pela característica extensível do ambiente, conferindo ao mesmo a capacidade de permitir a rastreabilidade de outros artefatos que venham a ser gerados.

1.3 Organização do Trabalho

O trabalho está organizado em seis capítulos, sendo que este capítulo apresentou o contexto no qual a proposta deste trabalho está inserida, assim como os objetivos e a motivação para o desenvolvimento da mesma.

O Capítulo 2 apresenta uma breve revisão bibliográfica abordando os conceitos sobre Engenharia de Requisitos e Modelo de Casos de Uso.

No Capítulo 3 são detalhados os aspectos sobre o gerenciamento de requisitos, focando, sobretudo, a rastreabilidade e as ferramentas que dão suporte a esse processo.

No Capítulo 4 estão documentados os trabalhos relacionados ao escopo deste mestrado.

O Capítulo 5 contém a descrição do ambiente COCAR, mostrando sua estrutura, aspectos de implementação e requisitos funcionais.

No Capítulo 6 é apresentado um estudo de caso, mostrando o uso da ferramenta e seus recursos, dando-se ênfase aos aspectos de rastreabilidade de requisitos, que é o foco principal deste estudo.

O Capítulo 7 apresenta a conclusão deste trabalho, dando ênfase às vantagens, desvantagens e limitações do mesmo.

No Apêndice I são apresentados exemplos da especificação de requisitos do ambiente COCAR. No Apêndice II também são exemplificados casos de uso gerados pela aplicação da TUCCA nos requisitos do ambiente.

Capítulo 2

Engenharia de Requisitos

2.1 - Considerações Iniciais

Como o objetivo principal desse trabalho foi a criação de uma ferramenta para a rastreabilidade de requisitos baseada em modelo de casos de uso, nesse capítulo são apresentados os principais conceitos envolvidos com Requisitos de Software, Engenharia de Requisitos e Modelo de Casos de Uso.

Inicialmente é apresentada a definição de Requisito de Software na Seção 2.2. Na Seção 2.3 é discutido o Processo de Engenharia de Requisitos. A Modelagem de Casos de Uso é apresentada na Seção 2.4. Por fim, a Seção 2.5 contém a apresentação das considerações finais.

2.2 - O que é Requisito de Software

Requisito de software compreende tudo aquilo que o software deve fazer. Segundo Abbott, requisito pode ser uma função, uma restrição ou outra característica que deve ser fornecida ou atendida para satisfazer as necessidades do usuário do sistema que será desenvolvido [Abbott,1986]. Segundo o IEEE (1990), requisito pode ser:

- 1) uma condição ou capacidade que um usuário necessita para solucionar um problema ou atingir um objetivo;
- 2) uma condição que um sistema ou componente do sistema precisa atender para satisfazer um contrato, padrão, especificação ou outro documento formalmente estabelecido;
- 3) uma representação documentada de uma condição ou capacidade, como em 1) e 2).

Todo o desenvolvimento de software está baseado no conjunto dos requisitos coletados e as pessoas que, direta ou indiretamente, são afetadas pelo sistema a ser construído para a solução de algum problema são chamadas *stakeholders* [Ryan, 1998], isto é, são os “envolvidos no projeto” e consistem, entre outros, de: cliente, usuários, desenvolvedores e líder do projeto.

Existem diversas maneiras de classificar os requisitos. Em seguida, mostra-se a classificação proposta por Sommerville (2003):

- **Requisitos Funcionais:** são os requisitos que estão ligados diretamente à funcionalidade do software. Eles definem como o sistema deve reagir às entradas específicas e como deve comportar-se em determinadas situações. Podem até declarar explicitamente o que o software não deve fazer. O autor destaca ainda a necessidade da especificação dos requisitos funcionais do sistema estarem completos (todas as funções solicitadas pelo usuário devem estar definidas) e consistentes (eles não devem ser contraditórios).
- **Requisitos Não Funcionais:** refletem os requisitos que expressam as qualidades específicas que o software deve ter. Requisitos não funcionais são aqueles que não dizem respeito diretamente às funções específicas fornecidas pelo sistema [Sommerville, 2003]. Enquanto cada requisito funcional expressa o que uma parte do sistema deve fornecer, o requisito não funcional pode estar associado a todo o sistema. É interessante notar que em alguns casos, se um desses requisitos não funcionais não for atendido, todo o software ficará comprometido, podendo tornar-se inútil. Por exemplo, caso um software financeiro não atenda o critério de confiabilidade, ninguém irá utilizá-lo, tornando-o inútil. Os requisitos não funcionais também podem dizer respeito a hardware, a aspectos legais e às organizações que desenvolvem o sistema. Sommerville (2003) propõe ainda a seguinte classificação dos requisitos não funcionais:
 - Organizacionais: vêm de políticas e procedimentos das organizações do cliente e do desenvolvedor tais como os padrões de processo usados para o desenvolvimento ou a linguagem de programação.
 - de Produto: especificam o comportamento do produto e as características do funcionamento do mesmo. Podem, por exemplo, fazer referência à critérios de

confiabilidade, como tempo de resposta a uma determinada solicitação ou a taxa de falhas aceita.

- Externos: abrangem todos os requisitos oriundos de fatores externos ao sistema, como por exemplo, a conformidade dos requisitos levantados com as leis em vigor no país.
- **Requisitos de Domínio:** são requisitos que se originam do domínio da aplicação e podem mostrar as características desse domínio. Eles podem ser tanto requisitos funcionais como não funcionais. Quando se trata de um software que calcula impostos ou taxas a serem cobradas, por exemplo, as regras que determinam como essa cobrança deve ser feita estão atreladas ao domínio da aplicação.

2.3 - Processo de Engenharia de Requisitos

O processo de descobrir, analisar, documentar e verificar essas funções e restrições é chamado Engenharia de Requisitos [Sommerville, 2003]. A Engenharia de Requisitos pode ser dividida em quatro fases: Levantamento dos Requisitos, Especificação dos Requisitos, Validação dos Requisitos e Gerenciamento de Requisitos. Essas quatro etapas serão explicadas em seguida:

- **Levantamento dos Requisitos:** é a etapa em que os requisitos são extraídos dos clientes e usuários. Nessa fase, busca-se determinar “o quê” o sistema deve fazer. São buscadas todas as informações do domínio da aplicação, ou seja, tudo aquilo que o sistema deve fornecer, bem como o desempenho exigido para as funcionalidades, as restrições do sistema e as qualificações de hardware. Trata-se de uma tarefa de grande dificuldade uma vez que envolve diversos aspectos como relações humanas e o contexto social e econômico do ambiente em que o software será implantado. Algumas dessas dificuldades listadas por Sommerville (2003) são:
 - Com frequência os *stakeholders* desconhecem em detalhes que desejam do sistema computacional. Apesar de terem uma idéia do sistema em termos muito gerais, quando questionados sobre detalhes, acham difícil definir com maior precisão o que desejam do sistema.

- Os *stakeholders* expressam naturalmente os requisitos em seus próprios termos, com conhecimentos implícitos de sua área de atuação.
 - Diferentes *stakeholders* têm em mente os mesmos requisitos e podem expressá-los de maneiras distintas, gerando conflitos.
 - Fatores políticos podem influenciar os requisitos do sistema.
 - O ambiente econômico e de negócios no qual o requisito ocorre é dinâmico. Inevitavelmente, ele se modifica durante o processo de análise. Como consequência, a importância dos requisitos específicos pode mudar. Novos requisitos podem surgir por parte dos novos *stakeholders*, que não haviam sido consultados no início.
- **Especificação de Requisitos:** é a etapa em que todos os requisitos levantados na fase anterior devem ser documentados no Documento de Requisitos, para uma posterior validação. Esse documento é a declaração oficial do que o sistema deve fazer. Existem modelos de Documento de Requisitos como, por exemplo, o padrão IEEE Std 830-1998 [IEEE, 1998]. Power e Moyniahn (2003) fizeram um estudo abordando a existência de grande variedade de padrões de documentos de requisitos para atender às mais diversas situações e sistemas. O estudo analisou vários tipos de sistemas em diferentes áreas (Financeira, Telecomunicações, Aeroespacial, Industrial e Governamental) e constatou que um único modelo de Documento de Requisitos não é capaz de atender as especificidades de cada uma das áreas. Os autores também definem o Documento de Requisitos como o contrato e o acordo entre cliente e aqueles que irão desenvolver o sistema, dando ênfase também à assinatura desse Documento de Requisitos pelas partes envolvidas, com o intuito de garantir que as expectativas de ambos sejam atendidas.
 - **Validação dos Requisitos:** é a etapa em que se busca eliminar as possíveis inconsistências do Documento de Requisitos. A validação visa à geração de um Documento de Requisitos em que estes estejam descritos da forma apropriada e completa, buscando eliminar problemas de ambigüidade e inconsistência. A preocupação maior desta fase é com a qualidade do Documento de Requisitos produzido. Essa etapa é importante porque a ocorrência de erros nesse tipo de documento ocasiona grandes custos em todo decorrer do desenvolvimento do sistema, principalmente com relação ao retrabalho. O processo de validação de requisitos deve gerar um relatório com os erros e

as inconsistências do Documento de Requisitos produzido na especificação de requisitos. Esse, por sua vez, deve ser refeito para posterior validação, constituindo-se assim, em um processo cíclico que tem o fim determinado pela aprovação do Documento de Requisitos após a sua validação. Muitos custos extras podem ser evitados com um processo de validação de requisitos bem feito. O objetivo final dessa fase é liberar o Documento de Requisitos que deve representar de forma clara e consistente o que deve ser implementado. Tal documento será o guia para etapas seguintes do processo de desenvolvimento do sistema.

- **Gerenciamento de Requisitos:** é a etapa em que se busca controlar a evolução dos requisitos do sistema, seja por constatação de novas necessidades ou pela necessidade de corrigir problemas nos requisitos registrados até o momento. Esse tópico será abordado em detalhes no Capítulo 3 deste trabalho, uma vez que os aspectos de rastreabilidade dos requisitos fazem parte dessa atividade de Gerenciamento de Requisitos.

2.4. Modelagem de Requisitos – Modelo de Casos de Uso

Nessa seção, apresentam-se os conceitos sobre Casos de Uso, uma vez que o trabalho aqui proposto tem por objetivo especificar e implementar uma ferramenta que apóie a rastreabilidade dos requisitos com base em informações representadas nesses modelos.

Os Casos de Uso são representações do Documento de Requisitos que têm como objetivo mostrar a funcionalidade do sistema, bem como a interação dos usuários com o mesmo. Possuem ainda a característica de facilidade de entendimento por parte dos usuários, contribuindo assim, no auxílio do levantamento e validação dos requisitos junto aos mesmos.

A técnica de Casos de Uso foi proposta em 1992 por Ivar Jacobson, em seu modelo de processo denominado Objectory [Jacobson et al., 1992]. Com o passar do tempo, os Casos de Uso foram incorporados pela UML e o modelo Objectory foi comprado pela Rational Software. O Objectory, juntamente com os Casos de Uso tornaram-se parte do Rational Unified Process (RUP) [Furlan, 1998; Schneider & Winters, 2001]. A notação UML possui uma representação diagramática de um Caso de Uso chamada Diagrama de Casos de Uso [Pressman, 2006]. Os

diagramas de Casos de Uso mostrados neste trabalho são baseados na notação UML mantida pelo OMG – *Object Management Group* [UML, 2003].

De acordo com Jacobson et al. (1992), a técnica de Casos de Uso funciona como um modelo central que é utilizado para todos os demais modelos das próximas fases do processo de desenvolvimento, ou seja, os Casos de Uso vão ser utilizados nas fases de Análise, Projeto, Implementação, Testes e Manutenção, de acordo com as necessidades correspondentes a elas. Além dessas características, os Casos de Uso podem ser construídos independentemente da abordagem de desenvolvimento adotada, seja ela orientada a objetos ou não.

No contexto desse trabalho, considera-se que um Modelo de Casos de Uso é composto pelo Diagrama de Casos de Uso (representação pictórica) e pelas especificações de Casos de Uso que contêm as descrições das interações do sistema com o seu lado externo. Esses dois itens são detalhados na seqüência.

2.4.1 - Diagrama de Caso de Uso

De acordo com Jacobson et al. (1992), para a elaboração do Diagrama de Caso de Uso, os seguintes itens devem ser definidos:

- **Fronteira do Sistema:** esse item do modelo é utilizado para estabelecer a fronteira na qual a funcionalidade do sistema está inserida. O objetivo é definir as partes do sistema que serão criadas dentro dos limites de prazo e custo predefinidos na fase de planejamento. A definição de atores e casos de uso do sistema auxilia na identificação de sua fronteira. A fronteira do sistema é algo abstrato e não é representado no diagrama; porém, como já explicado, é útil na definição da funcionalidade pretendida pelo sistema e pelos atores.
- **Atores:** este item representa os usuários do sistema. Os atores podem representar usuários humanos ou outros sistemas que interajam com o sistema a ser desenvolvido. A fronteira do sistema auxilia na identificação dos atores uma vez que qualquer elemento externo ao sistema é considerado um ator.

Os atores definem os papéis que os usuários do sistema podem assumir. Jacobson et al. (1992) classificam os atores da seguinte forma:

- **Atores Primários:** são aqueles que fazem uso do sistema diretamente no seu dia-a-dia. Esse tipo de ator realiza uma ou mais das principais tarefas do sistema.
- **Atores Secundários:** são aqueles que supervisionam e mantêm o sistema. Dessa forma, só existem atores secundários caso exista pelo menos um ator primário.
- **Casos de Uso:** esse item representa uma parte do comportamento do sistema. Após a definição do que está do lado de fora do sistema, deve-se iniciar a definição dos Casos de Uso. No Diagrama de Casos de Uso, um Caso de Uso é representado por uma elipse com o nome do Caso de Uso identificado logo abaixo da mesma, conforme pode ser observado na Figura 1.

A Figura 1 mostra o Caso de Uso “Alugar Carros” e a interação do Ator com o Caso de Uso (ou vice-versa) é representada por setas que podem conter informações. Essas informações ajudam no entendimento de quais são as entradas para o Caso de Uso, bem como suas saídas. Os Casos de Uso podem possuir associações com outros Casos de Uso. Os dois tipos de associações utilizadas são: <<extend>> e <<include>>.

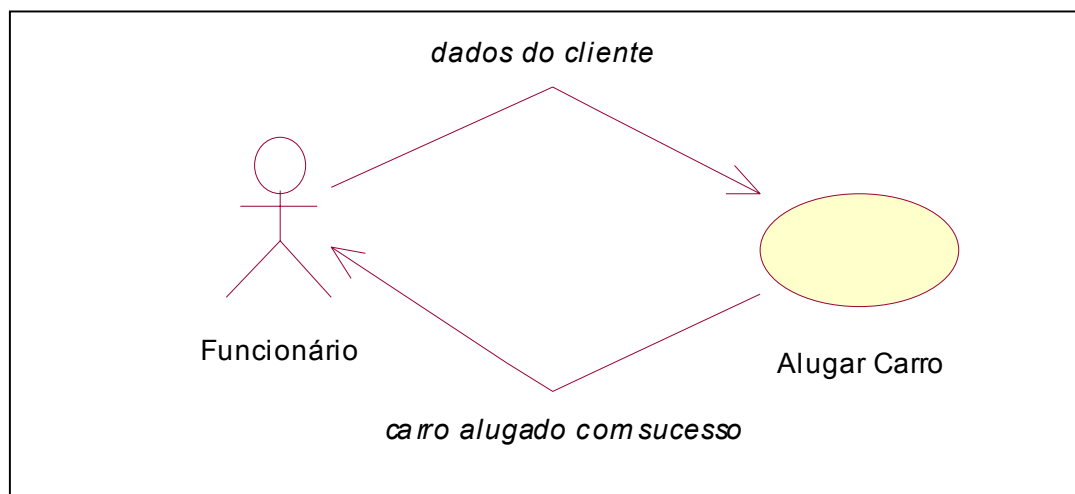


Figura 1- Interação entre Atores e Casos de Uso

Usa-se o estereótipo <<extend>> para estender, eventualmente, o comportamento de um Caso de Uso existente. Dessa forma, adiciona-se comportamento ao Caso de Uso sem mudar o Caso de Uso original [Schneider & Winters, 2001]. O uso do estereótipo <<extend>> é mostrado na Figura 2.

Usa-se o estereótipo <<include>> para evitar a duplicação de passos em vários Casos de Uso. Esse tipo de estereótipo é utilizado como uma estratégia de reuso para texto de Caso de Uso. Ao invés de colocar os passos em vários documentos de especificação de Caso de Uso, criam-se Casos de Uso que possuam associação com o estereótipo <<include>> [Kulak & Guiney, 2000]. Este tipo de associação aumenta a consistência entre os Casos de Uso, pois evita que a mesma funcionalidade seja especificada em dois locais diferentes, podendo torná-las inconsistentes.

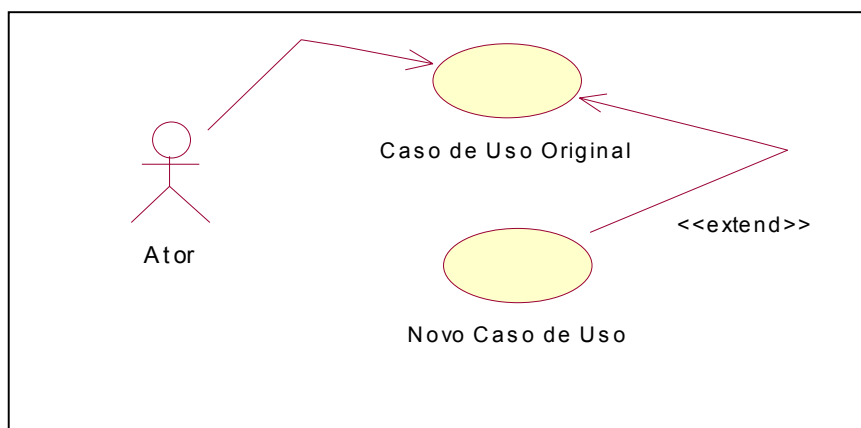


Figura 2 -Exemplo de associação com o estereótipo <<extend>>

A Figura 3 mostra a associação com o estereótipo <<include>> entre Casos de Uso. O “Caso de Uso B” foi identificado como um Caso de Uso com comportamento comum e o “Caso de Uso A” utiliza esse comportamento.

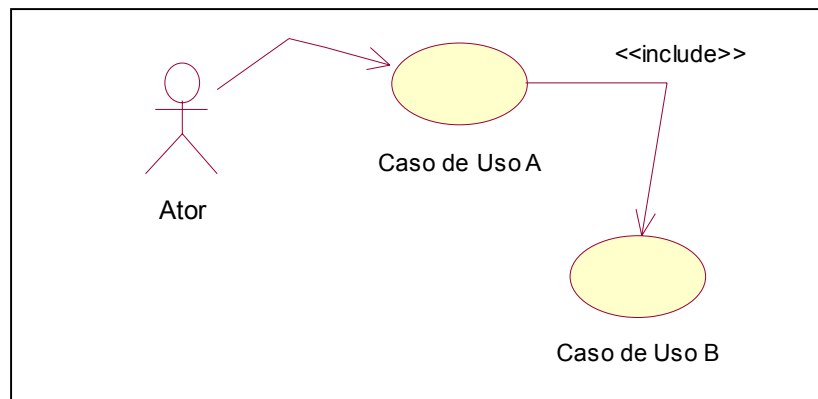


Figura 3 - Exemplo de associação com o estereótipo <<include>>

Vale ressaltar que quando é usado o estereótipo <<extend>> (Figura 2) e <<include>> (Figura 3) os sentidos das setas de associações são diferentes. No estereótipo <<extend>>, a seta da associação “chega” no Caso de Uso estendido, enquanto que no estereótipo <<include>>, a seta da associação “chega” no Caso de Uso que possui comportamento comum.

2.4.2 - Especificação de Casos de Uso

Uma vez identificados os principais Casos de Uso do sistema e tendo desenvolvido o Diagrama de Casos de Uso, o próximo passo é a documentação dos mesmos. Essa documentação dos Casos de Uso, também chamada de Especificação, é usada para descrever o que o Caso de Uso faz, podendo utilizar-se tanto uma linguagem natural como uma linguagem formal.

Em geral, a especificação de Casos de Uso é feita com linguagem natural. Embora essa especificação possa ser elaborada sem formatação, existem alguns modelos propostos na literatura que podem ser utilizados para a Especificação de Casos de Uso [Kulak & Guiney, 2000; Ryser & Glinz, 2000; Schneider & Winters, 2001]. Baseado nesses autores, Belgamo (2004) propôs a estrutura de especificação apresentada na seqüência:

- **Nome do Caso de Uso:** composto por um nome único que identifique o Caso de Uso.
- **Descrição ou Resumo:** composto por uma declaração do propósito e dos objetivos a serem atingidos pelo Caso de Uso. É importante salientar que não se deve descrever nesse item o que acontece no curso normal de eventos.

- **Atores:** composto pelo nome dos papéis dos atores que interagem com o Caso de Uso, podendo ainda serem especificadas as responsabilidades de cada ator.
- **Pré-Condição:** composto pelas condições necessárias para que o Caso de Uso possa ser realizado.
- **Pós-Condição:** composto pela declaração de qual estado o sistema vai estar após a realização do Caso de Uso. O modelo definido por Ryser e Glinz (1999) sugere a divisão da pós-condição no caso de haver um fluxo de eventos encerrando-se com sucesso e um outro fluxo encerrando-se devido a alguma falha.
- **Curso Normal:** composto de uma série de passos que representam o caminho correto do Caso de Uso. Também é conhecido como curso básico. Os passos do Caso de Uso devem ser numerados e a descrição desses passos não deve ser extensa. É importante ressaltar que deve existir somente um curso normal para cada Caso de Uso.
- **Curso Alternativo:** composto também de uma série de passos, porém mostra os caminhos menos comuns que podem ocorrer. Esse tipo de curso corresponde a uma seqüência diferente de passos que foi usada para o curso normal.
- **Caminhos de Exceção:** composto das interações que ocorrem quando existe um erro.
- **Evento Disparador:** composto do critério de entrada para o Caso de Uso que está sendo especificado. Um evento disparador descreve uma necessidade de negócio, podendo estar relacionado ao tempo ou à finalização de outro Caso de Uso.
- **Requisitos Não Funcionais:** referem-se aos requisitos não funcionais do sistema. Podem ser, entre outros, restrições de tempo, requisitos de interface ou segurança.
- **Autor:** pessoa que criou a especificação, sendo também a responsável pela sua manutenção.
- **Data:** data de criação da especificação.
- **Versão:** corresponde a um código associado à versão da especificação, para qual pode ser utilizado com propósitos de rastreabilidade.

A partir dessa especificação foi construído o Ambiente COCAR, baseada em Modelos de Casos de Uso e que será detalhado no Capítulo 5.

A Figura 5 exemplifica a descrição do Caso de Uso “Alugar Carro” mostrado na Figura 4.

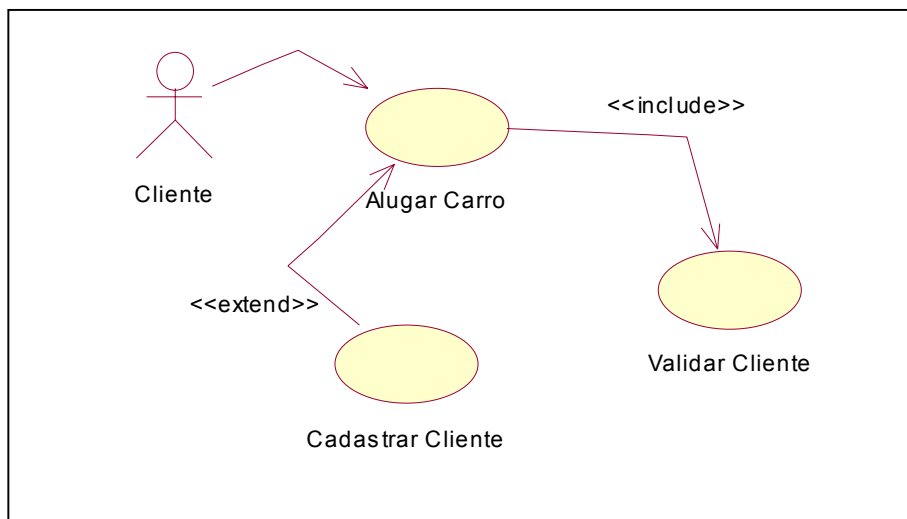


Figura 4 – Exemplo de um Diagrama de Caso de Uso

Nome do Caso de Uso	Alugar Carro
Descrição ou Resumo	O Caso de Uso tem como funcionalidade alugar determinados carros a clientes por um determinado período de tempo.
Atores	Cliente
Pré-Condição	Carro escolhido disponível para o aluguel.
Pós-Condição	Aluguel efetuado
Curso Normal	<ol style="list-style-type: none"> 1. Cliente seleciona o carro desejado. 2. Cliente fornece seu CPF. 3. Executar o Caso de Uso “Validar Cliente”. 4. Sistema efetua aluguel do carro.
Curso Alternativo	4. Sistema não efetua o aluguel ao cliente. Cliente tem débito com a empresa.
Caminhos de Exceção	Ainda não definido
Evento Disparador	Cliente deseja alugar um carro.
Requisitos Não Funcionais	Ainda não definido
Autor	André Di Thommazo
Data	15/03/2005
Versão	3

Figura 5 - Especificação do Caso de Uso “Alugar Carro”

Para especificar a interação entre um Caso de Uso e um ator, deve-se utilizar o curso normal de eventos contido no documento de especificação de requisitos. Conforme explicado, o curso normal é marcado pelos eventos que ocorrem de forma rotineira e natural no sistema. É importante ressaltar que os cursos normais são dependentes do ambiente no qual o sistema será utilizado. Sendo assim, uma outra locadora de carros poderia estipular que no curso normal do Caso de Uso “Alugar Carro” seria necessário realizar o cadastro do cliente, pois na maioria dos aluguéis, o cliente ainda não está cadastrado no sistema.

Quando é inserido um estereótipo <<include>> no Diagrama de Caso de Uso, deve-se observar o curso normal da especificação do Caso de Uso em questão. Na especificação, o Caso de Uso original executa até o ponto em que o novo Caso de Uso é inserido. Nesse momento, o novo Caso de Uso é executado e quando esse finaliza, é retornado ao curso normal do Caso de Uso original, seguindo os passos subseqüentes. Na Figura 5 observa-se essa situação, quando o sistema encontra-se no passo 3 do Caso de Uso “Alugar Carro”. É exatamente nesse passo que o Caso de Uso “Alugar Carro” dá lugar à execução do Caso de Uso “Validar Cliente”. Quando o Caso de Uso “Validar Cliente” completa sua execução, o sistema efetua o passo 4 do Caso de Uso “Alugar Carro”. Nesse ponto, é finalizada a execução do Caso de Uso “Alugar Carro”.

2.5 - Considerações Finais

Nesse capítulo foram apresentados os conceitos relacionados com Requisitos de Software, Engenharia de Requisitos e Modelagem de Casos de Uso. Com relação ao Modelo de Casos, foram mostrados a Especificação de Casos de Uso e o Diagrama de Casos de Uso. O conteúdo da Engenharia de Requisitos foi apresentado de forma superficial e será refinado no próximo capítulo, no qual será explorada a etapa de gerenciamento de requisitos, sobretudo os aspectos de rastreabilidade.

Capítulo 3

Gerenciamento de Requisitos – Aspectos de Rastreabilidade

3.1 – Considerações Iniciais

No capítulo anterior foram apresentadas, além da definição de Requisito de Software e do Modelo de Casos de Uso, as fases da Engenharia de Requisitos (Levantamento de Requisitos, Especificação de Requisitos, Validação de Requisitos e Gerenciamento de Requisitos). O foco principal deste trabalho recai justamente sobre a última fase, o Gerenciamento de Requisitos. É nessa fase que se busca acompanhar e controlar todas as alterações que possam ocorrer no conjunto de requisitos. É justamente desse assunto que este capítulo irá tratar, dando especial atenção para os aspectos de rastreabilidade.

O capítulo está organizado da seguinte maneira: na Seção 3.2 comenta-se a Rastreabilidade; na Seção 3.3 é abordado o Armazenamento de Requisitos e, na 3.4, as Atividades de Gerenciamento de Requisitos. Os problemas da Gerência de Requisitos são descritos na Seção 3.5 e as Ferramentas de Gerenciamento de Requisitos na Seção 3.6. Encerrando o capítulo, na Seção 3.7 são feitas as considerações finais.

3.2 – Rastreabilidade

Uma vez que os requisitos tenham sido levantados, documentados e validados, faz-se necessário o gerenciamento desses requisitos. Segundo Zisman e Spanoudakis (2004), as definições sobre Gerenciamento de Requisitos são divergentes, mas têm como objetivo: organizar e armazenar os requisitos e gerenciar mudanças dos requisitos.

Sempre que os requisitos levantados com os clientes forem alterados, os planos de software, os artefatos e as atividades afetadas devem sofrer ajustes para continuarem consistentes. O fato é que os requisitos são ativos e estão em uso durante todo o ciclo de vida, sendo a base para a modelagem do sistema.

Durante todo o processo de desenvolvimento de software os requisitos sofrerão alterações. Dessa forma, a agilidade no processo de tratamento dessas mudanças – que certamente acontecerão – é fundamental. Um fator crítico de sucesso para essa agilidade é a possibilidade de rastreamento dos requisitos.

O rastreamento de requisitos refere-se à habilidade de descrever e acompanhar a vida de um requisito [Hammer, 1997]. Esse controle deve abranger toda a sua existência, desde a fonte de origem - quando esse foi levantado, especificado e validado - passando pela fase de projeto, implementação e terminando na fase de testes do produto, sendo portanto, parte do processo de desenvolvimento do software. É uma técnica que permite a visibilidade do relacionamento de dependência entre os requisitos, identificando relacionamentos hierárquicos entre os mesmos. Além dessa dependência, deve ser possível identificar a dependência entre os requisitos e os artefatos que eles originaram e se relacionam.

Castor et al. (2004) classificam a rastreabilidade como um fator significativo na eficiência do gerenciamento de projeto de software e da qualidade do software desenvolvido, sendo que a falha na rastreabilidade despende tempo e gera altos custos nas correções e adaptações no software.

Segundo Pinheiro (1996), a motivação para o rastreamento de requisitos tem diversas justificativas, entre elas:

- durante a vida de um software, os requisitos irão evoluir;
- requisitos, em geral, estão dentro de um contexto e dependem da situação particular em que eles surgiram, sendo importante armazenar, por exemplo, os *stakeholders* que os propuseram, para eventuais esclarecimentos sobre alterações nos mesmos;
- rastreamento de artefatos do projeto é útil em todo o processo de desenvolvimento de software.

Existem várias razões que justificam a mudança, mas a principal é que durante o desenvolvimento do software, cliente e usuários passam a entender melhor suas necessidades e propõem alterações no sistema inicialmente solicitado. Algumas dessas necessidades são percebidas somente após parte do sistema ter sido implantado e utilizado. É muito comum o cliente receber a primeira versão do produto e questionar aqueles que o desenvolveram sobre a possibilidade de inserir novas funcionalidades ou de alterar as existentes. Essa situação ocorre em praticamente todos os software desenvolvidos e geram problemas para os gerentes de projetos, que sempre se questionam sobre o impacto dessas alterações propostas.

Congelar os requisitos após a etapa de validação é utopia [Rocha, 2001], uma vez que as regras de negócio e a dinâmica das organizações não são estáveis. Assim como os requisitos adaptam-se às mudanças, os sistemas também têm que se adaptarem. Essa capacidade de adaptação é mérito, em grande parte, do processo de gerenciamento de requisitos. A capacidade de adaptação do processo de desenvolvimento às alterações nos requisitos é, atualmente, um diferencial estratégico entre as empresas de software. As empresas que detêm essa vantagem competitiva podem oferecer produtos com melhor qualidade, e ainda prever, com mais precisão, o impacto de uma mudança no software proposta pelo usuário, sendo possível avaliar custos e prazos de forma mais eficiente.

3.3 - Armazenamento de Requisitos

Um aspecto muito importante relativo à rastreabilidade de requisitos é a necessidade de um repositório para armazená-los. Com o uso de um depósito de dados, torna-se possível guardar diversos níveis e tipos de informações sobre os requisitos, tais como:

- dependência entre os diferentes requisitos, com o intuito de estimarem-se os efeitos de uma proposta de alteração;
- *stakeholder* que o propôs, para ser consultado sobre eventuais alterações;
- prioridade do requisito para auxiliar no momento da elaboração do cronograma do projeto, de forma a executar aquelas tarefas que têm maior prioridade junto ao cliente;
- número de versão para o requisito, que permite armazenar todo o seu histórico.

Cordeiro (2002) definiu algumas características dos requisitos que devem ser armazenadas em um repositório. São elas:

- **Funções:** expressam explicitamente o que o requisito deve fazer. Pode ser composto de uma sentença, como por exemplo: o usuário pode consultar seu extrato bancário, fornecendo identificação e senha.
- **Atributos:** são parâmetros associados ao requisito, como por exemplo, quem é o *stakeholder* que o solicitou; qual o *status*, ou seja, se o requisito já foi modelado, implementado ou testado; a complexidade; o custo. Esses atributos serão mais explorados na Seção 3.6, sobre ferramentas de gerenciamento de requisitos.
- **Restrições:** são limitações que o requisito deve atender para satisfazer o cliente. Para o exemplo em que o usuário solicitava a consulta do extrato bancário, fornecendo a identificação e a senha, podemos definir como uma restrição que ambos devem ser criptografados. Nesses casos, a restrição de um requisito poderá dar origem a novos requisitos ou atributos comuns de dois requisitos que poderão ser usados para mapear o relacionamento entre eles.

Dentre os benefícios do uso de um repositório, Zisman e Spanoudakis (2004) citam:

- o repositório pode ser utilizado nas várias fases do ciclo de vida, permitindo acompanhar as mudanças e evoluções dos mesmos;
- o repositório é a base para decisões do projeto, sendo que, com base nas relações de dependência entre os requisitos, pode-se avaliar o impacto de alteração de cada um deles;
- o repositório propicia o reuso de requisitos, uma vez que eles já foram levantados e validados, podendo assim serem reutilizados em aplicações semelhantes.

Salem (2006) também ressalta a importância de armazenar os requisitos em um banco de dados. Segundo o autor, por meio desse repositório é possível a realização de consultas, buscas simples e avançadas e no contexto da rastreabilidade, a determinação do relacionamento entre os requisitos.

Como o intuito deste trabalho será o gerenciamento de requisitos, é interessante notar que as características levantadas anteriormente para os repositórios podem ser interessantes quando se busca armazenar e organizar os requisitos do software, sendo esse repositório útil para a

determinação do relacionamento e rastreabilidade durante o processo de desenvolvimento de software.

3.4 - Atividades de Gerenciamento de Requisitos

Em relação ao gerenciamento de mudanças, deve-se enfatizar que a Engenharia de Requisitos, sobretudo o Gerenciamento de Requisitos, não ocorre somente no início do desenvolvimento do software, mas sim durante todo o ciclo de vida do processo de desenvolvimento.

Sendo assim, se por um lado o Gerenciamento de Requisitos deve ser feito desde a etapa de Levantamento de Requisitos, buscando a melhor forma de armazenar as necessidades do usuário, prevenindo e diminuindo a quantidade de mudanças, por outro, deve gerenciar as mudanças que certamente irão ocorrer. O importante é buscar a chamada “mudança racional”, considerando a análise de impacto das propostas de mudanças, bem como o levantamento dos requisitos que serão afetados com a mudança sugerida. Como essa etapa reúne o conjunto de atividades que cuida do impacto e do custo das mudanças, ela deve ser aplicada a todas as mudanças propostas para os requisitos.

Considerando-se as várias fontes que abordam as atividades de Gerenciamento de Requisitos [Sommerville, 2003; SWEBOK, 2004; Pressman, 2006; Cordeiro, 2002], percebe-se que elas organizam e buscam separar o Gerenciamento de Requisitos em atividades, sendo que, para este trabalho, algumas das atividades discriminadas por Cordeiro (2002) são mais relevantes e serão apresentadas com mais detalhes:

- garantir a integridade do Documento de Requisitos;
- rastrear o relacionamento entre os requisitos;
- rastrear a dependência entre os requisitos e os artefatos do projeto;
- gerenciar mudanças;
- dar suporte ao processo de teste e validação do produto.

A partir dessa divisão, algumas dessas áreas serão exploradas na seqüência.

❖ Garantir a integridade do Documento de Requisitos

Sabe-se que, embora em alguns casos a solução proposta para o software seja tecnicamente de boa qualidade, muitos projetos falham se a solução desenvolvida não faz o que o cliente deseja. Nesse caso, pouco importa sua qualidade técnica. O papel-chave dos requisitos é mostrar aos desenvolvedores e usuários qual é a necessidade do sistema utilizando uma linguagem comum aos envolvidos. Power e Moyniahn (2003) definem como um dos principais papéis do Documento de Requisitos, o estabelecimento do acordo entre o cliente e aqueles que irão desenvolver o sistema. Para eles, o Documento de Requisitos pode servir como:

- contrato: nesse caso, o acordo tem a força de um contrato legal e formal entre as partes. Dessa forma, ele protege cada uma das partes de alguma insatisfação quando o sistema for entregue;
- aprovação: nesse caso, não há aprovação legal. O documento é revisto por um comitê da empresa que contratou o serviço de desenvolvimento do sistema. Esse caso é o mais comumente encontrado nas empresas de software;
- aceitação: nesse caso, o Documento de Requisitos não tem qualquer valor legal. É a maneira mais informal de tratar o acordo entre as partes.

A garantia da integridade do Documento de Requisitos é de fundamental importância para a garantia da satisfação daqueles que solicitaram o desenvolvimento do sistema. Para que a comunicação seja adequada do início ao fim do ciclo de vida do projeto, é fundamental um processo de gerenciamento de requisitos adequado que garanta a integridade do Documento de Requisitos, o que une os envolvidos no projeto.

Sommerville (2003) alerta ainda que, quando uma mudança nos requisitos é solicitada com urgência, a equipe de desenvolvimento é impelida a fazer a mudança no software para só depois voltar e realizar a alteração no Documento de Requisitos. Esse procedimento ocasiona, em geral, o desajustamento da implementação e da especificação dos requisitos. Isso ocorre porque as modificações no Documento de Requisitos podem ser esquecidas ou feitas de maneira inconsistente.

❖ Rastrear o relacionamento entre os requisitos

Um dos objetivos principais do processo de gerenciamento de requisitos é prover a avaliação de impacto da mudança em um projeto. Uma das informações fundamentais para prever esse impacto é a identificação do relacionamento entre os requisitos.

Quando uma requisição solicita a mudança de um requisito, é importante conhecer a dependência desse requisito com relação aos demais. A partir dessa informação, é possível avaliar se os requisitos que se relacionam com ele serão modificados ou não. A idéia é trabalhar com uma hierarquia dos requisitos. Somente a partir dessa avaliação hierárquica consegue-se identificar quais os requisitos que deverão sofrer modificação caso a mudança seja aprovada e levada à execução. Além disso, é possível prever o impacto das mudanças para avaliar o risco da mesma.

Apesar de ser muito fácil compreenderem-se os benefícios e as vantagens de haver a definição da relação entre os requisitos, essa é uma tarefa extremamente dispendiosa. A necessidade de armazenar todas essas informações aumenta a importância de um sistema com banco de dados para tal objetivo. Somente com os requisitos separados e relacionados podem-se obter essas informações com agilidade. Manter esse relacionamento sem o uso de uma ferramenta pode até ser possível em sistemas muito simples, mas com certeza, não será viável em software de médio e grande porte.

Uma das maneiras conhecidas e propostas para fazer o controle do relacionamento dos requisitos é o uso da Matriz de Rastreabilidade de Requisitos. Essa matriz é gerada relacionando-se todos os requisitos na primeira linha e na primeira coluna. Na intersecção linha/coluna é representada a existência ou não de relacionamento entre eles. Sommerville (2003) alerta para a dificuldade de obter-se e manter esse tipo de matriz. Como já foi mencionado, para sistemas de médio e grande porte, esse tipo de rastreamento do relacionamento entre os requisitos só pode ser feito com uso de uma ferramenta CASE (Computer-Aided Software Engineering), que dê suporte a essa atividade. Para sistemas pequenos também é proposto o uso de planilhas eletrônicas, processadores de texto ou bancos de dados simples para ajudar nessa atividade. O Standish Group (2004a) também destaca a importância dessa matriz, enfatizando a existência de diversas ferramentas de gerência de requisitos disponíveis no mercado e que se utilizam dela para permitir a relação entre os requisitos, fator fundamental para a rastreabilidade.

Na Tabela 3 - Exemplo de Matriz de Rastreabilidade de Requisitos [Sommerville, 2003], exemplifica-se uma Matriz de Rastreabilidade de Requisitos, segundo a notação proposta por Sommerville (2003), sendo que a letra U representa que um requisito usa o outro e a letra R que os requisitos têm um relacionamento fraco. No exemplo da tabela, está representado que o Requisito 1.1 utiliza o requisito 1.2 e que o Requisito 1.1 tem relacionamento fraco com o requisito 1.3.

Tabela 3 - Exemplo de Matriz de Rastreabilidade de Requisitos [Sommerville, 2003]

Requisitos	1.1	1.2	1.3	2.1	2.2	2.3	3.1	3.2
1.1		U	R					
1.2			U			R		U
1.3	R			R				
2.1			R		U			U
2.2								U
2.3		R		U				
3.1								R
3.2							R	

❖ Rastrear a dependência entre os requisitos e os artefatos do projeto

O IEEE (1990) define rastreabilidade como o grau em que o relacionamento pode ser estabelecido entre dois ou mais produtos do processo de desenvolvimento, especialmente produtos que contêm um antecessor-sucessor ou uma relação de mestre-subordinado a outro.

Para dar suporte à atividade de avaliação de impacto de mudanças, é de extrema importância que a informação de rastreabilidade entre os requisitos e os artefatos do projeto possa ser recuperada. Essa informação é necessária para o levantamento da estimativa de quantidade de esforço e custo para modificar os artefatos do projeto quando uma mudança é solicitada.

Diversos autores fazem referência à necessidade da rastreabilidade de maneira bi-direcional, ou seja, dos requisitos para os artefatos e vice-versa ([Castor et al.,2004; Ramesh & Jarke, 2001; Gotel & Finkelstein, 1997; Zisman & Spanoudakis, 2004]). Zisman e Spanoudakis (2004) relatam que apesar dos estudos existentes há mais de dez anos sobre a rastreabilidade, o uso de

ferramentas de suporte, e de se ter conhecimento da dimensão e da importância dela para a garantia da qualidade do processo de desenvolvimento de software, a rastreabilidade completa é, raramente, estabelecida na prática. Eles também mostram a existência de diferentes tipos de relação de rastreabilidade de requisitos. Algumas são baseadas nos tipos de artefatos que estão sendo relacionados e em como essa relação é usada nas diferentes atividades de desenvolvimento de software. A classificação é:

- horizontal, quando a relação acontece entre diferentes artefatos. Nesse caso, acompanha-se como o requisito foi definido no Documento de Requisitos, passou para a modelagem, código fonte e chegou até o plano de testes;
- vertical, quando se analisa um mesmo artefato, como por exemplo, a relação de dependência entre os requisitos ou entre os vários arquivos que compõem um código fonte. Nesse caso, também acompanhamos as várias versões de um mesmo documento ao longo do tempo.

Outro tipo de classificação proposto tanto por Zisman e Spanoudakis (2004) como por Letelier (2002), tem como referência o documento de requisitos. Dessa forma, temos:

- pré-rastreamento, caso se refira às fontes que originaram o Documento de Requisitos, rastreando cada um de seus requisitos funcionais ou não funcionais;
- pós-rastreamento, quando se trata dos diferentes artefatos que foram criados a partir do Documento de Requisitos nas outras fases do desenvolvimento de software.

Além dessas classificações, Zisman e Spanoudakis (2004) também propõem outros tipos de rastreabilidade, baseados na dependência entre os artefatos: generalização/refinamento, evolução, satisfação, sobreposição, conflito e contribuição.

Como tentativa de solucionar o problema da dificuldade de manter a rastreabilidade entre diferentes artefatos, Zisman e Spanoudakis (2004) propõem o uso de hiperdocumentos como a melhor maneira de inter-relacioná-los. No entanto, outro grande problema para manter a rastreabilidade de artefatos diferentes é que, em geral, eles são construídos e evoluem em ferramentas diferentes. Se o Documento de Requisitos estiver em um arquivo texto, o projeto e modelagem em uma ferramenta CASE, o código fonte em um compilador e o plano de testes em

uma planilha eletrônica, a tarefa de unir e interligar todos esses artefatos pode ser extremamente custosa ou inviável. A proposta para tentar a solução desse crítico e recorrente problema é o uso de uma linguagem de marcação, como XML (eXtensible Markup Language).

Quando os requisitos são bem gerenciados, a rastreabilidade pode ser estabelecida desde a origem do requisito até o seu nível de implementação, bem como no sentido contrário, da implementação até a sua origem [SEI, 2002]. Essa rastreabilidade exige que todas as fontes de requisitos sejam completamente conhecidas e que todos os níveis mais baixos de requisitos possam ser rastreados até a sua origem. Nesse trabalho publicado pela Software Engineering Institute – SEI (2002) dá-se ênfase à necessidade da rastreabilidade cobrir tanto as relações horizontais como verticais, atribuindo à rastreabilidade a tarefa de manter a consistência entre o plano de projeto de software, o cronograma de atividades e todos os artefatos de trabalho.

❖ Gerenciar mudanças

As mudanças nos requisitos irão ocorrer durante todo o ciclo de vida do sistema e essas alterações afetarão todas as fases do desenvolvimento do software, bem como os seus artefatos. O importante é que se tenha um processo de rastreabilidade dos requisitos que permita gerar uma mudança controlada, sendo possível prever o impacto da mesma e as conseqüências dessas para o sistema. Para tanto, é necessário que se conheça o relacionamento entre os requisitos, estabelecendo-se os diversos tipos de relacionamentos, como foi apresentado anteriormente. É com base na dependência entre eles que se pode ter fundamento para tomar decisão sobre uma mudança. Um bom conjunto de requisitos e seus relacionamentos orientam a avaliação da mudança.

Assim que novos requisitos surgem ou quando o cliente solicita uma mudança nos requisitos existentes, ou mesmo uma mescla dessas duas situações, isso deve ser caracterizado como uma “mudança”.

As propostas de gerenciamento de requisitos sempre passam pela etapa de análise de impacto de solicitações de alteração nos requisitos. Sommerville (2003) afirma que se deve utilizar um processo formal para o gerenciamento das mudanças propostas para os requisitos, permitindo, dessa forma, que todas as propostas de mudança sejam tratadas de modo consistente e que as

mudanças no Documento de Requisitos sejam feitas de maneira controlada. Ele identifica três estágios principais para o processo de gerenciamento de requisitos:

- **análise do problema e especificação da mudança:** trata-se do primeiro passo, no qual se busca a identificação da necessidade de alteração de um requisito;
- **análise de impacto da mudança:** verifica-se qual o efeito da mudança proposta. Segundo Sommerville (2003), o impacto da mudança é estimado com base nas modificações no Documento de Requisitos e, se apropriado, no plano de projeto de software e na implementação. Feita a análise do impacto da mudança, deve-se tomar a decisão de prosseguir ou não com a alteração no requisito. Enfatiza-se ainda que nessa fase é muito importante a existência de um depósito de dados com os requisitos, bem como o relacionamento entre eles;
- **implementação da mudança:** deve ser feita a partir da decisão de alterar o requisito do software. Trata-se da ação de colocar a mudança em prática. É enfatizado também o fato do Documento de Requisitos permitir a acomodação de novos requisitos, sem grande esforço, uma vez que este sofrerá alterações a cada requisito modificado.

Existem outras maneiras de dividirem-se as etapas de gerenciamento das mudanças, como proposto pelo Serpro (2002) e apresentado na seqüência:

- **receber as solicitações de alteração de requisitos:** fase na qual o engenheiro de requisitos recebe as solicitações de alteração de requisitos por formulário padronizado ou por meio de um sistema de requisição de mudança, como uma Intranet;
- **registrar os novos requisitos:** fase na qual as solicitações de alteração, os novos requisitos, ou a exclusão de algum existente, devem ser recebidos de maneira formal, ou por meio de um formulário padronizado, ou por meio de controle feito no software de requisição de mudança;
- **analisar impacto da mudança de requisitos:** fase na qual o impacto da mudança deve ser analisado, já que toda mudança tem um impacto associado à sua implementação. É necessário, portanto, análise criteriosa para avaliar o impacto do requisito a ser incluído, alterado ou excluído de cada um dos seus requisitos relacionados, o que pode ser obtido pelo uso das matrizes de rastreabilidade. Caso o impacto seja significativo, todos os

requisitos que têm relacionamento mapeado por meio da matriz de rastreabilidade com o requisito que recebeu a proposta de mudança, devem ser revistos;

- **elaborar o relatório de impacto:** fase na qual gera-se um relatório formal do impacto da mudança. O objetivo do relatório é manter um histórico de alterações para cada requisito. Dessa forma, torna-se possível uma visão cronológica das principais mudanças nos requisitos;
- **notificar os envolvidos:** fase na qual todos os envolvidos, ou seja, as pessoas para as quais pode haver um impacto devido à alteração de requisitos (alteração, inclusão ou exclusão de requisitos) devem ser notificados;
- **coletar métricas:** fase na qual as métricas devem ser coletadas para o acompanhamento das atividades de gerenciamento de requisitos. A coleta e utilização das métricas serão abordadas no Capítulo 4.

É interessante notar que nas duas abordagens os autores fizeram referência à necessidade de visualizarmos a relação entre os requisitos. Tanto um como outro versam em seus trabalhos sobre o uso da matriz de rastreabilidade de requisitos.

❖ **Dar suporte ao processo de teste e validação do produto**

Uma das características principais de um requisito é que ele seja testável. Dessa forma, pode-se considerar o Documento de Requisitos como base para a elaboração do plano de testes. Isso leva à necessidade de manter a integridade do plano de testes com os requisitos do software, ou seja, cada vez que um requisito for alterado, o plano de testes também deve ser revisto. Em algumas situações, novos casos de testes deverão ser incluídos quando os requisitos forem alterados. Em outras, os casos de testes deverão ser alterados ou excluídos. Sarkar (2005) propõe o uso de uma planilha eletrônica para controlar essa relação entre os requisitos e os casos de testes, como a maneira mais simples de se ter a aderência dos artefatos. Lott et al (2005) e Dalal et al. (1999) também baseiam seus trabalhos na dependência entre os requisitos do software e os casos de teste.

3.5 - Problemas da gerência de requisitos

Apesar da importância de uma gerência de requisitos efetiva durante o processo de desenvolvimento de software, definir um processo que propicie isso é uma tarefa muito difícil. Zisman e Spanoudakis (2004) definem dois problemas para manter a gerência de requisitos:

- não é trivial criar e manter a ordem e a organização dos requisitos quando se tem um grande número de requisitos;
- a informação obtida durante o processo de engenharia de requisitos não é estática. Ela muda a todo instante com o entendimento do domínio.

De acordo com Jones (1994), o principal risco que atinge 80% projetos de software é o da evolução de requisitos. Esse risco é caracterizado pelos seguintes fatos:

- levantamento tardio de requisitos, acarretando modificações e inclusões de novos requisitos no conjunto inicial já definido entre clientes e desenvolvedores;
- falhas para antecipar mudanças de requisitos, além de análise e planejamento para lidar com essas alterações.

Devido à volatilidade dos requisitos, isto é, ao fato inevitável do requisito ter que ser alterado ao longo do tempo, Jones (1994) cita problemas que acompanharão o processo de desenvolvimento do software:

- atritos entre usuários, equipe de desenvolvimento e gerentes, uma vez que os usuários sempre querem inserir ou alterar requisitos no produto, e os desenvolvedores não gostam de modificar o que já foi feito. Nesse contexto, o gerente do projeto busca encontrar a solução para o impasse;
- descumprimento do prazo vigente no acordo, uma vez que a estimativa original tenha sido feita sobre os requisitos inicialmente coletados;
- software de baixa qualidade e altos custos, já que a mudança, muitas vezes, não é planejada ou o impacto dela não é corretamente dimensionado.

Ramesh e Jarke (2001) definem os principais problemas da gerência de requisitos como sendo:

- falta de habilidade para manter o Documento de Requisitos consistente. Como o Documento de Requisitos é usado nas demais fases do desenvolvimento do sistema, tal problema compromete a qualidade do software;
- dificuldades e problemas no momento de levantar claramente as mudanças que devem ser feitas nos requisitos;
- falta de habilidade para chegar a um acordo sobre as mudanças-chave solicitadas pelos *stakeholders*;
- dificuldade para estimar, de maneira correta, os recursos necessários para viabilizar as mudanças solicitadas.

Os problemas apresentados anteriormente são mais evidentes quando se trata de grandes sistemas, nos quais o relacionamento entre os requisitos e a complexidade desse relacionamento aumenta muito. Sommerville (2003) enfatiza que uma gerência de requisitos efetiva para sistemas de grande porte dificilmente ocorrerá sem o uso de ferramentas que auxiliem e dêem suporte a todo processo de desenvolvimento de software. Ferramentas para o suporte da atividade de gerenciamento de requisitos serão abordadas na próxima seção.

3.6 - Ferramentas de gerenciamento de requisitos

A tarefa de gerenciamento de requisitos é extremamente dispendiosa e trabalhosa, tornando necessário o uso de ferramentas que auxiliem a equipe responsável pelo gerenciamento e desenvolvimento do software em todo esse processo. Existem diversas ferramentas à venda no mercado, e outras provenientes de trabalhos acadêmicos. Nesta seção serão abordadas algumas características inerentes às ferramentas de gerenciamento de requisitos, as razões pelas quais se usa esse tipo de ferramenta e, por fim, será apresentada uma comparação entre as existentes e disponíveis no mercado.

De acordo com o Standish Group (2004a), apenas 5% dos software são desenvolvidos com uso de ferramentas de gerenciamento de requisitos, o que pode explicar, em parte, os grandes

problemas das companhias em constituir uma gerência efetiva de requisitos e manter a rastreabilidade.

Existem diversas justificativas para o uso de uma ferramenta para a gerência de requisitos. Uma razão plausível é que, com o uso de uma delas, o gerenciamento de requisitos é feito de maneira muito mais consistente, armazenando todos os requisitos em uma base de dados. Além disso, pode-se ter o controle de versão desses requisitos e a possibilidade de geração da matriz de rastreabilidade, condição essencial para obtermos o relacionamento entre os requisitos, como visto na Seção 3.4, dentre outros benefícios.

Abaixo são listadas e comentadas algumas características das ferramentas de gerenciamento de requisitos, definidas por [Wiegers, 1999b].

- **gerenciamento de versão e mudanças:** a partir dos requisitos, definir uma *baseline* desses, ou seja, uma versão específica do documento de requisitos que contenha um conjunto de requisitos. Além disso, cada mudança feita no requisito deve ser armazenada, com o intuito de possibilitar a reversão desta alteração;
- **armazenamento dos atributos dos requisitos:** guardar dados inerentes aos requisitos do sistema. A ferramenta também deve permitir às pessoas envolvidas no projeto consultar e, eventualmente, atualizar os valores desses atributos. Em geral, as ferramentas já armazenam algumas informações dos requisitos, como a data de criação e o número de versão do mesmo. Outros atributos também podem ser criados para o requisito, como:
 - Autor: pessoa que levantou o requisito junto do usuário.
 - Pessoa responsável: indivíduo responsável pelo ciclo de vida daquele requisito, uma vez que esse foi levantado.
 - Origem do requisito: indica qual a necessidade do cliente que originou esse requisito.
 - Número da versão de distribuição: cada versão de distribuição do software deve rotular a versão do requisito que o gerou.
 - Status: indica qual o andamento do requisito no processo de desenvolvimento de software, ou seja, se ele já foi modelado, implementado ou testado.

- **Prioridade:** define qual a necessidade de implementação desse requisito com relação aos demais.
 - **Custo:** pode ser expresso tanto em horas quanto em valores monetários, de acordo com o hábito dos que estão desenvolvendo o software.
 - **Dificuldade:** trata-se de uma escala que define o grau de dificuldade de implementação daquele requisito. Em geral está intimamente ligado com o custo.
 - **Risco:** define o risco do requisito dentro do projeto. Pode estar tanto atrelado ao impacto no produto final, caso a implementação desse requisito falhe, bem como à possibilidade ou não de implementação desse requisito, considerando-se os aspectos técnicos.
- **relacionamento entre os requisitos:** armazenar a relação dos requisitos, tornando possível a geração da matriz de rastreabilidade. Essa característica é essencial para qualquer ferramenta de gerenciamento de requisitos;
 - **relacionamento entre os requisitos e os outros elementos do sistema:** rastrear requisitos com outros componentes do sistema ajuda a garantir que os artefatos estejam consistentes ao longo do desenvolvimento e que o Documento de Requisitos esteja condizente com alterações em qualquer um dos artefatos produzidos ao longo do processo;
 - **acompanhamento do status:** acompanhar o status de cada requisito durante o decorrer do projeto é muito importante. Assim, é possível saber quais requisitos já foram implementados e quais ainda precisam ser. Caso um requisito esteja atrasado, e seja de alta prioridade, podem-se alocar recursos de modo a recuperar o atraso;
 - **consultas sobre os requisitos:** possibilitar ao usuário da ferramenta de gerenciamento de requisitos a consulta dos requisitos por meio de filtros, com palavras-chave, data, prioridade ou qualquer expressão de busca que ele deseje montar;
 - **controle de acesso:** possibilitar na ferramenta a configuração de permissão para indivíduos ou grupos de usuários, para que esses tenham acesso a níveis diferentes de informação. Os grupos de usuários podem ser separados, por exemplo, por função da pessoa no processo de desenvolvimento. Sendo assim, os testadores podem visualizar os requisitos, mas não alterá-los. Uma possibilidade é uma interface web da ferramenta,

permitindo que uma equipe separada geograficamente possa trabalhar num mesmo projeto;

- **comunicação com os *stakeholders*:** prover formas para facilitar a comunicação entre os envolvidos no projeto. Essa comunicação pode ocorrer por meio de correio eletrônico, grupos de discussão ou vídeo-conferência.

Zisman e Spanoudakis (2004) destacam a dificuldade do gerenciamento de requisitos e apontam o uso de uma ferramenta como o caminho para torná-lo viável. Entretanto, levantam alguns pontos negativos das ferramentas existentes. Segundo eles, as disponíveis no mercado não se adequam às características próprias de cada empresa, ou seja, elas não permitem um nível alto de refinamento para as necessidades específicas de cada companhia. Eles são ainda mais críticos com relação às ferramentas existentes no mercado. De acordo com os autores, a rastreabilidade gerada pelas ferramentas existentes no mercado pode ser dividida em três categorias:

- **Rastreabilidade manual:** nesse caso, toda a rastreabilidade entre os requisitos e artefatos é feita de maneira manual. O usuário da ferramenta é o responsável por criar e manter a rastreabilidade. Caso haja falha nesse processo, a rastreabilidade estará comprometida. A maioria das ferramentas existentes e disponíveis no mercado inclui-se nessa categoria.
- **Rastreabilidade semi-automática:** nesse caso, parte da rastreabilidade é feita de maneira automática e parte dela é feita pelo usuário da ferramenta. Representam grande avanço em relação às ferramentas da primeira categoria, pois garantem parte da rastreabilidade, mas ainda dependem do trabalho do usuário em manter parte da rastreabilidade.
- **Rastreabilidade automática:** nesse caso, toda a rastreabilidade entre os requisitos e artefatos deve ser mantida de maneira automática. Não existe ainda disponível para comercialização nenhum tipo de ferramenta nesse molde, somente protótipos que ainda necessitam de avanços para tornarem-se produtos confiáveis e de uso prático.

Para Zisman e Spanoudakis (2004), apesar das ferramentas possuírem poderosas formas de visualização das informações e de navegação entre os dados existentes, o fato de a consistência ser feita manualmente, compromete a integridade da rastreabilidade, principalmente no que diz respeito a software grandes e complexos. Outro agravante levantado para as ferramentas existentes no mercado, é que elas não se comunicam e interagem com as diferentes ferramentas

utilizadas durante o processo de desenvolvimento de software, como editores de texto, planilhas eletrônicas, gerenciadores de projeto, software para modelagem de dados, compiladores e software de controle de versão. Dessa maneira, fica praticamente impossível garantir a rastreabilidade automaticamente. O caminho apontado é o uso de linguagens de marcação, como o XML ou XMI (padrão da indústria para a integração entre as ferramentas de modelagem UML). Nesse sentido, existem ferramentas que têm a possibilidade de exportar os dados em XML. Outro caminho apontado é o uso de documentos com *hyperlinks* para relacionar os mais diversos artefatos envolvidos no processo.

Existem diversas ferramentas comerciais que dão suporte ao gerenciamento de requisitos [Incose, 2005]. No entanto, no contexto deste trabalho não se fez uma investigação mais detalhada sobre elas, pois os aspectos de gerenciamento de requisitos aqui abordados levam também em consideração o modelo de casos de uso, que não é o caso das ferramentas encontradas. Além disso, a ferramenta de requisitos aqui desenvolvida está inserida no contexto de um ambiente com diversas funcionalidades, capaz de dar suporte, além do gerenciamento de requisitos, a outras funcionalidades do processo de desenvolvimento de software.

3.7 - Considerações Finais

Nesse capítulo foram apresentados os principais conceitos e alguns trabalhos relacionados ao tema da dissertação. Explorou-se a fase de Gerenciamento de Requisitos, que foi apresentada superficialmente no Capítulo 2, dando-se ênfase na Rastreabilidade de Requisitos. Dessa forma, foram comentadas a necessidade e as vantagens de ter-se um repositório para os requisitos do software. Também foram discutidas algumas tarefas que compõe o Gerenciamento de Requisito, focando sobretudo as tarefas relacionadas à manutenção de integridade do documento de requisitos, relacionamento entre os requisitos e a gerência das mudanças que os requisitos sofrem ao longo do processo de desenvolvimento de software, bem como a importância da rastreabilidade para essas tarefas.

O capítulo também abordou dificuldades de obter-se um gerenciamento de requisitos e os problemas enfrentados. Finalmente, comentou-se sobre as ferramentas de gerenciamento de

requisitos, seus tipos e sua importância para a implantação de uma gestão de requisitos eficiente.

Capítulo 4

Trabalhos Relacionados

4.1 - Considerações Iniciais

Neste capítulo serão apresentados e comentados alguns trabalhos relacionados às atividades de Gerenciamento de Requisitos, assunto tratado no capítulo anterior. Esses artigos foram provenientes de um levantamento bibliográfico *ad-hoc* e também da aplicação de uma Revisão Sistemática baseada nos trabalhos de Kitchenham (2004) e Biolchini et al (2005). Do conjunto de artigos da revisão sistemática que foram aceitos, apenas os que tiveram uma maior influência neste trabalho é que serão apresentados.

Assim, na Seção 4.2 discutem-se os artigos, individualmente e, na Seção 4.3, apresentam-se as considerações finais, destacando-se os pontos mais importantes para o contexto deste trabalho.

4.2 – Apresentação dos Artigos

Os artigos aqui apresentados são aqueles que mais se relacionaram e deram subsídios aos vários conceitos e alternativas adotadas na criação do ambiente proposto neste trabalho. Para cada artigo apresentam-se título, autor, ano de publicação e resumo com os principais pontos associados ao trabalho.

❖ SemiAutomatic Tracing of Requirement Versions to Use Cases - Experiences & Challenges; Alexander, Ian; 2003

Esse artigo descreve a experiência de criação de uma ferramenta para gerenciamento de requisitos que permite a rastreabilidade entre requisitos de software e casos de uso, bem como o

controle de versões da evolução dos requisitos. Segundo o autor, o trabalho de manter a rastreabilidade atualizada é uma atividade braçal mesmo que auxiliado por ferramentas de suporte disponíveis no mercado. Esse artigo é uma continuidade do trabalho do mesmo autor [Alexander 2002] no qual foi descrito um ambiente para a rastreabilidade de casos de uso na ferramenta comercial DOORS, baseada na customização de relatórios para hipertextos, gerando uma rastreabilidade entre os artefatos. O autor discorda da proposta de Zisman et al (2003) focada na rastreabilidade totalmente automática e baseada na análise do texto, pois, segundo Alexander, caso ocorra algum erro na detecção automática da rastreabilidade, os erros irão acumular-se ao longo do processo, de forma negativa.

A ferramenta utilizada nesse artigo também foi a DOORS, contando com a implementação de um módulo específico para o tipo de rastreabilidade desejado por meio da linguagem DXL (que permite acesso à base de dados da ferramenta) implementando, assim, o *Scenario Plus Toolkit*. O artigo descreve a experiência da adequação da ferramenta DOORS para suporte ao *template* de requisitos do VOLARE rastreando as dependências e os conflitos entre os requisitos e a relação com os casos de uso. O projeto previa ainda a possibilidade de exportação do modelo de casos de uso para HTML. Outra proposta do desenvolvimento é a possibilidade de controle de versão dos requisitos. O relacionamento entre os requisitos foi obtido tentando-se encontrar o nome de cada Caso de Uso nos passos dos demais casos de uso. Dessa forma, a detecção não foi feita utilizando-se técnicas avançadas de lógica *fuzzy*, mas, simplesmente, buscando-se os casos mais evidentes. Entretanto, segundo o autor, nem sempre é possível obter a rastreabilidade dessa forma. Os *links* de rastreabilidade podem ser de “dependência”, “satisfação” ou “conflito”. O projeto mapeou os requisitos formatados do VOLARE na ferramenta comercial DOORS evitando assim os *links* manuais. Deu-se atenção especial para a interface da ferramenta gerada, tomando-se cuidado para que fosse a mais intuitiva possível.

Segundo o autor, a vantagem da análise textual dos requisitos é a acomodação de qualquer quantidade de tipos de *links* sem a necessidade de codificação ou sobrecarga das estruturas de dados. Por outro lado, alerta sobre o perigo da geração de *links* errados. Por esse motivo, defende a abordagem semi-automática, uma vez que erros gerados nessa fase terão custos muito altos no decorrer do processo de desenvolvimento de software. Ele também explora o fato do Gerenciamento de Requisitos sempre buscar simplificação da escrita e da documentação das

necessidades do usuário, evitando termos técnicos nos documentos de requisitos. Por outro lado, a rastreabilidade acaba retomando essa complexidade, com ligações e representações de rastreabilidade. Para minimizar esse efeito, ele justifica o uso de poucos tipos de dependência, como “dependência”, “satisfação” e “conflito”. Um ponto que o autor salienta como uma evolução seria a exportação dos dados não somente para HTML, mas de forma complementar para um servidor, sendo disponível em ambiente de Intranet/Internet.

O autor conclui que é um grande desafio da Engenharia de Requisito a construção de ferramentas poderosas que estimulem as pessoas a documentarem, da melhor forma possível, as necessidades do usuário e que permitam a construção e manutenção dos *links* de rastreabilidade.

A relação desse artigo com o trabalho proposto nesta dissertação está no mapeamento da relação de dependência entre os casos de uso. A diferença está na forma com que essa dependência é obtida. No trabalho apresentado no artigo, ela é fruto de palavras utilizadas na especificação dos casos de uso, enquanto que na ferramenta desenvolvida neste trabalho, é produto da dependência entre os dados vinculados nos requisitos funcionais, conforme será discutido e exemplificado nos Capítulos 5 e 6.

❖ **When and How to Visualize Traceability *Links*?; Marcus, Adrian; Xie, Xinrong; Poshyvanyk, Denys; 2005**

Esse artigo discute a visualização de *links* de rastreabilidade, explorando o momento, o contexto, a circunstância e a forma que eles devem ser exibidos para o usuário. Apesar de aparentar simples e trivial, os autores procuram mostrar que se trata de uma tarefa que pode tornar-se complexa e que deve ser realizada com muita atenção. Para dar suporte a esse processo, é apresentado ainda o protótipo de uma ferramenta CASE chamada TraceViz.

O autor também sugere algumas propriedades que devem ser armazenadas com relação aos artefatos rastreados e com relação aos *links* de rastreabilidade, a saber:

i) com relação ao artefato: nome do artefato; tipo do artefato (documento de requisitos, diagramas, casos de teste, manual, etc.); localização; momento de criação; última atualização; e número de versão.

ii) Com relação aos *links* de rastreabilidade: método de levantamento (automático, semi-automático ou manual); momento de criação; última atualização; descrição do relacionamento; e autor do relacionamento.

O tipo do *link* de rastreabilidade é também discutido no trabalho. São mostradas diversas formas de classificação do *link* de rastreabilidade, como por exemplo: satisfação; dependência; envolvimento; e relacionamento ou condição; conteúdo; documentação; evolução; e abstração. O artigo fala da necessidade do usuário criar suas próprias categorias de *link*, adequando a rastreabilidade às suas necessidades.

Com relação à visualização do *link* entre os artefatos, o artigo aponta a matriz de rastreabilidade e grafos como as formas mais comuns, alertando que eles são insuficientes, uma vez que esse relacionamento pode ser melhor definido utilizando-se os atributos e as características apresentados anteriormente. Outro ponto para o qual é feito um alerta, é a falta de padronização para armazenar e representar os *links* de rastreabilidade.

Os autores afirmam que não existe rastreabilidade totalmente automática. Ela sempre deve passar pelas mãos de um componente humano. Sendo assim, deve haver uma ferramenta de suporte que ofereça ao usuário o máximo de informação para tomar a decisão de manutenção desses *links*. Nesse sentido, é proposto o seguinte conjunto de requisitos para dar suporte à recuperação, navegação e manutenção de *links*:

1. Visualizar e armazenar rastreabilidade entre vários artefatos, independentemente do método de extração usado.
2. Relacionar ou integrar ferramentas de recuperação com *links* de rastreabilidade.
3. Permitir ao usuário navegar nos *links* de rastreabilidade por meio de múltiplos tipos de interações de usuários.
4. Permitir ao usuário adicionar, apagar e editar as propriedades dos *links* existentes e seus artefatos de conexão.
5. Integrar com IDE (Integrated Development Environment) visando suporte à visualização dos artefatos do software e dos *links* de rastreabilidade.
6. Interoperar com outras ferramentas de engenharia de software (por exemplo, ferramentas de análise, ferramentas de gestão/administração de documentos, etc.).

7. Capturar e manter históricos de navegação para *links* de rastreabilidade.
8. Prover gerenciamento de configuração focada nas propriedades do *link* e integrada com ferramentas de gerenciamento de código fonte.
9. Dar suporte a vários formatos de representação de dados.
10. Dar suporte à consultas e aos filtros sobre os *links* de rastreabilidade.
11. Oferecer flexibilidade e customização para o usuário.
12. Permitir análise e sumarização dos dados envolvidos no processo de rastreabilidade.

A partir desses requisitos, foi desenvolvido um protótipo da ferramenta - a TraceViz - integrada com a IDE do Eclipse. Nela, os *links* de rastreabilidade são armazenados em um arquivo XML. Nem todos os requisitos foram implementados. Atualmente, têm-se implementados os itens 1, 2, 4, 5, 11 e 12 dos listados anteriormente. O usuário pode ainda definir suas próprias categorias de *link*.

A idéia de visualização dos *links* de rastreabilidade foi implementada no ambiente COCAR, no qual é permitido ao usuário verificar o relacionamento existente entre dois artefatos – o documento de requisitos e o Modelo de Casos de Uso - conforme será discutido nos Capítulos 5 e 6. Algumas das sugestões feitas nesse artigo serão propostas de futuros desenvolvimentos sobre a ferramenta em questão e serão abordados no Capítulo 7 deste trabalho.

**❖ Good requirements practices are neither necessary nor sufficient;
Davis, Alan M.; Zowghi, Didar; 2005**

O título controverso do artigo foi criado para levantar a discussão sobre as boas práticas de Engenharia de Requisitos. O artigo afirma que as tais práticas de Engenharia de Requisitos são aquelas que reduzem os custos de desenvolvimento e aumentam a qualidade do produto de software. Entretanto, afirma que poucas dessas práticas foram realmente validadas como “boas” e que diferentes autores têm seus próprios pontos de vista sobre o conceito de boas práticas, como pode ser visto na Tabela 4.

Tabela 4 - Consenso de “boas” práticas de engenharia de requisitos[Davis & Zowghi,2005].

	Robertsons	Sommerville e Sawyer	Wiegers	Young
Qualidade	Ponto de passagem e caminho para a qualidade	Controlar o gerenciamento de requisitos é a base para a qualidade	Precisa definir o critério de aceitação	Possuir controle de qualidade independente
Stakeholders	Identificar os stakeholders relevantes	Identificar e consultar os stakeholders	Identificar classes de usuários	Compreender o papel dos stakeholders e envolvê-los no processo
Revisões e inspeções	Realizar revisões e inspeções	Organizar inspeção formal	Validar os requisitos com inspeções	Utilizar revisões e inspeções para remover defeitos do processo e dos artefatos

Para justificar as afirmações feitas no título do trabalho de que boas práticas aplicadas na Engenharia de Requisitos não são nem suficientes tampouco necessárias, os autores baseiam-se na Figura 6. Nela são mostradas quatro diferentes situações relacionando o uso ou não de boas práticas de Engenharia de Requisitos com o sucesso ou fracasso dos produtos de software gerados. Para mostrar que a Engenharia de Requisitos não é suficiente para o sucesso de um produto de software, basta encontrar uma ocorrência de algum software desenvolvido em que foram utilizadas as boas práticas de Engenharia de Requisitos e que o produto criado não atingiu sucesso ao final, ou seja, que esteja na região B da figura. Os autores argumentam que não é difícil encontrar software que esteja nessa situação e que, apesar de ter passado por um processo meticuloso de Engenharia de Requisitos, esteja repleto de defeitos. Dessa forma, prova-se que as boas práticas não são suficientes para garantir a qualidade do produto final gerado.

Da mesma forma, provar que as boas práticas de Engenharia de Requisito não são necessárias, seria mostrar que existe ao menos um software que foi desenvolvido sem a aplicação de boas práticas de Engenharia de Requisitos e que resultou em sucesso no produto obtido, ou seja, que esteja na região C da figura. Segundo os autores, encontrar algum software que se encaixe nessa categoria também é perfeitamente possível considerando-se uma situação em que houve perfeito entendimento das necessidades do cliente e não houve alterações dessas necessidades ao longo do projeto. Apesar da existência da possibilidade de uma situação como essa ocorrer, ela é classificada no artigo como uma ocasião de grande sorte.

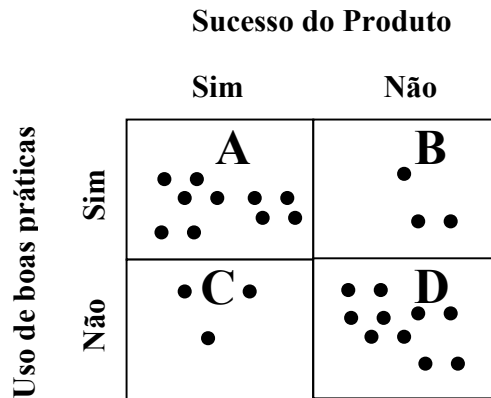


Figura 6 – Relação entre as boas práticas de Engenharia de Requisitos e o sucesso do produto [adaptado de Davis & Zowghi, 2005]

A conclusão do trabalho é que, apesar das boas práticas de Engenharia de Requisitos para o sucesso de um projeto de software não serem nem suficientes e tampouco necessárias, elas devem ser buscadas. O artigo também destaca que:

- não se devem descartar as boas práticas de Engenharia de Requisitos somente porque o projeto falhou;
- resultados a longo prazo são mais importantes que resultados a curto prazo, mesmo que esses sejam mais fáceis de serem mensurados;
- requisitos são uma parte pequena de um grande caminho.

❖ **Concordance, Conformance, Versions, and Traceability; Munson, Ethan V.; Nguyen, Tien N.; 2005**

Nesse artigo são apresentados os resultados do projeto *Software Concordance Project*, que estuda aspectos de rastreabilidade de software. As pesquisas são baseadas em quatro aspectos:

- a análise efetiva da rastreabilidade precisa abranger um grupo completo de documentos produzidos pelo processo de desenvolvimento de software;
- em um futuro previsível, grande parte da documentação de software será escrita em linguagem natural, como textos informais ou diagramas, e uma análise automática completa desses documentos não será rastreável;
- as práticas de rastreabilidade somente melhorarão quando houver ferramentas que reduzam o esforço requerido para manter essa rastreabilidade;
- a documentação estruturada e o uso de hiperdocumentos são uma boa base para o suporte à rastreabilidade em ambientes de desenvolvimento integrados.

Como fruto desse projeto, foi desenvolvida uma ferramenta para suporte à representação de relacionamentos de rastreabilidade entre elementos de documento de software com o uso de *hyperlinks*. Essa ferramenta foi denominada *Software Concordance Editor*. Como continuidade dos trabalhos desenvolvidos pelos autores, uma outra ferramenta também foi criada com o intuito de realizar o controle de versão para a rastreabilidade.

O *Software Concordance Editor* é uma ferramenta para criação de código Java e de arquivos XML. Os códigos Java são representados como uma *Árvore de Sintaxe Abstrata* (estrutura de dados específica utilizada para organização e representação de dados) e os arquivos XML como a representação clássica de *árvore de elementos*. Tanto os nós do código Java como do XML podem ser ligados a outros por meio de *hyperlinks*. Também é permitida a inserção de *links* com outros tipos de mídia como imagens, áudio e outros arquivos texto.

Entretanto, é sabido que os sistemas não são estáticos e estão em contínua evolução, sendo de extrema importância refletir essas mudanças nos *links* de rastreabilidade. Para tanto, os autores acreditam que seja possível construir ferramentas que direcionem a atenção dos desenvolvedores para indicar os documentos que possam não estar em acordo com os demais. Esse processo foi

chamado de Análise de Conformidade. Eles propõem o uso de rótulo de tempo nos documentos e nos *links*. Dessa forma, cada vez que for feita alguma alteração em algum documento ou *link*, as marcações de tempo serão atualizadas, indicando a necessidade de alteração dos artefatos relacionados.

Com relação ao gerenciamento de versão dos artefatos, o trabalho propõe e explora o controle de versão de cada um dos nós atômicos criados durante a geração do código fonte em Java ou durante a criação dos arquivos XML. Dessa forma, cada nó deve conter seu identificador único com toda sua evolução ao longo do processo de desenvolvimento de software. A rastreabilidade, nesse caso, é mantida por cada parte de artefato envolvido no processo.

A ferramenta desenvolvida ainda não se encontra totalmente estável e não foram realizados testes em ambientes reais de desenvolvimento, sendo esses os próximos passos a serem seguidos pela equipe de pesquisadores. É enfatizada a necessidade de equilíbrio entre o uso de técnicas formais de rastreabilidade com técnicas manuais, que permitem maior flexibilidade para os usuários, permitindo-lhes interagir com a ferramenta e tomar decisões em determinados momentos. A discussão de técnicas manuais e automáticas, assim como o formalismo exigido para a obtenção da rastreabilidade no ambiente COCAR serão discutidos com maiores detalhes no Capítulo 5.

❖ **A reusable traceability framework using Patterns; Kelleher, Justin; 2005**

Nesse artigo é apresentado um *framework* para modelar a rastreabilidade constituído do TRAM - *Traceability Metamodel* - e do TRAP - *Traceability Process* -, no qual os artefatos são publicados com suas respectivas rastreabilidades. O processo de modelagem e de apresentação da rastreabilidade é feito por meio de padrões de rastreabilidade, provendo assim, a padronização da visualização, comunicação e reuso das práticas de rastreabilidade. Uma ferramenta para dar suporte a esse framework também é apresentada e descrita no artigo.

O *framework* seguiu os princípios do modelo apresentado por Latelier (2002), sendo baseado na UML. Ele é dividido em quatro camadas. A primeira, denominada MOF – *Meta Object Facility*,

é um meta-modelo. Trata-se da fundação para definir qualquer linguagem de modelagem. No segundo nível está o TRAM, que é uma instância da MOF. É uma definição precisa das regras e construções necessárias para a criação da próxima camada, a TRAP. No TRAP estão sintetizadas as melhores práticas que foram levantadas e desenvolvidas durante o tempo dessa pesquisa. Elas estão caracterizadas por passos que descrevem as atividades, os papéis e os artefatos relacionados. A última etapa é o processo concreto de rastreabilidade em produção, com base na TRAP. A proposta do modelo é que ele possa ser configurado e utilizado em qualquer projeto.

Durante a criação do TRAM foram identificados padrões de rastreabilidade. Esses padrões são descritos em UML, que constitui a parte lógica, enquanto que o meta-modelo do TRAM tem natureza física. Segundo o autor, esses padrões de rastreabilidade descrevem as melhores práticas, o projeto de soluções de rastreabilidade e a captura das experiências de sucesso. O artigo é focado na parte do processo para obter a rastreabilidade e não fornece detalhes dos padrões construídos para a parte do meta-modelo.

❖ **A heterogeneous solution for improving the return on investment of requirements traceability; Cleland-Huang, Jane; Zemont, Grant; Lukasik, Wiktor;2004**

O artigo apresenta uma abordagem para maximizar o retorno sobre o investimento feito em rastreabilidade. Para tanto, utiliza soluções isoladas ou prontas como a matriz de rastreabilidade ou um simples gerenciamento de requisitos. A solução proposta, definida como TraCS (*Traceability for Complex Systems*), define estratégias de rastreabilidade de acordo com a categoria do projeto, focando minimizar o risco inerente à evolução do software e otimizar o retorno do esforço feito com a rastreabilidade. O autor define quatro tipos de rastreabilidade:

- **Links simples:** método de rastreabilidade no qual são definidas, simplesmente, as ligações entre dois artefatos. Nesse caso, uma simples consulta informa quais são os artefatos que possuem relacionamento entre si. Em alguns casos, podem ser definidos atributos para esses *links*. O ponto fraco desse tipo de rastreabilidade é a dificuldade em manter grande número de *links*.

- **Recuperação semântica de links:** método de rastreabilidade no qual os *links* não são determinados pelo usuário e sim por meio da geração dinâmica e automática da rastreabilidade. A construção dessa rastreabilidade ocorre com palavras-chave que existem nos artefatos. Entretanto, essa recuperação semântica funciona somente se existir uma forte correlação léxica entre os artefatos. Sua grande vantagem é a geração e a manutenção automática dos *links*.
- **Links executáveis:** método de rastreabilidade no qual se relacionam requisitos a modelos e simulações. Para alguns requisitos que envolvem desempenho e tempo de resposta, em geral, existe uma especificação com modelos e simulações. Nesses casos, os requisitos que os definem podem estar ligados à variáveis desses modelos.
- **Requisitos não funcionais:** método de rastreabilidade para mapear a rastreabilidade de requisitos não funcionais por meio de padrões de projeto e uso da UML o que, segundo os autores, apresenta uma relação custo-benefício bastante satisfatória.

Essas quatro formas de rastreabilidade servem para ilustrar que os diferentes tipos de requisitos são melhor rastreados com métodos distintos, sendo que cada técnica possui seus pontos positivos e negativos, o que justifica a análise do custo-benefício da aplicação da rastreabilidade. Para cada uma das técnicas, devem ser observadas três questões que caracterizam sua aplicação:

- **Automação:** é a capacidade de dar suporte às consultas para geração dos *links* de rastreabilidade. Existem três níveis de automação:
 - não-automática, a qual requer total interferência humana na criação dos *links*;
 - semi-automática, na qual uma consulta pelos *links* sugere uma lista de possíveis candidatos, cabendo ao usuário a determinação de quais são ou não satisfatórios;
 - completamente automática, na qual os *links* são gerados todos de maneira automática, rastreando os artefatos envolvidos e disparando eventos com a mudança de cada um deles.
- **Corretitude** (*correctness*): é a capacidade dos *links* estarem condizentes com a realidade do projeto em determinado momento do tempo. Pode ser dividido em duas características: manutenibilidade e precisão. A primeira refere-se à capacidade de, dado um conjunto

inicial de *links* de rastreabilidade, preservá-los ao longo do projeto. A segunda está vinculada com o acerto dos *links* representados. Pode ser determinada pela divisão entre o número de *links* relevantes recuperados pelo total de *links* recuperados

Segundo o autor, a manutenibilidade pode ser melhorada com a redução do número de *links*; entretanto, isso pode causar a redução do escopo da rastreabilidade.

- **Cobertura** (*coverage*): é a capacidade de rastrear todos os tipos de requisitos nos mais diferentes níveis de abstração. Diz respeito à abrangência da rastreabilidade dos requisitos.

Após ter levantado alguns tipos de rastreabilidade (*links* simples, recuperação semântica de *links*, *links* executáveis e rastreabilidade de requisitos não funcionais) e de ter explorado características gerais da rastreabilidade (automação, corretitude e cobertura), os autores abordaram o retorno sobre o investimento que a rastreabilidade pode trazer para o processo de desenvolvimento de software. Segundo ele, em alguns casos, como por exemplo, na rastreabilidade de sistemas críticos, apesar do investimento ser alto, ele é justificado pela importância dos requisitos que serão mapeados e acompanhados. Por outro lado, sistemas simples não exigem necessidade de altos investimentos na rastreabilidade.

Com base nisso, são definidas algumas estratégias de rastreabilidade:

- **Maximização da geração de *links* dinâmicos**: utilizar *links* dinâmicos para reduzir o custo de manutenção.
- **Rastreabilidade por objetivo**: analisar o porquê da criação do *link*, para evitarem-se *links* desnecessários e focar em *links* que podem ser mais importantes para o processo de desenvolvimento de software.
- **Seleção dos mecanismos mais efetivos de rastreabilidade**: utilizar várias técnicas de rastreabilidade para mapear os diversos requisitos, aplicando técnicas específicas para cada caso.
- **Diferenciação entre rastreabilidade *throw-away* e *long-term***: separar as rastreabilidades que devem ser úteis somente ao longo do desenvolvimento, daquelas que devem ser mantidas durante todas as fases do processo de desenvolvimento de software.

A partir dessas estratégias, deve-se proceder o equacionamento do investimento a ser feito em rastreabilidade para obter o retorno satisfatório e condizente. Para tanto, os seguintes aspectos devem ser levados em consideração:

- 1- Custo para estabelecer e manter os *links* de rastreabilidade.
- 2- Custo para executar uma análise manual da rastreabilidade quando uma mudança ocorrer e nenhum *link* de rastreabilidade estiver disponível.
- 3- Custo pelo impacto de corrigir erros decorridos pelo impacto de uma mudança.

Os autores fazem comparações entre técnicas de rastreabilidade com o custo e riscos estimados das mesmas. Além disso, foi construído um protótipo implementando para a TraCS. Nele o usuário pode definir *links* de rastreabilidade por meio da matriz de rastreabilidade, como solicitar a geração dinâmica de *links* por meio da recuperação semântica e da Rastreabilidade Baseada em Eventos (EBT-*Event Based Traceability*). No protótipo da ferramenta pode ser inserida uma solicitação de alteração em um conjunto de requisitos, sendo então exibido um relatório de impacto.

O maior objetivo do artigo foi propor a idéia de abordagem da rastreabilidade com várias técnicas aplicadas de modo sinérgico, melhorando o retorno sobre o investimento. A integração de novas técnicas é apresentada como uma solução para mapear requisitos antes não controlados. Segundo a proposta, os investimentos em rastreabilidade devem ser guiados pelas características do projeto.

❖ **Improving Software Quality through Requirements Traceability Models; Salem, Ahmed M. ;2006**

O artigo enfatiza a importância da rastreabilidade de requisitos para a garantia da qualidade na produção de software. O autor considera que a rastreabilidade possibilita que a modelagem do que foi especificado para o software possa ser verificada de forma apropriada e que os requisitos funcionais possam ser validados. Também salienta que os testes são beneficiados quando o

processo de rastreabilidade é efetivo, pois, caso ocorram falhas, é possível identificar, documentar e revisar os requisitos envolvidos.

O autor destaca a importância da engenharia de requisitos e, sobretudo, da rastreabilidade de requisitos, enfatizando que alguns problemas de rastreabilidade ocorrem devido à informalidade dos métodos existentes. No artigo é proposto um modelo no qual o *link* entre os artefatos é baseado na dependência entre eles e uma mudança nos requisitos gera um evento que notifica todos os dependentes. Também é proposta a identificação de *links* de rastreabilidade pelo uso de palavras-chave coincidentes entre os artefatos.

O modelo é dividido em três etapas: a primeira, denominada *Traceability Engine Component* (TEC), é usada para ajudar os desenvolvedores a correlacionar o código-fonte com os requisitos do sistema, permitindo o armazenamento de todos os requisitos num banco de dados, tornando possível a criação da matriz de rastreabilidade. A segunda etapa é denominada *Traceability Viewer Component* (TVC) e permite a visualização dos *links* e requisitos do sistema. A terceira etapa é denominada *Quality Assurance Interface* (QAI) e é responsável pela validação e verificação dos requisitos, além de garantir a rastreabilidade entre os requisitos e as partes do código-fonte. Caso haja alguma falha, um e-mail é enviado para o desenvolvedor, indicando que determinada parte do código-fonte tem que ser refeito.

O artigo enfatiza ainda a necessidade de colocar os requisitos em um banco de dados para permitir organização, consultas e determinação do relacionamento entre eles. O ambiente COCAR segue esse conceito, conforme será explorado no Capítulo 5.

❖ **Uma Proposta para Melhorar o Rastreamento de Requisitos; Toranzo, Marco; Castro, Jaelson. F. B.; Mello, Elton; 2002**

Segundo os autores, diversos trabalhos que visam solucionar muitos problemas do pré e pós-rastreamento foram propostos na última década. Gotel (1996) deu ênfase para o processo de produção e refinamento de requisitos, Pohl (1996) propôs um ambiente no qual o processo de desenvolvimento de software inclui o rastreamento de requisitos; Pinheiro (apud [Pinheiro,

1996a]) discutiu uma abordagem formal para a questão do rastreamento. No entanto, essas pesquisas não fornecem um processo bem definido para desenvolver um modelo de rastreamento usando suas próprias propostas de trabalho; não fornecem uma classificação das informações que desejam rastrear; não fornecem e/ou não aplicam seus meta-modelos para desenvolver modelos de rastreabilidade particulares. Somente Ramesh (2001) aborda modelos de referência para o rastreamento de requisitos. Dessa forma, o artigo apresenta uma proposta para solucionar os problemas apresentados acima, a qual corresponde à seguinte estratégia:

- Classificar as informações a serem rastreadas.
- Propor um meta-modelo.
- Definir um modelo intermediário de requisito que possua muitos dos artefatos que, em geral, são encontrados nos modelos de rastreamento.
- Definir um processo que reúna e aplique as três estratégias apresentadas anteriormente.

Os autores fizeram um levantamento sobre gerenciamento de requisitos, o qual foi sintetizado na Tabela 5.

Tabela 5 – Comparação das pesquisas sobre rastreamento gerenciamento de requisitos [Toranzo et al., 2002]

	Crítérios de comparação	Toranzo	Ramesh	Gotel e Filkenstein	Jarke	Pohl	Pinheiro
1	Apresenta um meta-modelo para o rastreamento	Sim	Sim	Não	Sim	Sim	Sim
2	Classifica as informações	Sim	Não	Não	Não	Não	Não
3	Propõe tipos de relacionamentos	Sim	Sim	Não	Sim	Sim	Não
4	Propõe modelo intermediário de rastreamento	Sim	Não	Não	Não	Não	Não
5	Fornecer um processo para construir um modelo de rastreamento	Sim	Não	Não	Não	Não	Não
6	Tratamento explícito de aspectos externos de uma organização	Sim	Não	Não	Não	Não	Não
7	Tratamento explícito com aspectos organizacionais	Sim	Sim	Não	Não	Não	Não
8	Tratamento explícito da gerência de projeto	Sim	Não	Não	Não	Não	Não

A seguir, comentam-se, com mais detalhes, essas estratégias propostas pelos autores.

1 - Classificar as informações a serem rastreadas:

Toranzo et al. divide as informações que serão rastreadas em quatro níveis:

- **Ambiental:** representa as informações do ambiente para o qual o software está sendo desenvolvido como, por exemplo, leis que regem o ambiente, fatores econômicos e políticos, além de padrões que devem ser seguidos.
- **Organizacional:** representa conceitos e regras da organização que desenvolve o sistema.
- **Gerencial:** representa o relacionamento entre os requisitos do sistema e as definições das tarefas. Segundo Wiegers (1999), a gerência de projetos de software deveria estar mais relacionada com os requisitos de software.

- **Desenvolvimento:** representa os elementos e os artefatos produzidos nas diferentes atividades durante o processo de desenvolvimento, tais como: documento de requisitos, diagramas, programas entre outros. É justamente nessa área que se concentra o maior número de pesquisas sobre rastreamento.

2 - Propor um meta-modelo

O objetivo é propor um modelo instanciável que comporte as relações entre as informações rastreáveis. No modelo apresentado, busca-se uma forma de permitir a modelagem do relacionamento entre tudo que irá ser rastreado, definindo níveis de detalhamento e tipo de relação usado para rastrear, como: generalização, agregação, alocado, satisfação, recurso, responsabilidade e representação.

3- Modelo intermediário para rastreamento de requisitos

Esse modelo intermediário trata-se de uma forma de incluir todos os elementos que devem ser rastreados. É originário de uma combinação de diversos fatores, como as boas práticas de gerenciamento de requisitos, estudos de casos, abstração, aplicação do meta-modelo definido anteriormente e dos níveis de informação que foram propostos pelo autor, bem como uma extensão da proposição de Ramesh (2001).

O autor salienta ainda que, possivelmente, o modelo intermediário proposto pode não abranger todos os casos, o que não inviabiliza seu uso e sua adaptação para necessidades de outros projetos e organizações.

4 - Definir um processo

Toranzo et al. (2002) definem diretrizes para que se obtenha um processo de desenvolvimento de software pautado em atividades de rastreamento de requisitos. É interessante notar que o autor usa o termo “classe” para definir dados que serão rastreados, pois o modelo por ele proposto, segue a notação do diagrama de classes da UML. As diretrizes são mostradas abaixo:

1. **Identificar as informações que podem afetar o sistema:** consiste em buscar as classes, no nível de informação ambiental, que podem afetar o sistema.

2. **Identificar objetivos, estratégias ou regras de negócio que serão rastreados:** consiste em montar uma tabela com todos os objetivos, estratégias ou regras de negócio.
3. **Incluir classes de informação da gerência de projeto no modelo de rastreamento:** consiste em criar as classes que mostrarão as tarefas do projeto. Essa etapa é muito importante, uma vez que busca reduzir a distância entre a gerência de requisitos e a gerência de projetos. Essa amarração entre ambos auxilia muito na obtenção da rastreabilidade bi-direcional, que já foi abordada anteriormente na Seção 2.3.2.
4. **Identificar subsistemas:** consiste em identificar os subsistemas que podem compor o sistema.
5. **Identificar requisitos:** consiste em incluir uma classe: requisito do modelo intermediário. Os requisitos podem ser colocados em uma tabela, com identificação única para cada um deles.
6. **Identificar diagramas:** consiste em identificar os tipos de diagramas em que foram ou que serão representados os requisitos do sistema. Além desse levantamento, devem-se determinar quais os caminhos lógicos da relação entre os requisitos e os diagramas.
7. **Identificar programas:** consiste em identificar os programas em que foram implementados os requisitos do sistema e determinar o caminho entre eles e os requisitos levantados.
8. **Identificar teste:** consiste em mostrar o caminho entre trechos do documento de testes que estão relacionados com cada requisito levantado.
9. **Remover as classes de informação irrelevantes:** consiste em selecionar as informações que realmente devem ser rastreadas. Dessa forma, evita-se o trabalho com informações sem importância no contexto da rastreabilidade.
10. **Integrar as classes com o mesmo significado:** consiste em fazer o agrupamento das informações de mesmo sentido. Esse agrupamento pode facilitar a organização das informações e, conseqüentemente, favorecer a rastreabilidade.
11. **Integrar novas classes:** consiste em adequar o modelo para as necessidades dos que irão fazer a rastreabilidade. Caso o modelo intermediário necessite de classes que não foram previstas, a inserção de novas classes deve ocorrer nesse instante.
12. **Organizar as classes:** consiste em organizar, em níveis hierárquicos, todas as classes levantadas nas etapas anteriores.

- 13. Estabelecer relacionamentos:** consiste em descrever todos os relacionamentos entre as classes candidatas. Esse relacionamento deve representar as relações de dependência entre as classes e qual o tipo de relacionamento entre elas.
- 14. Recomendar atributos sobre as classes:** consiste em levantar atributos para as classes que foram apontadas nas etapas anteriores. Esses atributos podem ser úteis, pois ilustrarão características dos elementos relacionados.
- 15. Definir uma matriz para cada relacionamento do modelo:** consiste em dispor em matrizes os relacionamentos encontrados no modelo. Essas matrizes geradas deverão ser revistas e validadas.

Essa seqüência de passos foi proposta pelos autores com o intuito de incluir no processo de desenvolvimento de software diretrizes que levem ao gerenciamento de requisitos, mais precisamente, à rastreabilidade. Segundo os autores, o grande diferencial de sua proposta é a construção de um modelo para a rastreabilidade e da determinação do processo de estabelecimento dessa rastreabilidade. Esse processo foi explorado em um sistema desenvolvido para a biblioteca da Universidade Estadual do Oeste do Paraná (UNIOESTE).

❖ **A Framework for Requirements Traceability in UML-based Projects; Letelier, Patricio; 2002**

Esse trabalho apresenta um meta-modelo de referência para rastreabilidade de requisitos baseado nos elementos da UML. Valendo-se dos mecanismos de extensão da UML, o modelo proposto é definido como um caminho natural que pode ser adequado às necessidades dos projetos. Segundo o autor, os modos propostos para obter-se o rastreamento de requisitos são baseados em manter a relação entre textos ou partes de texto. Dessa forma, os requisitos são ligados formando um gráfico de rastreabilidade. As ferramentas CASE existentes dão suporte a esse tipo de rastreabilidade, permitindo exportação e importação dos dados de um módulo para outro. Outra possibilidade é o uso de relação entre as especificações com os diagramas ou modelos. No entanto, essa forma pode também não ser viável devido à complexidade dos sistemas. A proposta do trabalho é fazer uso da UML e suas possíveis extensões para estabelecer um framework que represente a especificação dos requisitos, do desenvolvimento e dos testes. O meta-modelo

proposto busca cobrir os quatro pontos considerados necessários por Ramesh e Jarke (2001) para permitir a rastreabilidade, sendo esses listados abaixo:

- Requisitos: deve considerar o levantamento e a alocação das necessidades do cliente em um repositório de dados.
- Relacionamento: deve registrar as dependências entre os elementos.
- Alocação dos requisitos em um modelo: deve dispor os elementos a serem rastreados, registrando os relacionamentos entre eles.
- Teste: deve possibilitar a validação dos requisitos e do modelo.

O trabalho propõe ainda uma integração do meta-modelo com o processo de desenvolvimento de software da Rational Software, o RUP (Rational Unified Process). As etapas para obter o rastreamento são descritas abaixo:

1. **Selecionar os tipos de artefatos que serão rastreados:** os artefatos devem ser selecionados e indicando-se qual o tipo de relacionamento que terão com as classes do meta-modelo proposto.
2. **Definir as relações de agregação entre os tipos de artefato:** as relações entre os artefatos devem mostrar quais são parte de outro.
3. **Estabelecer as ligações entre os tipos de rastreabilidade:** as ligações devem mostrar os relacionamentos entre os artefatos.
4. **Estabelecer rastreabilidade implícita:** as ligações são estabelecidas entre artefatos do mesmo nome.

O trabalho é concluído salientando-se que as pesquisas em gerenciamento de requisitos, abordagens sobre modelagem e desenvolvimento de software têm sido feitas em paralelo, sem grande integração dos envolvidos. A proposta do uso da UML é uma forma de integrar as pesquisas das duas áreas, uma vez que, segundo o autor, tendo a rastreabilidade definida na UML, a mesma pode ser mais facilmente mantida.

❖ Automating Requirements Traceability: Beyond the Record & Replay Paradigm; Egyed, Alexander, Grünbacher, Paul; 2002

Esse trabalho apresentou uma técnica para utilização de Rastreamento de Requisitos apoiado por uma ferramenta que automatizou a geração dos relacionamentos entre requisitos, bem como entre os requisitos e os artefatos gerados durante o processo de desenvolvimento de software.

A ferramenta de automatização é o analisador de rastros. A partir das dependências fornecidas pelo engenheiro de software, o analisador monta um grafo de dependências e analisa-o, com o intuito de diminuir ambigüidades e encontrar as dependências que não são explícitas.

O artigo propõe o uso de uma lista de requisitos e um diagrama de estados para a montagem de uma tabela de cenários de teste. Cada cenário de teste tem relacionado a ele um ou mais requisitos, bem como as classes (código) que são utilizadas.

Então, o analisador utiliza-se de diversas regras para determinar quais artefatos dependem de quais classes. Em alguns casos, o analisador consegue remover as ambigüidades de dependência, encontrando a dependência exata de um artefato com suas classes.

A ferramenta também auxilia o engenheiro de software a identificar o impacto de mudanças nos requisitos ou a adição de novos requisitos. Embora ainda não haja a base comum (o código ainda não foi implementado), é possível utilizar hipóteses, gerando a tabela de cenários de teste, permitindo analisar quais requisitos serão afetados pelas mudanças.

Sumarizando o trabalho realizado, o artigo apresentou os pontos do rastreamento de requisitos que são apoiados pelo analisador de rastros, por gerar o rastreamento de forma automatizada:

- esclarece origem de requisitos e fornece argumentos que justifiquem sua existência através do rastreamento;
- permite identificação de dependências de requisitos não funcionais com outros requisitos e artefatos;
- auxilia na identificação de conflitos entre requisitos. Embora o analisador não os identifique automaticamente, permite verificar suas dependências, bastando uma análise dessa dependência para concluir que existe um conflito;

- auxilia na verificação dos requisitos. É possível identificar quais porções de código implementam determinados requisitos, auxiliando na verificação desses;
- permite que a ausência de requisitos seja percebida, uma vez que pode existir código ou outro artefato que não tenha um requisito relacionado a ele;
- auxilia na determinação do impacto de mudanças bem como na adição de novos requisitos, apresentando requisitos e artefatos que serão afetados;
- auxilia na determinação de quão dependente um requisito é de outro. Isso pode ser feito avaliando-se a quantidade de classes que compõem a base comum de dois requisitos, por exemplo. Assim, é possível ter dependências de 100%, 30%, etc. Essa proposta de visualização do percentual de dependência foi utilizada no ambiente COCAR. Nesse caso, entretanto, o valor foi obtido de forma diferente, conforme será mencionado na Seção 5.4;
- ajuda a balancear a granularidade dos requisitos, indicando quais deles são muito genéricos (afetam muitas classes) para que possam ser reavaliados.

A conclusão do artigo ressaltou, como sua contribuição principal, a redução no esforço de geração do rastreamento, uma vez que o analisador deriva as dependências existentes a partir de um conjunto pequeno de dependências reais ou hipotéticas. Além disso, a geração de dependências não-óbvias entre requisitos permite uma melhor avaliação de situações que não são comuns dentro do escopo do sistema.

❖ **An Approach Based on the Concept of Use Case to Produce Dynamic Object-Oriented Specifications; Dano, Bénédicte; Briand, Henri; Barbier, Franck; 1997**

Dano et al. (1997) propôs uma maneira de relacionar o processo de gerenciamento de requisitos com a abordagem de métodos orientados a objetos. Segundo ele, o Diagrama de Casos de Uso da UML facilita a comunicação entre o analista e os *stakeholders*, embora apresente duas desvantagens:

- 1) Fornecimento de uma visão isolada de cada uma das funcionalidades do sistema.

2) Inexistência de um padrão para a definição de casos de uso, o que pode gerar uma série de inconsistências e ambigüidades na definição desses.

Com o intuito de obter a relação entre os casos de uso e, conseqüentemente, entre os requisitos do software, são propostos e definidos *links* temporais entre os casos de uso. Encontraram-se sete tipos desses *links* entre os casos de uso, todos remetendo idéia de tempo entre eles:

- antes: relação entre dois casos de uso quando um começa antes do outro;
- encontram: situação em que um Caso de Uso inicia-se quando o outro acaba;
- começam: ambos começam no mesmo instante de tempo;
- acabam: ambos acabam no mesmo instante;
- iguais: quando os casos de uso ocorrem simultaneamente, ou seja, começam e terminam no mesmo instante;
- durante: um Caso de Uso começa depois de outro e termina antes;
- sobrepostos: quando existe uma sobreposição do tempo de vida dos casos de uso.

Embora tenha sido definida a relação de tempo entre os Casos de Uso, não foi definida a forma de como obter essas relações entre os Casos de Uso e como modelar essa dependência entre eles. Zisman e Spanoudakis (2004) também afirmam que faltam pesquisas e estudos nessa área de dependência entre requisitos. Cordeiro (2002) enfatiza a necessidade de formas automáticas de encontrar o relacionamento entre os requisitos. Dano et al. (1997) apontam ainda a necessidade da existência de formalismo e padronização para a especificação de Casos de Uso. Nesse sentido, a implementação da técnica TUCCA no ambiente COCAR visa à resolução desse problema, padronizando os resultados dos Casos de Uso gerados.

❖ **Indicadores para a Gerência de Requisitos; Hazan, Claudia; Julio Cesar Sampaio do Prado Leite; 2003.**

Como visto nos tópicos anteriores, a mudança nos requisitos é algo inevitável. Além da necessidade de prever o impacto dessas mudanças, é muito interessante poder quantificá-las. Na engenharia de software, a necessidade por medições tem evoluído e, se no passado muitas

empresas tratavam as medições como um trabalho adicional e sem muita valia, hoje as organizações maduras encaram a medição como uma necessidade e ferramenta estratégica para auxiliar na tomada de decisões [Lam, 1999]. No trabalho de Hazan e Leite (2003) aplicado em projetos do Serviço Federal de Processamento de Dados (SERPRO), são propostos dois grupos de indicadores sobre a gerência de requisitos:

- **Indicador de Estabilidade:** analisa o grau de mudanças para a *baseline* dos requisitos do software e o impacto dessas mudanças no esforço de desenvolvimento. Como visto, a rastreabilidade é necessária para controlar as mudanças nos requisitos, avaliar o impacto das mudanças e os riscos. Quantificar a estabilidade torna-se uma medida importantíssima para auxiliar no gerenciamento de requisitos.
- **Indicador de Rastreabilidade:** mede a aderência dos artefatos utilizados no processo de desenvolvimento de software (especificações, modelagem e código) e seus requisitos associados nos vários níveis de detalhamento. A medida, nesse caso, faz referência ao percentual de requisitos que pode ser rastreado nas especificações alocadas para o nível imediatamente abaixo, descendo até o nível de código.

No artigo, um indicador é apresentado como um dado numérico, expresso em uma unidade de medida, que é trazido periodicamente para a análise dos gestores dos processos. Sendo assim, os indicadores definidos acima devem ser levados aos gerentes de projeto para que esses avaliem a qualidade do processo de desenvolvimento de software. Se o indicador de estabilidade sinalizar que os requisitos estão mudando muito, é sinal de que o levantamento de requisitos não foi bem feito. Dessa forma, pode-se avaliar o porquê do levantamento não ter sido bem feito e corrigir esses defeitos para os próximos projetos. De fato, tomar como base uma decisão pautada em indicadores é a melhor maneira de acertar o caminho que deve ser tomado.

É interessante notar ainda que no trabalho de Hazan e Leite (2003) consideraram-se apenas os requisitos funcionais. Essa decisão foi tomada porque, segundo Leite (apud [Leite, 1994]), os requisitos funcionais podem ser representados em forma de uma sentença. Dessa forma, uma alteração, inclusão ou exclusão de uma dessas sentenças é caracterizada como uma mudança nos requisitos.

Os autores utilizaram o método GQM (*Goal Question Metric*) [Basili, 1992 (apud [Hazan & Leite (2003))] para definir as métricas apropriadas. Esse método inicia-se com a seleção de objetivos da medição. São declarados os objetivos, sendo esses quantificáveis e mensuráveis. Feito isso, para cada objetivo, encontraram-se as perguntas que precisam ser respondidas para determinar se o objetivo está sendo alcançado ou não. Por fim, identificam-se métricas que ajudam a responder cada pergunta. Como visto anteriormente, o indicador de estabilidade busca quantificar quanto os requisitos mudaram no decorrer do desenvolvimento. O objetivo definido para o método GQM, foi o de controlar as mudanças nos requisitos e as perguntas visavam à determinação da estabilidade dos requisitos. Além disso, foram definidas as formas de cálculo desses indicadores. Vale ressaltar que também foi proposto o uso de Pontos por Função (PFs) para gerar outro indicador.

Objetivo: Controlar as Mudanças nos Requisitos

- **Pergunta:** Qual o percentual de novos requisitos no período?

Indicadores:

- Número de requisitos novos / Número de requisitos alocados.
- Requisitos novos (PFs) / Requisitos alocados(PFs).

- **Pergunta:** Qual o percentual de requisitos alterados no período?

Indicadores:

- Número de requisitos alterados / Número de requisitos alocados.
- Requisitos alterados (PFs) / Requisitos alocados(PFs).

- **Pergunta:** Qual o percentual de requisitos excluídos no período?

Indicadores:

- Número de requisitos excluídos / Número de requisitos alocados.
- Requisitos excluídos (PFs) / Requisitos alocados(PFs).

Ainda explorando os indicadores de estabilidade, os autores propuseram que essa medição fosse incorporada no processo de desenvolvimento dos que adotarem a coleta de métricas como proposto, sugerindo também o uso de um *template*, manual ou automático, que auxilie nessa

coleta. Tendo coletado os dados, é possível gerar gráficos da evolução da estabilidade do sistema no decorrer do tempo.

Outro indicador proposto no trabalho é o indicador de rastreabilidade, relacionado com a capacidade de rastrear o requisito até sua origem. Assim sendo, esse indicador está relacionado com a possibilidade de acompanhar a vida de um requisito de maneira bi-direcional, ou seja, desde o nível mais alto (origem do requisito com o *stakeholder*) até os níveis mais baixos (até chegar à implementação). Esse indicador foi definido da mesma forma, utilizando-se GQM:

Objetivo: Controlar a aderência dos produtos de software (especificações, modelagem e código) com seus requisitos, deles nos vários níveis de especificação do produto.

- **Pergunta:** Qual o percentual de requisitos rastreáveis até sua origem? (pré-rastreabilidade)

Indicadores:

- Número de requisitos rastreáveis para a origem.
- Número de requisitos rastreáveis para a sua origem / Número total de requisitos alocados.

- **Pergunta:** Qual o percentual de requisitos rastreáveis para o próximo nível mais baixo? (pós-rastreabilidade)

Indicadores:

- Número de requisitos rastreáveis para a próxima atividade.
- Número de requisitos rastreáveis a próxima atividade / Número total de requisitos alocados.

- **Pergunta:** Qual o impacto operacional dos requisitos modificados? (efeitos sobre componentes do software)

Indicadores:

- Número de requisitos impactados / Número de requisitos alocados.
- Requisitos impactados (PFs)/ Requisitos alocados(PFs).

Complementando essas medições, o trabalho propôs e mostrou o uso da matriz de rastreabilidade com o intuito de auxiliar a tomada de decisão sobre as mudanças. A conclusão do trabalho mostrou a necessidade e a tendência cada vez maior de as empresas de software introduzirem métricas no processo de desenvolvimento de software, com o objetivo de aprender com as próprias experiências e poder repetir sucessos anteriores. Nesse sentido, o uso de indicadores não só ajuda o controle dessas organizações como também as auxilia a aprender sobre elas próprias. Outra conclusão interessante é que, embora a evolução esteja sempre presente, é comum que alguns requisitos sejam mais estáveis que outros e que os requisitos voláteis sejam acompanhados com bastante cuidado pela equipe.

Os indicadores propostos nesse artigo foram implementados no ambiente COCAR conforme será apresentado e discutido no Capítulo 5, na Seção 5.4.

❖ **TUCCA: Técnicas de Leitura para Construção de Modelos de Casos de Uso e Análise do Documento de Requisitos; Belgamo, Anderson; 2004**

No seu trabalho de mestrado, Belgamo definiu uma técnica para leitura do Documento de Requisitos, com o intuito de fornecer diretrizes para a construção dos Modelos de Casos de Uso e, simultaneamente a essa tarefa, encontrar os defeitos no Documento de Requisitos. Como um dos primeiros artefatos elaborados durante o processo de desenvolvimento de software é o Documento de Requisitos, encontrar defeitos nesse documento é de grande valia para o processo, uma vez que o custo de correção quando o problema é encontrado nessa fase é muito menor do que quando encontrado nas demais etapas do processo de desenvolvimento.

A importância de obter-se o Modelo de Casos de Uso de maneira padronizada é muito grande. Como a UML não estabelece diretrizes para a criação da especificação do Modelo de Casos de Uso, a atividade de construção desses modelos torna-se uma tarefa subjetiva, sendo de difícil padronização e totalmente dependente da experiência do projetista.

O resultado do trabalho foi a definição da técnica TUCCA – *Guidelines for Use Case Construction and Requirements Analysis*, cujo objetivo principal é a construção de Modelos de

Casos de Uso de forma sistemática e procedimental, além de fazer a busca de defeitos no Documento de Requisitos. Sobre o Documento de Requisitos, devem ser feitas duas leituras, segundo as técnicas: AGRT (*Actor-Goal Reading Technique*) e UCRT (*Use Case Reading Technique*). Elas devem ser utilizadas obrigatoriamente nessa seqüência, pois os resultados da AGRT são entradas para a aplicação da UCRT.

A técnica AGRT usa como entrada o Documento de Requisitos e visa à identificação dos candidatos a atores e seus respectivos objetivos de utilização do sistema, fazendo isso com base nos substantivos e verbos, respectivamente, que devem ter sido previamente assinalados no Documento de Requisitos. A partir dessa leitura, gera-se o Formulário Ator x Objetivo (FAO), que registra, além dos atores e seus objetivos, os requisitos funcionais em que esses foram identificados no Documento de Requisitos. Durante o processo de criação do FAO, o inspetor é guiado pela técnica AGRT a encontrar defeitos no Documento de Requisitos que inviabilizem a construção desse formulário e, conseqüentemente, o entendimento do sistema e a modelagem dos Casos de Uso. Ao final da aplicação da AGRT, têm-se como artefatos produzidos o Formulário Ator X Objetivo e um Relatório de Defeitos encontrados.

A técnica UCRT usa como entrada os artefatos produzidos pela técnica AGRT e visa à criação do Modelo de Casos de Uso propriamente dito, ou seja, determinar quais Casos de Uso devem existir e os relacionamentos existentes entre eles, além da geração da especificação de cada Caso de Uso identificado. Essas informações devem ser colocadas em um formulário denominado FCUP (Formulário de Casos de Uso Preliminares). Da mesma forma que na AGRT, durante a aplicação da UCRT, também é possível a identificação de defeitos no Documento de Requisitos. Ao final da aplicação da UCRT, tem-se como resultado o Formulário de Casos de Uso Preliminares, as especificações de cada Caso de Uso identificado preenchidas de acordo com o *template* adotado e o Relatório de Defeitos acrescido dos defeitos encontrados pela aplicação dessa segunda leitura.

Alguns estudos experimentais já foram conduzidos para avaliar a contribuição da técnica e os resultados mostram que os Modelos de Casos de Uso gerados são mais padronizados, isto é, quando elaborados por diferentes pessoas, com base no mesmo Documento de Requisitos, os

modelos são bastante parecidos, o que mostra que a experiência e a subjetividade do projetista não têm muita interferência nesse processo [Belgamo & Fabbri, 2005].

❖ **Diretrizes para elaboração de Documento de Requisitos com ênfase nos Requisitos Funcionais; Kawai, Karina K.; 2005**

Nesse trabalho de mestrado, Kawai explorou a criação do Documento de Requisitos com foco na especificação dos requisitos funcionais, abordando:

- elementos que devem estar presentes no Documento de Requisitos;
- recomendações de escrita para evitar defeitos durante a confecção do Documento de Requisitos;
- *checklist* pré-inspeção, no intuito de definir se o Documento de Requisitos já se encontra em um estado em que uma inspeção seria produtiva.

Como resultado do trabalho foi apresentado um conjunto de elementos que devem ser levantados e documentados para cada um dos Requisitos Funcionais, sendo eles apresentados abaixo:

- Identificação do Requisito: representa univocamente cada requisito do software.
- Descrição: explicação sucinta da funcionalidade, descrevendo “o que fazer” e não “como fazer”.
- Agente Fornecedor/Receptor: indica qualquer entidade que interage com o sistema de forma direta.
- Agente Executor: indica qualquer entidade que interage com o sistema de forma indireta.
- Entrada: representa os dados que estão envolvidos na funcionalidade.
- Processamento: contém, de forma detalhada, o fluxo de atividades ou ações que o sistema deve realizar para alcançar o que foi escrito no item “Descrição”.
- Condição/Restrição: indica a restrição ou a condição necessária para que uma funcionalidade seja realizada.
- Saída: é a resposta do sistema à determinada funcionalidade.

Quanto às recomendações para a escrita, o trabalho determina a melhor forma de registrar as funcionalidades no Documento de Requisitos, abrangendo desde redundâncias e ambigüidades até termos consistentes que para a descrição dos requisitos.

Foi aplicado um estudo de caso para analisar a variação do número de defeitos em documentos de requisitos quando são aplicadas as diretrizes criadas. Os documentos que seguiram o padrão e que foram submetidos à TUCCA para a identificação de defeitos apresentaram uma menor quantidade de defeitos que os documentos que não seguiram as diretrizes desse trabalho.

De todos os aspectos abordados no trabalho de Kawai, no ambiente COCAR contempla-se apenas o conjunto de elementos que representam os requisitos funcionais. Dessa forma, na seqüência deste trabalho quando se fizer referência às diretrizes de Kawai, está-se fazendo referência somente aos elementos que devem representar os requisitos.

- Considerações Finais

Nessa seção foram apresentados alguns trabalhos relacionados à etapa de gerenciamento de requisitos e rastreabilidade que tiveram influência mais direta na implementação do ambiente COCAR no que diz respeito aos aspectos de rastreabilidade. Um desses trabalhos foi o de Alexander (2003), do qual os pontos mais relevantes citados foram o desenvolvimento de uma ferramenta semi-automática para rastrear Casos de Uso. Nesse trabalho são explorados ainda os tipos de *link* de rastreabilidade e a identificação pela lógica *fuzzy* do relacionamento entre Casos de Uso. Assim como no trabalho de Munson et al (2005), Alexander explora o controle de versão desses *links*, sendo que essa tarefa é considerada de extrema importância para o sucesso da difícil atividade de mapeamento da rastreabilidade. No trabalho da equipe de Munson, além da ênfase para o gerenciamento de versão, é salientado que a rastreabilidade foi obtida em partes do código fonte ou de arquivos XML. Esse artifício de dividir o artefato em partes também foi adotado por Kawai (2005), que propôs separar os requisitos funcionais do sistema em partes, de acordo com um *template* por ela definido. No ambiente que foi desenvolvido neste trabalho de mestrado e que será apresentado no próximo capítulo, a rastreabilidade também foi mapeada nessas “partes” que descrevem os requisitos, seguindo o padrão de Kawai (2005). De acordo com Munson et al

(2005), quando a informação do artefato está dividida em partes, a rastreabilidade fica mais rica e interessante.

No trabalho da equipe de Marcus et al (2005) é discutida a visualização dos *links* de rastreabilidade, levantando-se as características dos artefatos que são importantes para o rastreamento, bem como as características dos *links* que denotam a rastreabilidade, apresentando, inclusive, uma ferramenta para dar suporte a essa tarefa. No trabalho de Dano et al. (1997) também são abordados os *links*. Nesse caso, os *links* entre os Casos de Uso, sendo propostos relacionamentos temporais entre eles, não se definindo, entretanto, a forma de determiná-los.

O trabalho de Toranzo et al. (2002) descreve os passos que devem ser seguidos para se obter a rastreabilidade, passando pelo levantamento das informações que devem ser rastreadas, pela proposta de um meta-modelo e modelo intermediário, até culminar com a definição do processo que abranja a rastreabilidade. No trabalho de Letelier (2002), foi apresentada uma proposta de *framework* de rastreamento de requisitos baseada em UML. Tanto um como outro enfatizam a importância de incorporar o rastreamento no processo de desenvolvimento de software. Nesses dois trabalhos, apesar de serem apresentados aspectos para a rastreabilidade, não é proposta nenhuma ferramenta para viabilizar esse rastreamento. Já o trabalho de Egyed e Grübacher (2002) propõe uma ferramenta de apoio para buscar o relacionamento entre os requisitos de maneira automática, embora não aborde o rastreamento de requisitos com os demais artefatos produzidos no processo de desenvolvimento de software.

Outro tópico abordado nessa revisão diz respeito às métricas. Hazan e Leite (2003) propuseram uma maneira de quantificar as alterações nos requisitos e o nível de rastreamento obtido. Segundo o Standish Group (2004a), apesar de existirem diversas métricas que ajudem a determinar o *status* de um projeto, uma maneira efetiva de dimensioná-lo é medir a volatilidade dos requisitos. Através dessa medida, pode-se ter o indicativo da estabilidade dos requisitos. A importância dessa métrica está associada ao fato de que, quanto antes forem detectadas as alterações nos requisitos, menor será o impacto para incorporar a mudança no software que está sendo desenvolvido. Zisman e Spanoudakis (2004) afirmam que é extremamente importante uma medida que quantifique o nível de aderência entre os artefatos produzidos durante o processo de desenvolvimento do software. Ele considera esse ponto em aberto uma vez que não existe um

mecanismo que garanta a completeza da rastreabilidade. Aponta tal dificuldade como um dos fatores críticos, uma vez que não é possível melhorar algo que não é medido, pois fica-se sem parâmetros para afirmar se a rastreabilidade foi alcançada ou não.

Foi apresentado ainda o recente trabalho de Davi e Zowghi (2006), que comentam que, apesar das boas práticas de engenharia de requisitos não garantirem a qualidade do software, o não uso dessas práticas também não implicam, necessariamente, em insucesso nos projetos de software. No entanto, essas boas práticas devem ser utilizadas como caminho para um processo de desenvolvimento de software maduro e com maior probabilidade de sucesso. No artigo de Salem (2006) a rastreabilidade de requisitos também foi apresentada como um caminho para a melhoria da qualidade de software e no artigo de Cleland-Huang et al. (2006) a adesão à rastreabilidade de requisitos visa à maximização do retorno sobre o investimento, devendo ser feita de acordo com a complexidade e a natureza dos projetos.

Por último, foi comentado o trabalho de Belgamo (2004) que propôs uma técnica para a construção de Modelos de Casos de Uso de maneira sistemática e padronizada, e que ajuda na detecção de defeitos no Documento de Requisitos. O ambiente COCAR, que será apresentado no próximo capítulo, está fundamentado nessa técnica.

Os trabalhos apresentados neste capítulo foram de grande contribuição para a fundamentação teórica na construção do ambiente COCAR e orientaram o desenvolvimento tanto das funcionalidades de cadastro de requisitos [Kawai, 2005] e de geração do Modelo de Casos de Uso [Belgamo, 2004], como das funcionalidades específicas relacionadas à rastreabilidade [Hazan & Leite, 2003], [Salem,2006], [Alexander, 2003], [Munson & Nguyen, 2005].

Capítulo 5

O ambiente COCAR

5.1 – Considerações Iniciais

Como visto nos capítulos anteriores, diversos autores falam da necessidade e da importância de ferramentas que forneçam apoio às atividades de desenvolvimento de software ([Munson & Nguyen, 2005], [Marcus et al, 2005], [Egyed & Grübacher, 2002], [Sommerville, 20003] e [Alexander, 2005]). Tais ferramentas são denominadas ferramentas CASE e o sucesso do uso está diretamente ligado à capacidade que têm de abrangerem o processo de desenvolvimento de software de forma integrada. De acordo com Munson e Nguyen (2005), quanto maior o número de etapas do processo de desenvolvimento de software que a mesma ferramenta abranger, melhor o resultado que ela poderá proporcionar, evitando erros de integração e favorecendo a rastreabilidade dos artefatos gerados, aumentando a qualidade no processo.

Assim, o ambiente COCAR foi idealizado com objetivo de oferecer suporte a algumas atividades do desenvolvimento de software, considerando como base fundamental os Modelos de Casos de Uso. Visando à construção desse ambiente, alguns trabalhos de mestrado estão sendo conduzidos com objetivo de avaliar o estado da arte e outras propostas existentes na literatura, de forma a estabelecer estratégias e soluções que possam auxiliar e melhorar a qualidade no processo de desenvolvimento de software.

Sendo um ambiente baseado em Modelos de Casos de Uso, o suporte à elaboração dos mesmos foi a primeira funcionalidade a ser implementada, para que outras funcionalidades pudessem ser providas pelo ambiente, inclusive a rastreabilidade entre os requisitos e os Modelos de Casos de Uso, que foi o principal objetivo deste trabalho. A construção desses modelos foi totalmente baseada na técnica TUCCA [Belgamo, 2004] e a definição dos requisitos funcionais com base nos quais a TUCCA é aplicada para gerar os modelos, foi baseada nas diretrizes propostas por Kawai (2005). Esses dois trabalhos foram comentados no Capítulo 4 e a implementação dos

mesmos no ambiente COCAR foi realizada no âmbito deste trabalho, em conjunto com o mestrado de Martins (2007), uma vez que essas funcionalidades eram necessárias para ambos os trabalhos. Dessa forma, os itens 5.2 e 5.3, foram escritos em conjunto com o trabalho de Martins (2007) e por isso apresentam textos equivalentes.

Assim, o objetivo deste capítulo é apresentar os aspectos funcionais e de implementação do ambiente COCAR. Na Seção 5.2 são apresentadas as principais funcionalidades. Os aspectos de implementação são abordados na Seção 5.3. Em seguida, na Seção 5.4, apresentam-se os requisitos funcionais do ambiente COCAR no que diz respeito aos aspectos de rastreabilidade e, por fim, a Seção 5.5 contém as considerações finais.

5.2 – Funcionalidades do ambiente COCAR

O ambiente COCAR apresentado neste trabalho corresponde à sua primeira versão, que contém as seguintes funcionalidades:

- i) Inserção dos Requisitos de um sistema.
- ii) Elaboração de Modelos de Casos de Uso.
- iii) Geração Automática de Pontos de Casos de Uso.
- iv) Suporte ao Gerenciamento de Requisitos.
- v) Geração de Casos de Teste com base em Casos de Uso.

Cada uma dessas cinco funcionalidades, comentadas brevemente a seguir, corresponde a um trabalho de mestrado.

i) Inserção dos Requisitos de um sistema:

Esse módulo do COCAR corresponde ao trabalho de mestrado de Kawai (2005), intitulado “Diretrizes para elaboração de Documento de Requisitos com ênfase nos Requisitos Funcionais” que, como apresentado no Capítulo 4, propôs diretrizes para a construção do Documento de

Requisitos, facilitando a identificação das entidades envolvidas e seus papéis, informações de entrada e saída para o processamento associado à funcionalidade, descrição do processamento da funcionalidade em si, além de restrições e condições de execução de cada funcionalidade. No caso deste trabalho, o ambiente COCAR dá suporte à inserção dos requisitos, de acordo com as diretrizes propostas. Essa funcionalidade permite que as informações de um documento de requisitos sejam inseridas no ambiente para que, a partir delas, o Modelo de Casos de Uso possa ser gerado. Em vista das necessidades específicas da rastreabilidade de requisitos, dois outros campos de informação são tratados no ambiente, além daqueles necessários para contemplar as diretrizes de Kawai (2005), conforme será apresentado na Seção 5.4.4. O trabalho de Salem (2005) também salienta a importância de armazenar os requisitos em um banco de dados.

ii) Elaboração de Modelos de Casos de Uso:

Esse módulo do COCAR corresponde ao trabalho de mestrado de Belgamo (2004) comentado no Capítulo 4. O trabalho de Belgamo propôs uma técnica de leitura cujo objetivo principal é auxiliar na elaboração de Modelos de Casos de Uso fazendo com que essa atividade torne-se mais sistematizada e independente da experiência do desenvolvedor, como também menos suscetível a erros entendimentos que gerem modelos não coerentes com o Documento de Requisitos. Ao mesmo tempo em que auxilia na atividade de construção, os passos da TUCCA propiciam a identificação de defeitos no Documento de Requisitos. Cada Caso de Uso contém a descrição de seus passos, informações de entrada e saída, condições para execução do curso normal e dos cursos alternativos, restrições de tempo e desempenho, relações de dependência com outros Casos de Uso, além dos relacionamentos de inclusão e extensão em que o Caso de Uso está envolvido. Ao final da aplicação da TUCCA, têm-se o Diagrama de Casos de Uso e as Especificações dos Casos de Uso, sendo que as informações para compor o diagrama estão contidas nas próprias especificações. Ou seja, o ambiente não gera o diagrama, graficamente, mas as especificações possuem toda informação para que se consiga construí-lo.

iii) Geração Automática de Pontos de Casos de Uso:

Esse módulo do COCAR corresponde ao trabalho de mestrado de Martins (2007), intitulado “Suporte à Geração de Pontos de Casos de Uso no ambiente COCAR”. Essa funcionalidade faz a geração dos Pontos de Casos de Uso com base nas especificações dos Casos de Uso elaboradas

com a aplicação da TUCCA. Seguindo o *template* proposto em [Belgamo, 2004a], o qual é usado para descrever as especificações, as informações necessárias para a geração dessa métrica são organizadas de maneira a facilitar a sua geração. O valor dos Pontos de Casos de Uso também pode ser convertido em Pontos por Função. Esses dados auxiliam na fase de planejamento do software, uma vez que podem ser usados para estimar tempo, custo e esforço de desenvolvimento. No futuro, pretende-se que tais estimativas sejam também geradas pela ferramenta.

iv) Suporte ao Gerenciamento de Requisitos

Esse módulo do COCAR corresponde a este trabalho de mestrado e possibilita o gerenciamento dos requisitos. Ele oferece suporte para a rastreabilidade tanto no âmbito do próprio Documento de Requisitos como entre esse documento e o Modelo de Casos de Uso. Considerando-se a natureza dinâmica dos requisitos, essa funcionalidade possibilita o acompanhamento das alterações, informação esta que poderá ser usada também para auxiliar a geração de Casos de Teste de regressão, os quais devem ser aplicados sempre que o sistema sofrer alterações.

Nesse módulo é gerada a Matriz de Rastreabilidade de Requisitos, a qual indica de forma automática qual a relação existente entre os requisitos funcionais do software. Essa matriz pode ser utilizada para a previsão do impacto gerado nos demais requisitos do software, quando ocorrem modificações em um requisito específico. Por meio desse relacionamento, o usuário pode averiguar a necessidade ou não de alterar os demais requisitos relacionados com aquele em que a mudança foi planejada ou executada.

Esse módulo do ambiente COCAR também calcula o Indicador de Estabilidade, proposto por Hazan e Leite (2003), o qual fornece o percentual de alterações, inclusões ou exclusões de requisitos no sistema, em qualquer período do ciclo de desenvolvimento do software. Além disso, também são oferecidas facilidades para consultas sobre os requisitos cadastrados.

v) Geração de Casos de Teste com base em Casos de Uso:

Esse módulo do COCAR corresponde ao à proposta de trabalho de mestrado - em andamento - de Gregolin (2006), intitulada “Uma Proposta de Ferramenta para a Geração de Casos de Teste a partir de Casos de Uso”, que consiste em gerar Casos de Teste a partir dos Casos de Uso. Como

esse trabalho ainda está em andamento, não se tem maiores detalhes sobre essa funcionalidade. Em linhas gerais, pode-se dizer que, a partir dos Casos de Uso definidos pela aplicação da TUCCA, serão gerados Casos de Teste que explorem cenários de uso do sistema, definidos a partir da elaboração de um grafo construído com base nas especificações dos Casos de Uso. Além disso, pretende-se que, no futuro, a ferramenta dê suporte à geração de Casos de Teste que possam ser aplicados para testar os Casos de Uso individualmente. Como o ambiente COCAR prevê o controle das alterações que ocorrem nos requisitos, este módulo deverá fornecer suporte à geração de Casos de Teste de regressão. Dessa forma, será atendida a característica extensível do ambiente COCAR, ampliando a rastreabilidade dos artefatos envolvidos no processo de desenvolvimento de software.

Em resumo, o propósito deste mestrado foi integrar os trabalhos de Kawai (2005) (funcionalidade i) e Belgamo (2004) (funcionalidade ii) permitindo o Gerenciamento e Rastreabilidade de Requisitos (funcionalidade iv). Dessa forma, foi possível gerar a rastreabilidade entre os requisitos funcionais e o modelo de Casos de Uso, além de acompanhar a evolução dos requisitos, permitindo a previsão de impacto de alteração, inclusão ou exclusão de requisitos, com base na criação da matriz de rastreabilidade. A proposta de Hazan & Leite (2003), baseada na geração de indicadores para o gerenciamento de requisitos, foi implementada no ambiente com o intuito de avaliar as mudanças ocorridas com os requisitos ao longo do processo de desenvolvimento de software.

5.3 - Aspectos de Implementação do ambiente COCAR

Nesta seção serão comentados os aspectos mais relevantes relacionados ao desenvolvimento do ambiente COCAR. Primeiramente, aborda-se o processo de desenvolvimento escolhido para o desenvolvimento do ambiente; em seguida, comenta-se sobre a linguagem de programação bem como os padrões de projeto presentes na literatura e utilizados na implementação do ambiente COCAR; e, finalmente, trata-se da implementação, abordando-se o modelo utilizado para a codificação do ambiente.

5.3.1 - Processo de Desenvolvimento

De acordo com Pressman (2006), um processo de desenvolvimento de software deve ser escolhido baseado em três fatores:

- na natureza do projeto e da aplicação;
- nos métodos e ferramentas a serem utilizados;
- nos controles e produtos que precisam ser entregues.

Quanto à natureza do projeto e da aplicação, o ambiente COCAR pode ser considerado um software relativamente complexo, uma vez que seu objetivo é fornecer suporte a várias atividades do desenvolvimento de software que, embora completamente diferentes, são baseadas no mesmo alvo, isto é, os Modelos de Casos de Uso. Conseqüentemente, a implementação do ambiente deveria ser iniciada com as funcionalidades relativas à automatização da TUCCA [Belgamo, 2004] e ao tratamento da entrada dos requisitos funcionais, de acordo com as diretrizes propostas por Kawai (2005).

No que diz respeito à TUCCA, embora a técnica estivesse totalmente bem especificada e com seus passos bem detalhados e definidos, transformá-la em uma ferramenta implica na definição de uma série de outros detalhes que não são necessários para sua aplicação manual, como por exemplo, detalhes de interface, entradas e saídas de cada passo, etc. Dessa forma, havia o domínio completo da aplicação da técnica manual e possuíam-se os requisitos gerais para a sua automatização, mas vários detalhes precisavam ser definidos.

Quanto aos métodos, o ambiente COCAR foi desenvolvido utilizando-se a própria abordagem que essa ferramenta implementa, ou seja, a especificação de requisitos foi feita por meio das diretrizes propostas por Kawai (2005), para a especificação de requisitos, a geração do Modelo de Casos de Uso foi realizada aplicando-se, manualmente, a TUCCA [Belgamo, 2004] e o cálculo da estimativa de tamanho foi realizada aplicando-se, manualmente, Pontos de Caso de Uso [Kerner, 1993]. Uma vez gerado o Modelo de Casos de Uso do ambiente COCAR por meio da TUCCA, para obter-se a representação gráfica do diagrama de Casos de Uso, utilizou-se a ferramenta Rational Rose [Rational, 2006].

Quanto aos controles e produtos que precisavam ser entregues, considerou-se como imprescindíveis: o documento de especificação de requisitos do ambiente definido de acordo com as diretrizes de Kawai (2005); o Modelo de Casos de Uso gerado pela técnica TUCCA, manualmente; e, as versões parciais do ambiente COCAR, implementando as funcionalidades especificadas por meio de ciclos de desenvolvimento.

Levando-se em consideração os critérios sugeridos por Pressman (2006) para a escolha do processo de desenvolvimento de software, decidiu-se adotar o ciclo de vida de prototipação descartável seguida de prototipação evolutiva para o desenvolvimento do ambiente COCAR, pois:

- como dito anteriormente, o ambiente COCAR, no início de seu desenvolvimento, caracterizava-se como sendo um projeto no qual se conheciam os requisitos gerais do sistema, mas não todos os seus detalhes. Para um projeto dessa natureza, o modelo de prototipagem funciona como um mecanismo para identificar esses requisitos, principalmente quando se tem a definição de um conjunto de objetivos gerais para o software, mas os requisitos detalhados ainda não foram identificados ou o desenvolvedor está inseguro em relação à forma com que a interação homem-máquina deve acontecer [Pressman, 2006];
- relativo aos métodos utilizados, segundo Sommerville (2003), o modelo de prototipagem descartável exige, ao final da fase de Engenharia de Requisitos, um protótipo descartável juntamente com um documento de especificação. Contudo não define nenhum *template* desse documento e também esse modelo não faz restrições à forma de modelagem e especificação dos requisitos do software, sendo permitido, portanto, o uso do documento de requisitos de acordo com as diretrizes propostas por Kawai (2005) e a aplicação da técnica TUCCA para a geração do Modelo de Casos de Uso;
- relativo às ferramentas utilizadas, a prototipação evolucionária não exige o uso de nenhuma ferramenta de apoio específica, enquanto que a prototipação descartável sugere o uso de alguma ferramenta que seja capaz de gerar interfaces gráficas rapidamente. No caso deste trabalho, para a geração do protótipo descartável, foi escolhida a linguagem HTML que, embora não seja uma ferramenta propriamente dita, é uma linguagem pela qual se desenvolve interfaces gráficas rapidamente e foi utilizada para gerar as telas do

protótipo descartável e também, posteriormente, para a implementação da interface gráfica do protótipo evolucionário;

- relativo aos produtos e controles a serem entregues, a prototipação evolucionária, como já mencionado, exige a entrega de um documento de requisitos o qual foi realizado na especificação do ambiente COCAR por meio das diretrizes especificadas por Kawai (2005). Outro controle exigido pela prototipação descartável seguida pela prototipação evolucionária são os *releases* de protótipos e, posteriormente, os *releases* de versões executáveis, exigências essas que atendem perfeitamente às necessidades de desenvolvimento do ambiente COCAR.

O Processo de prototipação descartável deu-se conforme enunciado por Sommerville (2003):

- 1) Desenvolvimento de um documento de requisitos, segundo as diretrizes propostas por Kawai (2005), contendo as principais funcionalidades a serem atendidas pelo ambiente COCAR, dentre elas a técnica TUCCA [Belgamo, 2004], a técnica Pontos de Casos de Uso e o Gerenciamento de Requisitos. Além disso, também foram consideradas características de ferramentas relacionadas ao contexto do ambiente COCAR e discutidas nos capítulos anteriores.
- 2) Projeto e construção de um protótipo, a partir do documento de requisitos.
- 3) Submissão do protótipo à avaliação do grupo envolvido no projeto (professores e alunos) que continha pessoas experientes principalmente no uso da técnica TUCCA.
- 4) Se o protótipo fosse avaliado como completo, seguia-se para o passo 7; caso contrário, continuava-se no passo 5.
- 5) Refinamento do documento de requisitos com base nos dados da avaliação do passo 3.
- 6) Retorno ao passo 2.
- 7) Entrega do protótipo final com a especificação.

Após algumas iterações do processo de prototipação e com uma versão final do documento de requisitos do ambiente COCAR (Apêndice I), aplicou-se, manualmente, a técnica TUCCA gerando-se o Modelo de Casos de Uso (Apêndice II).

5.3.2 - Projeto

Com objetivo de facilitar o acesso ao ambiente COCAR, foi decidido a sua implementação em ambiente *Web* Cliente Servidor. Com essa abordagem, abstraem-se as dificuldades referentes à instalação da ferramenta, versões de software terceiros como banco de dados e não preocupação com requisitos mínimos para a execução do sistema. Dessa forma, foi estabelecido que a ferramenta fosse desenvolvida como um servidor que pudesse ser acessada através de um *browser* em uma máquina remota via *Internet*, permitindo assim um fácil acesso de todos os envolvidos no projeto com a ferramenta.

Para a implementação da ferramenta em um ambiente *Web* Cliente Servidor foi escolhida a linguagem Java por vários motivos, a saber:

1. Vasta documentação sobre desenvolvimento *Web* disponível tanto no site do fabricante da linguagem bem como na *Internet* de maneira geral;
2. Compiladores e ferramentas de desenvolvimento disponibilizados sem ônus nenhum na *Internet*, como por exemplo, o ambiente de desenvolvimento Eclipse [Eclipse, 2006];
3. Suporte completo ao paradigma de orientação a objeto;
4. Amplamente utilizada no mercado e na academia por cinco milhões de desenvolvedores configurando-se como a maior comunidade de desenvolvedores de software [Sun, 2006a];
5. O Java EE 5 ganhou o prêmio de melhor plataforma de *Web Services* em janeiro de 2006 do “2005 SOA *Web Services Journal Readers' Choice Awards*” [Sun, 2006b];
6. Como o ambiente COCAR é uma ferramenta cujo crescimento já está previsto com a adição de outros recursos e funcionalidades, como por exemplo, a geração automática de Casos de Teste prevista na proposta de trabalho de Gregolin (2006), buscou-se utilizar padrões de projetos existentes na literatura que permitam a fácil manutenção do software buscando conferir ao produto final alta manutenibilidade.

Dessa forma, o projeto do ambiente COCAR foi realizado com padrões MVC (*Model-View-Controller*), de maneira a separar as camadas de regras de negócio, de apresentação e de controle, promovendo um menor acoplamento do software, permitindo que manutenções de uma determinada parte da ferramenta possam ser feitas sem interferir nas demais.

Para garantir a independência da camada *VIEW* das demais, foi utilizado o *framework* Struts [Apache, 2006] por já estar consagrado no mercado de desenvolvimento de *software Web*, pela facilidade em usá-lo e pela vasta documentação disponível na *Internet*.

Com o propósito de implementar a camada *MODEL* com baixo acoplamento em relação às demais, foi utilizado o padrão de acesso a banco de dados DAO (*Data Access Object*) e o *Transfer Object* como exemplificado nas Figuras 7, 8, 9 e 10, respectivamente.

Como opção para o Gerenciador de Banco de Dados, foi adotado o *MySQL*, que atende às necessidades da ferramenta, além de ser gratuito e amplamente utilizado, contando com ampla documentação. Caso haja necessidade, a migração para outro gerenciador de banco de dados é facilitada devido ao uso do padrão DAO.

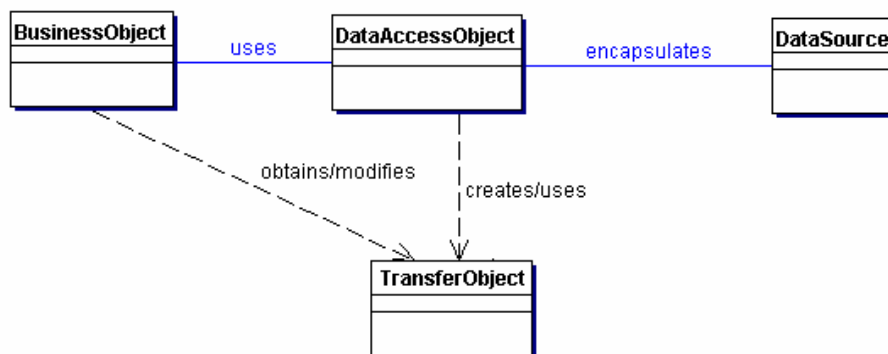


Figura 7 – Diagrama de Classe UML do Padrão DAO [Java, 2006]

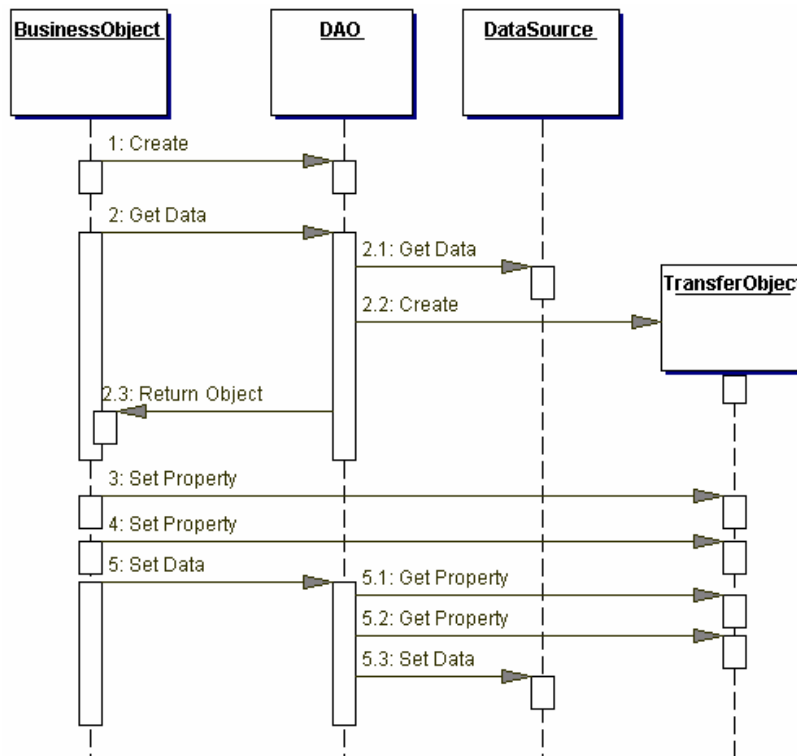


Figura 8 - Diagrama de Classe UML do Padrão DAO [Java, 2006]

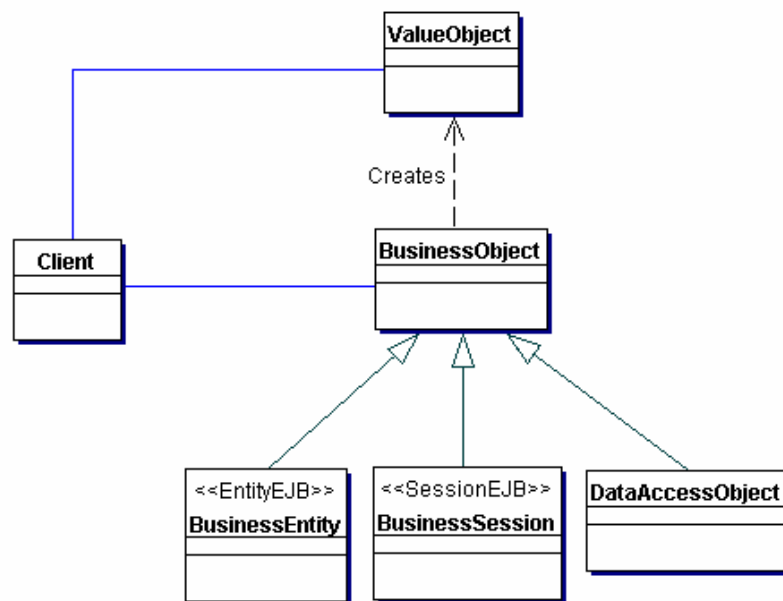


Figura 9 - Diagrama de Classes UML do Padrão *Transfer Object* [Java, 2006]

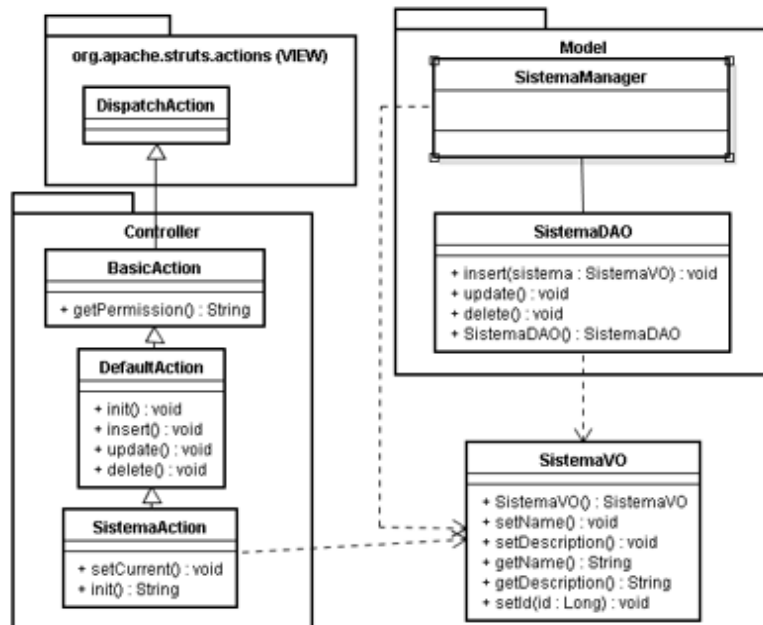


Figura 10 - Diagrama de Classes UML da funcionalidade Cadastrar Sistema

5.3.3 - Implementação

Conforme discutido na seção 6.2.1 e 6.2.2, para a implementação do ambiente COCAR foi usada a Prototipação Evolucionária, os Padrões de Projeto DAO e *Transfer Object* e a linguagem Java, sendo que o software foi implementado conforme o processo definido em Sommerville (2003) e mostrado na Figura 11 abaixo:

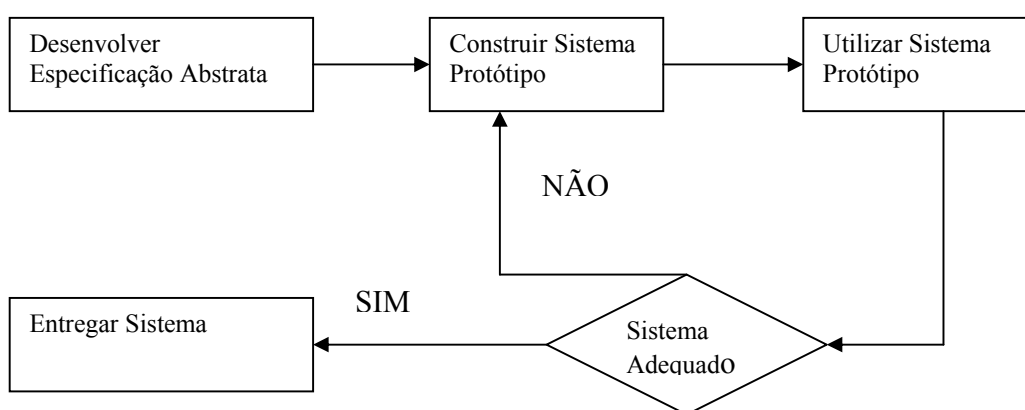


Figura 11 – Prototipação Evolucionária [Sommerville, 2003]

Contudo, houve uma pequena adaptação ao processo de Prototipação Evolucionária definido em Sommerville (2003) no que diz respeito à sua primeira fase, “Desenvolver Especificação Abstrata”, que foi substituída pelo uso da Prototipação Descartável. Dessa forma, a Prototipação Descartável gerou um documento de requisitos segundo as diretrizes propostas por Kawai (2005) que serviu como artefato de entrada para o processo de Prototipação Evolucionária.

5.4 - Requisitos Funcionais do ambiente COCAR – Gerenciamento de Requisitos e Aspectos de Rastreabilidade

Como visto anteriormente, diversos trabalhos enfatizam a necessidade de uma ferramenta de suporte para tornar viável a tarefa de gerenciamento e rastreabilidade de requisitos. Zisman e Spanoudakis (2004) acreditam que esse seja o único caminho para o sucesso nessa tarefa.

Da mesma forma, Munson e Nguyen (2005), Marcus et al (2005), Egyed e Grübacher (2002), Sommerville (2003) e Alexander (2005) enfatizam a necessidade de suporte de ferramentas para

a viabilização dessa atividade. Nesse sentido, serão apresentados na seqüência os requisitos funcionais do ambiente COCAR no escopo do Gerenciamento e da Rastreabilidade de Requisitos.

As funcionalidades específicas são:

- Geração da Matriz de Rastreabilidade de Requisitos.
- Criação dos Indicadores de Estabilidade.
- Rastreabilidade entre Documento de Requisitos e Modelo de Casos de Uso.
- Consultas sobre os Requisitos.

Na seqüência, cada uma dessas funcionalidades será detalhada. Exemplos serão apresentados no Capítulo 6.

5.4.1 – Geração da Matriz de Rastreabilidade de Requisitos

A matriz de rastreabilidade indica o relacionamento existente entre os requisitos do software. Todos os requisitos são listados na primeira linha e na primeira coluna. A intersecção linha/coluna irá representar a existência ou não de relacionamento entre eles. Diversos autores enfatizam a dificuldade de obter-se e manter esse tipo de matriz. O ambiente COCAR gera a matriz de rastreabilidade de requisitos de forma automática, assim como a manutenção dela. Diversos autores enfatizam a importância e a necessidade desse tipo de relacionamento para o processo de desenvolvimento de software ([Serpro, 2002], [Sommerville, 2003], [Salem, 2006], [Hazan & Leite, 2003], [Ramesh & Jarke, 2001]). É possível por meio dela a previsão do impacto de uma mudança ou inserção de um requisito no sistema.

Somerville (2003) propõe que a indicação da dependência entre os requisitos determine, além da existência ou não de relação entre eles, uma forma de registrar se essa relação é forte ou fraca, de forma subjetiva. A proposta deste trabalho vai além, indicando qual o percentual de dependência que existe entre os requisitos, baseando-se no percentual de dependência entre os dados dos requisitos. Isso é possível porque a forma de armazenamento dos requisitos no ambiente COCAR separa os dados de entrada de cada um desses requisitos de forma atômica, conforme descrito no

trabalho de Kawai (2005). A ferramenta oferece suporte à criação desses dados de entrada para o sistema, sendo eles selecionados ou não como parte de cada um dos requisitos funcionais do sistema.

5.4.2 – Criação do Indicador de Estabilidade

Outra funcionalidade implementada no ambiente COCAR é o Indicador de Estabilidade, proposto por Hazan e Leite (2003) e apresentado no Capítulo 4. A proposta dos autores é a criação de indicadores para a obtenção de métricas durante o processo de gerenciamento de requisitos. A necessidade de medidas durante esse processo também é enfatizada e defendida pelo SWEBOK (2004) e pelos trabalhos de Serpro (2002). Com o apoio das métricas é possível encontrar estratégias mais eficientes para a gerência de requisitos, identificando, por exemplo, as maiores fontes de mudança, relacionando o grande número de requisitos alterados, inseridos ou excluídos com a ocorrência de erros ou atrasos no desenvolvimento do software.

O ambiente COCAR mantém o controle dos requisitos que são inseridos, alterados ou excluídos em cada sistema. A partir desse histórico mantido, é possível a determinação dos indicadores propostos, a saber:

- Percentual de novos requisitos no período, obtido pela divisão do número de requisitos novos pelo número de requisitos alocados.
- Percentual de requisitos alterados no período, obtido pela divisão do número de requisitos alterados pelo número de requisitos alocados.
- Percentual de requisitos excluídos no período, obtido pela divisão do número de requisitos excluídos pelo número de requisitos alocados.

Para a determinação do período a ser analisado, é solicitado ao usuário que determine qual o período em que a análise deve ser feita. Isso permite ainda determinar quais as fases do processo de desenvolvimento de software em que houve mais solicitações de exclusão, alteração ou inserção de requisitos no sistema.

5.4.3 – Rastreabilidade entre Documento de Requisitos e Modelo de Casos de Uso

A rastreabilidade entre os artefatos criados ao longo do processo de desenvolvimento de software é fonte de estudo de diversos autores. Alexander (2003) apresentou uma proposta para a rastreabilidade entre os Requisitos e o Modelo de Casos de Uso, baseado na ferramenta comercial DOORS. Adrian et al.(2005) exploraram a visualização dos *links* de rastreabilidade entre artefatos, definindo inclusive atributos para os mesmos, no intuito de agregar conhecimento nessa ligação. Munson e Nguyen (2005) consideraram importante que a rastreabilidade abrangesse um número cada vez maior de artefatos, sendo que esse processo deveria ser apoiado por ferramentas que, segundo os autores, reduziriam o esforço requerido para manter a rastreabilidade. Kelleher (2005) propôs um *framework* para representar a rastreabilidade, englobando também a dependência ente os artefatos. Wiegers (1999b) destacou que rastrear requisitos com outros componentes do sistema iria ajudar a garantir que os artefatos estivessem consistentes ao longo do processo de desenvolvimento de software. Outros pesquisadores abordaram o mesmo tema ([Zisman & Spanoudakis, 2004], [Sommerville, 1997]).

Nesse sentido, o ambiente COCAR fornece suporte à rastreabilidade entre os requisitos funcionais e o Modelo de Casos de Uso. A determinação da rastreabilidade é feita de forma automática e é possível por dois motivos:

- 1) As especificações dos requisitos funcionais são armazenadas individualmente, uma vez que o repositório foi baseado nas diretrizes propostas por Kawai (2005). O uso desse repositório de requisitos permite que eles sejam tratados separadamente. Do contrário, seria necessário que um Caso de Uso fizesse referência a uma parte do texto corrido do Documento de Requisitos, o que dificultaria a determinação e a manutenção da rastreabilidade.
- 2) O Modelo de Casos de Uso é obtido a partir da especificação dos Requisitos, por meio da aplicação da técnica TUCCA [Belgamo, 2004]. Nesse sentido, não é necessário que o usuário da ferramenta indique manualmente qual Caso de Uso estaria relacionado com qual Requisito Funcional, sendo essa informação derivada automaticamente da aplicação dos passos da TUCCA.

Por essas razões, a determinação da rastreabilidade entre os Requisitos Funcionais e os Casos de Uso do software é obtida de forma automática, seguindo as orientações de Zisman e Spanoudakis (2004). O risco dessa rastreabilidade ser obtida de maneira incorreta é muito baixo, uma vez que ele não é gerado com base na análise textual, conforme criticado por Alexander (2005). No caso da TUCCA, os próprios passos da técnica encarregam-se de garantir a consistência dessa dependência entre os requisitos funcionais do documento de requisitos e os Casos de Uso gerados ([Belgamo & Fabbri, 2004a], [Belgamo & Fabbri, 2004b] e [Belgamo & Fabbri, 2005]). A proposta deste trabalho possibilita a geração dessa rastreabilidade, garantindo a aplicação da TUCCA sobre um repositório de requisitos, no qual os requisitos funcionais encontram-se cadastrados de forma atômica.

É interessante notar que, quando a rastreabilidade é apresentada no ambiente COCAR, tanto para os Requisitos Funcionais como para os Casos de Uso, é possível o acesso aos detalhes de ambos por meio de *hyperlinks*. O uso de *hyperlinks* e hiperdocumentos para as ferramentas de rastreabilidade de requisitos é sugerido por Zisman e Spanoudakis (2004) e Munson & Nguyen (2005).

5.4.4 – Consultas sobre os requisitos

Uma vez que os requisitos são armazenados no repositório, é possível a realização de consultas sobre os mesmos. Assim sendo, dois atributos que fornecem importantes informações para a rastreabilidade foram incorporados às diretrizes por Kawai (2005). Esses atributos foram sugeridos no trabalho de Wiegers [1999b], e incluem:

- Solicitante do Requisito: refere-se ao *stakeholder* que levantou esse requisito como necessidade para o sistema. Caso alguma alteração seja proposta nesse requisito ao longo do desenvolvimento, essa pessoa deve ser questionada sobre a mudança. Como já visto neste trabalho, o acompanhamento do requisito desde sua origem, é denominado pré-rastreamento ([Zisman e Spanoudakis, 2004] e [Letelier, 2002]).
- Gerente do Requisito: refere-se à pessoa responsável por acompanhar o ciclo de vida de um requisito, ao longo do processo de desenvolvimento de software. Dessa forma, é

possível a realização de pesquisa para verificar quais são os requisitos sob responsabilidade de determinado gerente.

5.5 - Considerações Finais

Neste capítulo foram apresentadas as características do ambiente COCAR, cujo início de implementação foi fruto deste trabalho de mestrado. Lembra-se que, embora o objeto principal de pesquisa deste trabalho tenha sido o aspecto de rastreabilidade entre os requisitos e os Casos de Uso, para que essa questão pudesse ser implementada, fez-se necessário dar início à implementação do ambiente como um todo, automatizando-se a aplicação da TUCCA [Belgamo, 2004] e da especificação de requisitos [Kawai, 2005], frutos de outros trabalhos de mestrado.

Apresentaram-se também as principais funcionalidades do ambiente e seus aspectos de implementação, descrevendo-se o processo pelo qual o ambiente foi construído, as tecnologias utilizadas e as justificativas para as decisões tomadas. Foram ainda exploradas as funcionalidades implementadas no ambiente COCAR referentes ao gerenciamento de requisitos e à rastreabilidade entre os requisitos e os Casos de Uso, justificando-se cada uma delas com o apoio da literatura consultada e das propostas de outros autores, como foi discutido com detalhes no Capítulo 4.

A seguir, no Capítulo 6, será apresentado um estudo de caso, no qual todas as características das funcionalidades aqui descritas poderão ser observadas e entendidas com maior clareza.

Capítulo 6

Exemplo de uso do ambiente COCAR

6.1 – Considerações Iniciais

No capítulo anterior foi apresentado o ambiente COCAR, descrevendo-se suas principais funcionalidades com foco, sobretudo, nos aspectos de rastreabilidade. Neste capítulo será apresentado um exemplo de uso do ambiente descrevendo-se os passos que devem ser seguidos para a criação de um sistema e cadastramento de seus requisitos, seguindo as diretrizes propostas por Kawai (2005). Também será mostrada a criação do Modelo de Casos de Uso por meio da aplicação da TUCCA [Belgamo, 2004], além das funcionalidades relativas à rastreabilidade de requisitos.

O exemplo em questão é referente a um sistema denominado UserPrePaid, desenvolvido pela empresa VoxAge Teleinformática, que atua no seguimento de CTI (*Computer Telephone Integration*), com sede em São Paulo. Trata-se de um sistema simples e com poucas funcionalidades, mas capaz de ilustrar o funcionamento do ambiente COCAR, abrangendo todas as funcionalidades referidas anteriormente. O objetivo desse sistema é dar suporte aos usuários de cartões pré-pagos para realização da compra de créditos e verificação do histórico de atividades, exigindo para isso a autenticação do usuário. Devido à simplicidade do sistema, ele já foi implementado e pôde-se acompanhar o relacionamento e a evolução dos requisitos conforme será detalhado no decorrer deste capítulo.

A apresentação do ambiente COCAR é descrita na Seção 6.2. Na Seção 6.3 é enfocada a criação de um sistema. O cadastramento de requisitos é abordado na Seção 6.4; a criação do Modelo de

Casos de Uso fica detalhada na Seção 6.5, enquanto que os aspectos de rastreabilidade, na Seção 6.6. Por fim, na Seção 6.7, apresentam-se as considerações finais.

Ainda vale lembrar, conforme já salientado no Capítulo 5, que como duas das principais funcionalidades do ambiente COCAR (inserção de requisitos de software baseado nas diretrizes definidas por Kawai (2005) e geração do Modelo de Casos de Uso baseado na técnica TUCCA de Belgamo (2004)) são partes comum deste trabalho e do mestrado de Martins (2007), os itens 6.2, 6.3, 6.4 e 6.5 foram escritos em conjunto com o trabalho de Martins (2007) e, por isso, apresentam textos equivalentes.

6.2 - Apresentação do ambiente COCAR

O projeto de interfaces do ambiente COCAR foi implementado pensando-se na melhor forma de interação entre o usuário e o sistema, valendo-se dos benefícios gerados pelos protótipos criados, conforme comentados na Seção 5.3.1. Para auxiliar a navegação, as funcionalidades ficam disponíveis ao usuário no momento em que podem ser aplicadas.

Para aumentar o entendimento do usuário em relação à forma correta de uso do ambiente COCAR, a primeira tela mostrada diz respeito a um texto de ajuda mostrando ao usuário as principais funcionalidades, com *hyperlinks* para detalhes de cada uma das funcionalidades. Essa tela é mostrada na Figura 12

Após a apresentação das instruções para o usuário, a primeira etapa a ser processada para a elaboração do Modelo de Casos de Uso é a escolha de um sistema já existente ou a criação de um novo sistema. Essa interface fez-se necessária, pois os menus referentes à técnica TUCCA e à geração da métrica de PCU só ficam disponíveis quando se tem um sistema cadastrado e ativo no ambiente COCAR, como é mostrado na Figura 13 e na Figura 14.

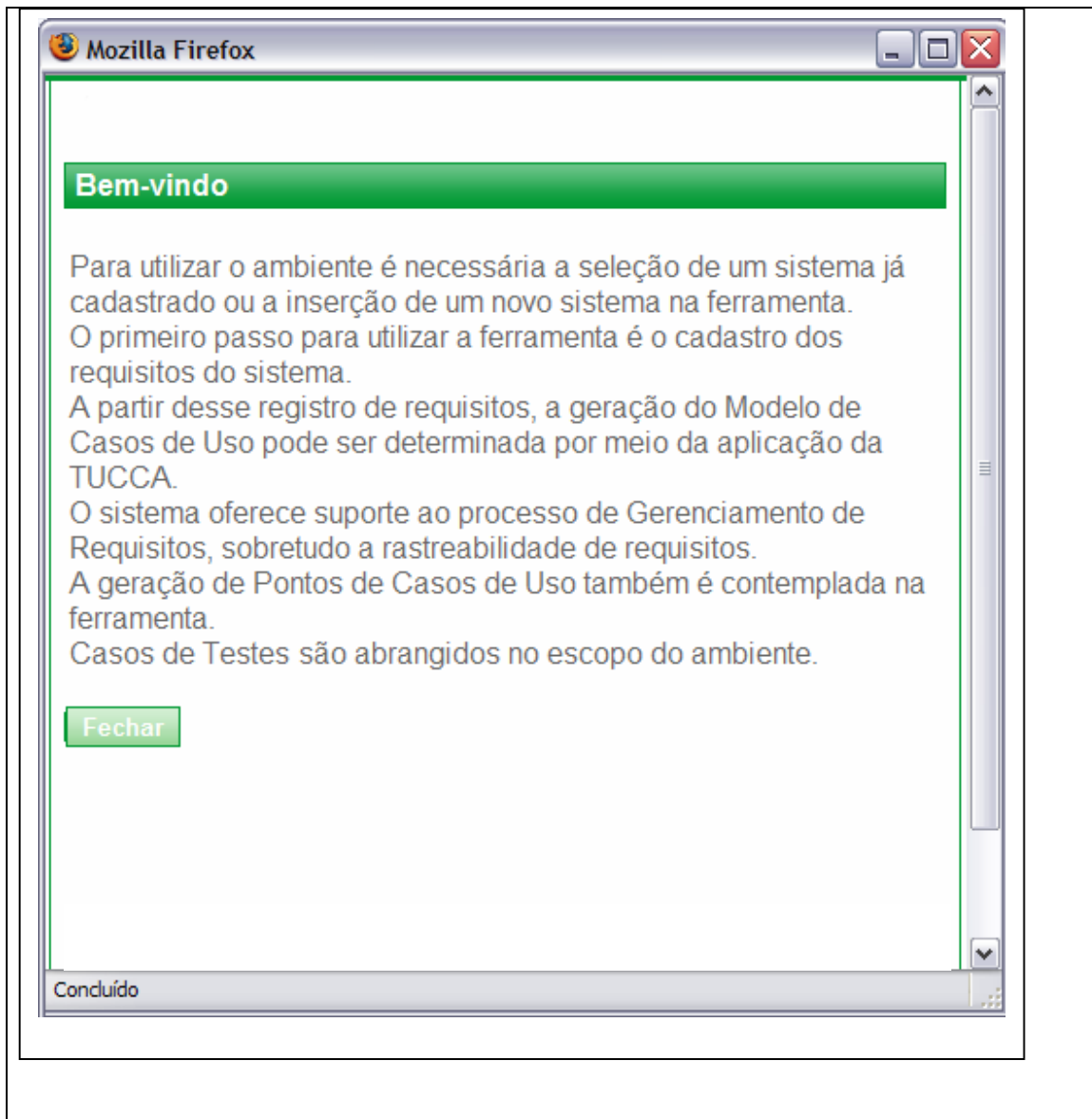


Figura 12 – Interface de ajuda do sistema COCAR

A criação do sistema e os passos subsequentes para a geração do Modelo de Casos de Uso são mostrados nas próximas seções.

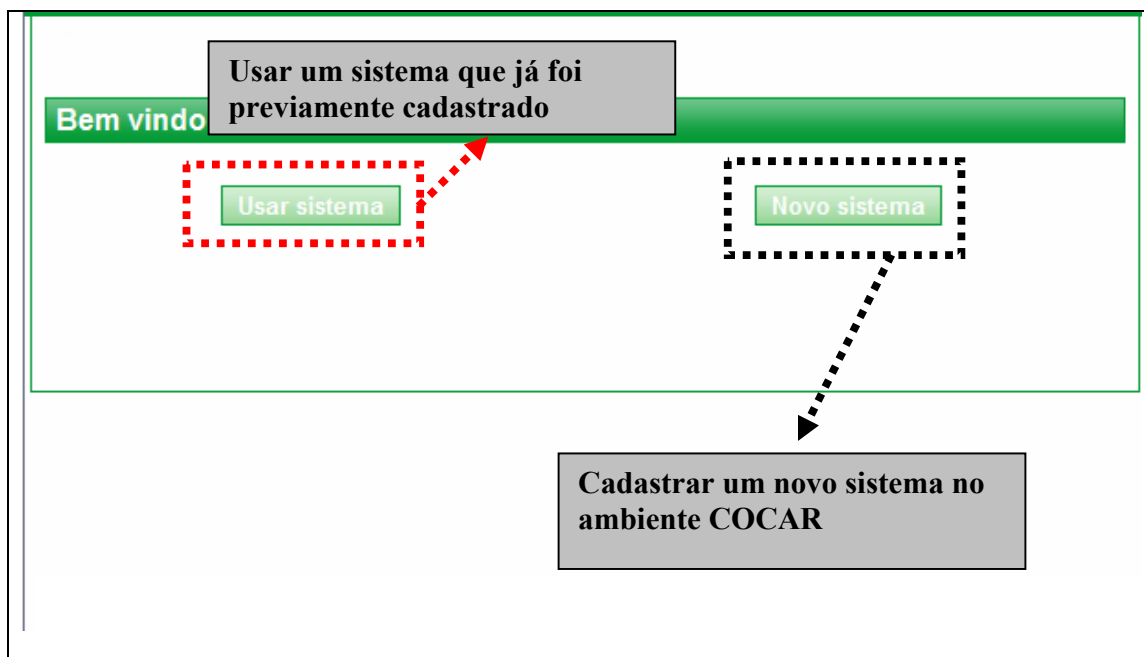


Figura 13 – Escolha de um sistema ou a criação de um novo sistema

Sistemas cadastrados					
	Nome	Descrição	Cliente	Responsável	Data cadastro
<input type="radio"/>	Sistema de Seguradora	Sistema para controle das tarefas da seguradora.	PPG CC UFSCar	Gilberto Gil	14/11/2006
<input type="radio"/>	Locadora de Veículos	Sistema de locadora de veículos.	PPG CC UFSCar	Gilberto Gil	20/11/2006
<input type="radio"/>	SAPES	Sistema Sapes	PPG CC UFSCar	Gilberto Gil	17/06/2006
<input type="radio"/>	Sistema Hoteleiro	Sistema Hoteleiro		Inezita Barroso	17/06/2006
<input checked="" type="radio"/>	UserPrePaid	O UserPrePaid é utilizado por usuários de cartões		Inezita Barroso	29/11/2006

Após escolha do sistema, ele deve ser selecionado como ativo no sistema

Selecionar como ativo

Figura 14 – Selecionar um sistema já cadastrado como ativo

6.3 – Criação de Sistema

O ambiente COCAR organiza os dados por meio de Sistemas. Neles são inseridos os requisitos do software. Quando um Sistema é criado, deve-se fornecer seu nome, descrição, cliente que o solicitou, gerente responsável e funções do produto. A Figura 15 ilustra o cadastramento de um sistema.

Bem Vindo **admin!**

Configurações do Sistema : TUCCA - Gerar Casos de Uso : PCU - Pontos de caso de

Resumo de todas as funções que o software deve executar

Sistemas

Nome	UserPrePaid
Descrição	O UserPrePaid é um software que possibilita aos usuários de cartões pré pago (calling cards) acessar seu histórico de ligações, consultar o seu saldo e realizar recargas de
Cliente	PPG DC UFSCar
Responsável	Marcos Danilo C. Martins
Funções do produto	O UserVosPrePaid fornece aos usuários de cartão pré pago as seguintes funcionalidades: 1) Autenticar Usuário; 2) Listar histórico de atividades; 3) Recarregar cartões;

Cancelar Ok

Figura 15 – Cadastramento de um sistema no ambiente COCAR

6.4 - Cadastramento de Requisitos

Conforme discutido no Capítulo 5, o repositório de requisitos do ambiente COCAR segue o modelo proposto por Kawai (2005). Além dos atributos propostos nesse modelo, foram ainda inseridos dois atributos baseados na proposta de Wieggers (1999b), sendo eles: o solicitante e o gerente do requisito. Apesar de boa parte desses atributos já terem sido apresentados e discutidos no trabalho de Kawai (2005), eles serão detalhados na seqüência, enfocando-se suas particularidades no ambiente COCAR.

- Identificação do Requisito: a identificação dos requisitos é feita internamente, mantendo-se a consistência no banco de dados. Esse aspecto de implementação é transparente para o usuário.
- Descrição: explicação breve do requisito. O objetivo deste item é registrar “o quê” a funcionalidade deve fazer e não o “como”. Uma vez cadastrados diversos requisitos no sistema, uma listagem deste item “descrição” permite uma boa idéia de tudo o que o

sistema deve realizar. Na interface do ambiente COCAR utilizou-se esse campo para apresentar um requisito quando fosse necessário.

- Processamento: composto pelo fluxo de atividades ou ações que o sistema deve realizar para alcançar o que está registrado no item “Descrição”. O campo do processamento permite inserção de grandes textos.
- Entradas: nesse campo são indicados todos os dados que estão envolvidos na funcionalidade. É composto por uma lista de dados que são cadastráveis no ambiente, indicando-se o nome, a descrição e o tipo (inteiro, *float*, texto, data ou lógico) do dado de entrada.
- Restrição: nesse campo são registradas restrições ou condições necessárias para que o requisito seja realizado. O nome original desse campo era Condição/Restrição e no contexto do ambiente foi renomeado simplesmente para Restrição.
- Saída: a resposta do sistema ao requisito deve ser indicada nesse campo.
- Ator Principal: sua nomenclatura original era Agente Fornecedor/Receptor. No contexto do ambiente foi renomeado para Ator Principal. É representado por uma lista de atores que podem ser cadastrados no sistema, indicando-se o nome e a descrição. Mais de um agente pode ser selecionado para um requisito. Representa o ator que fornece ou utiliza os dados no sistema.
- Ator Executor: originalmente era chamado Agente Executor. Representado por uma lista de atores que podem ser cadastrados no sistema, indicando-se o nome e a descrição. Mais de um agente pode ser selecionado para um requisito. Representa o ator que utiliza o sistema, inserindo os dados fornecidos pelo Ator Principal.
- Solicitante do Requisito: esse atributo não foi baseado nas diretrizes apresentadas por Kawai (2005). Ele é fruto da sugestão de Wiggers (1999b) e foi inserido no ambiente COCAR com o propósito de dar suporte à pré-rastreabilidade, conforme definição de Zisman e Spanoudakis (2004) e Letelier (2002). Nesse caso, o usuário do sistema deve escolher os solicitantes do requisito dentre os disponíveis em uma lista. Os solicitantes são cadastrados no ambiente COCAR, fornecendo-se nome e descrição.
- Gerente do Requisito: esse atributo também não foi baseado nas diretrizes apresentadas por Kawai (2005), sendo proposta de Wiggers (1999b). Esse atributo também tem foco na rastreabilidade. Nesse caso, o usuário do sistema deve escolher somente um gerente

dentre os disponíveis em uma lista. Os gerentes são cadastrados fornecendo-se nome e descrição.

Bem Vindo **admin!** Sistema: **UserPrePaid** [Trocar Sistema](#)

Configurações do Sistema : TUCCA - Gerar Casos de Uso : PCU - Pontos de caso de uso : Rastreabilidade :

Requisitos	
Descrição	Solicitar senha - Enviar email contendo a senha do usuário
Processamento	Para que o sistema possa enviar a senha para o usuário este deverá digitar o nº do seu cartão e solicitar ao sistema o envio da senha. O sistema consulta o cadastro do cartão e caso o cartão seja válido e possua um email cadastrado ele envia a senha para o email, caso contrário exibe mensagem de erro para o usuário.
Entradas	<div style="border: 1px solid black; padding: 2px;"> Início Reserva Itens Bibliograficos modelo Nome Nome Numero Cartao </div> Dica: Use CTRL para selecionar várias entradas
Restrição	
Saída	Senha enviada para o email cadasatrado; Mensagem de cartão na cadastrado. Mensagem de email não cadastrado para o cartão
Ator Principal	<div style="border: 1px solid black; padding: 2px;"> Cliente Gerente Site Pesquisador Responsavel Usuário Calling Cards </div> Dica: Use CTRL para selecionar vários atores
Ator Executor	<div style="border: 1px solid black; padding: 2px;"> Cliente Gerente Site Pesquisador Responsavel Usuário Calling Cards </div> Dica: Use CTRL para selecionar vários atores
Solicitante do requisito	<div style="border: 1px solid black; padding: 2px;"> Chico Buarque Joao Gilberto </div> Dica: Use CTRL para selecionar vários requisitantes
Gerente do requisito	Inezita Barroso

[Cancelar](#) [Ok](#)

Figura 16 – Cadastramento de Requisitos no ambiente COCAR

A Figura 16 mostra a tela do ambiente COCAR, na opção de cadastro de requisito. Percebe-se que nela, conforme especificado por Kawai (2005), os campos Entrada, Ator Principal, Ator Executor, Solicitante do Requisito e Gerente do Requisito são representados por um componente

list box (componente HTML que permite a seleção de vários itens previamente cadastrados) e composto por uma lista de dados que são cadastráveis no ambiente. O cadastro desses dados deve ser feito antes da inserção dos requisitos por meio das telas representadas pela Figura 17, Figura 18, Figura 19 e Figura 20.

Nome da entrada identificada para um determinado requisito

Texto trazendo maiores informações sobre a entrada

Tipo que essa entrada deve aceitar

Figura 17 – Cadastramento de Entradas no ambiente COCAR

Nome do ator identificado na análise dos requisitos

Descrição das atribuições do ator no sistema

Figura 18 – Cadastramento de Atores (primário e secundário)

Figura 19 – Cadastramento de Solicitante de Requisitos

Figura 20 – Cadastramento de Gerente de Requisito

6.5 – Criação do Modelo de Casos de Uso

No ambiente COCAR o Modelo de Casos de Uso é gerado seguindo os passos que compõem a TUCCA, tendo como ponto de partida o documento de requisitos definido de acordo com as diretrizes estabelecidas por Kawai (2005), implementadas da forma apresentada na seção anterior. Como já foi apresentado no Capítulo 4, a TUCCA é composta de duas técnicas de leituras: a *Actor-Goal Reading Technique* (AGRT), que recebe como entrada o documento de requisitos e tem o objetivo de identificar os possíveis atores e os objetivos de uso do sistema; e a

Use Case Reading Technique (UCRT), responsável pela identificação dos Casos de Uso e da elaboração de suas especificações.

As interfaces foram implementadas por meio de uma abordagem *wizard*, ou seja, o ambiente COCAR oferece um passo a passo para a execução das atividades que fazem parte do processo de geração do Modelo de Casos de Uso, respeitando-se a precedência entre os passos a serem seguidos. Dessa forma, procura-se facilitar a aplicação da TUCCA pelo usuário, à medida que a seqüência dos passos conduz naturalmente à criação do Modelo de Casos de Uso.

6.5.1 - Actor-Goal Reading Technique (AGRT)

Após a inserção dos requisitos do sistema para o qual se deseja definir o modelo de Casos de Uso, o usuário pode dar início à aplicação da AGRT. Ela é composta de quatro passos principais:

- i) Marcação dos atores, funções e restrições.
- ii) Determinação dos atores e objetivos.
- iii) Resolução de problemas de redundância.
- iv) Criação do Formulário Ator X Objetivo (FAO).

Em seguida, será abordado cada um desses passos, mostrando as telas do ambiente COCAR para uma melhor visualização da solução.

i) Marcação dos atores, funções e restrições

Nesse passo, são exibidos para o usuário cada um dos requisitos funcionais cadastrados, além das funções do produto, que foram inseridas no ambiente previamente, conforme descrito nos itens 7.2 e 7.3. O usuário deve ler cada um dos itens apresentados e marcar os candidatos a ator (amarelo), as funções (objetivos) (rosa) e as restrições (azul) existentes no texto. Esse processo é apresentado na Figura 21.

Cadastro
Sair

atos de caso de uso : Rastreabilidade :

Para a marcação dos requisitos, o usuário seleciona a palavra e aperta um dos botões “ator”, “função” ou “restrição”. Automaticamente, a palavra assume a cor da marcação escolhida

Funções do produto

O UserVosPrePaid fornece aos **usuários** de cartão pré pago as seguintes funcionalidades: 1) **Autenticar** Usuário; 2) **Listar** histórico de atividades; 3) **Recarregar** cartões;

<p>Requisito - Relacionar Cartão a Usuário - Cadastrar dados do usuário para um determinado cartão</p> <p>No primeiro acesso do usuário ao site o sistema deverá relacionar o cartão a usuário. Para tanto o usuário informa o número do cartão, senha, confirmação da senha e e-mail e o sistema associa as informações ao número de cartão e configura o acesso para o usuário. Caso o número informado já esteja cadastrado, o sistema mostra a mensagem dizendo que o cartão já está cadastrado. Caso contrário, e o usuário tenha informado um email válido e o campo senha seja igual ao campo confirmação senha, o sistema mostra uma mensagem de sucesso. Após digitar os dados do cartão e a senha o sistema realiza um post para a API que retorna sucesso ou falha.</p>	<input type="button" value="ator"/> <input type="button" value="funcao"/> <input type="button" value="restricao"/>
<p>Requisito - Autenticar Usuário - Deve permitir o acesso do usuário ao site através do número do cartão e senha.</p> <p>Para autenticar o usuário este deverá digitar o nº do cartão e a senha para entrar no sistema. A senha é cadastrada após o 1º acesso do cartão. Após digitar o nº do cartão e a senha o sistema realiza um post para a API que retorna sucesso ou falha.</p>	<input type="button" value="ator"/> <input type="button" value="funcao"/> <input type="button" value="restricao"/>
<p>Requisito - Listar Atividades - Listar atividades realizadas pelo usuário com o cartão</p> <p>Para listar as atividades do cartão o usuário, após estar logado no sistema, pode escolher listar as atividades executadas no seu cartão. Dessa forma o sistema deverá exibir os seguintes dados de todas as ligações e recargas: Data, Hora, Dia da Semana, ANI, ToNumber, Tempo, Destination Area e Saldo. O usuário pode escolher não mostrar todas as opções, mas sim apenas aquelas que atendem a determinados parâmetros configurados através de filtros, a saber: Data Inicial, Data Final e/ou ToNumber;</p>	<input type="button" value="ator"/> <input type="button" value="funcao"/> <input type="button" value="restricao"/>
<p>Requisito - Recarregar Cartão - Inserir créditos em um cartão</p> <p>Para realizar a recarga do cartão o usuário informa seu Nome, o tipo do seu cartão de crédito, o número do cartão de crédito, a data de vencimento desse cartão e o código de verificação. Posteriormente o usuário escolhe um plano de recarga de 30, 50, 75 ou 100 reais. O sistema então busca aprovação da compra na operadora do cartão de crédito e caso ela seja conseguida o sistema reajusta o valor do saldo do cartão de acordo com o plano escolhido, caso contrário o sistema mostra uma mensagem de erro para o usuário</p>	<input type="button" value="ator"/> <input type="button" value="funcao"/> <input type="button" value="restricao"/>
<p>Requisito - Solicitar senha - Enviar email contendo a senha do usuário</p> <p>Para que o sistema possa enviar a senha para o usuário este deverá digitar o nº do seu cartão e solicitar ao sistema o envio da senha. O sistema consulta o cadastro do cartão e caso o cartão seja válido e possua um email cadastrado ele envia a senha para o email, caso contrário exibe mensagem de erro para o usuário.</p>	<input type="button" value="ator"/> <input type="button" value="funcao"/> <input type="button" value="restricao"/>

Figura 21 - Processo de marcação de atores, funções e restrições

ii) Determinação dos atores e objetivos

A partir da marcação, toma-se como referência cada uma das funções levantadas e solicita-se ao usuário a determinação do objetivo e de qual dos atores está vinculado com esse objetivo. Esse processo deve ser realizado para cada uma das funções que foram marcadas no passo anterior. Salienta-se que, durante a execução desse passo, a criação do FAO está sendo realizado, simultaneamente. A Figura 22 ilustra esse processo.

Escolha do ator

Associando a função a um objetivo

Função no texto do requisito que está gerando esse objetivo

FAO: Ator e Objetivo

Funções do produto

O UserVosPrePaid fornece aos **usuários** de cartão pré pago as seguintes funcionalidades: 1) **Autenticar** Usuário; 2) **Listar** histórico de atividades; 3) **Recarregar** cartões;

Ator	Objetivo	Referência
<input checked="" type="checkbox"/> Usuário Calling Cards	Autenticar Usuario	autenticar
<input checked="" type="checkbox"/> Usuário Calling Cards	Listar Atividades	Listar
<input checked="" type="checkbox"/> Usuário Calling Cards	Recarregar Cartao	Recarregar

Requisito

No primeiro acesso do usuário ao site o **sistema** deverá **relacionar** o cartão a usuário. Para tanto o usuário informa o número do cartão, senha, confirmação da senha e e-mail e o **sistema** associa as informações ao número de cartão e configura o acesso para o usuário. Caso o número informado já esteja cadastrado, o **sistema** mostra a mensagem dizendo que o cartão já está cadastrado. Caso contrário, e o usuário tenha informado um email válido e o campo senha seja igual ao campo confirmação senha, o **sistema** mostra uma mensagem de sucesso. Após digitar os dados do cartão e a senha o **sistema** realiza um post para a **API** que retorna sucesso ou falha.

Ator	Objetivo	Referência
<input checked="" type="checkbox"/> Usuário Calling Cards	Relacionar Cartao a Usuario	relacionar

Requisito

Para **autenticar** o **usuário** este deverá digitar o nº do cartão e a senha para entrar no sistema. A senha é cadastrada após o 1º acesso do cartão. Após digitar o nº do cartão e a senha o sistema realiza um post para a **API** que retorna sucesso ou falha.

Ator	Objetivo	Referência
<input checked="" type="checkbox"/> Usuário Calling Cards	Autenticar Usuario	autenticar

Requisito

Para **listar** as atividades do cartão o **usuário**, após estar **logado** no **sistema**, pode escolher **listar** as atividades executadas no seu cartão. Dessa forma o **sistema** deverá exibir os seguintes dados de todas as ligações e recargas: Data, Hora, Dia da Semana, ANI, ToNumber, Tempo, Destination Area e Saldo. O **usuário** pode escolher não mostrar todas as opções, mas sim apenas aquelas que atendem a determinados parâmetros configurados através de filtros, a saber: Data Inicial, Data Final e/ou ToNumber.

Ator	Objetivo	Referência
<input checked="" type="checkbox"/> Usuário Calling Cards	Listar Atividades	listar

Requisito

Para realizar a **recarga** do cartão o **usuário** informa seu Nome, o tipo do seu cartão de crédito, o número do cartão de crédito, a data de vencimento desse cartão e o código de verificação. Posteriormente o **usuário** escolhe um plano de **recarga** de 30, 50, 75 ou 100 reais. O **sistema** então busca aprovação da compra na **operadora** do cartão de crédito e caso ela seja conseguida o **sistema** reajusta o valor do saldo do cartão de acordo com o plano escolhido, caso contrário o **sistema** mostra uma mensagem de erro para o **usuário**.

Ator	Objetivo	Referência
<input checked="" type="checkbox"/> Usuário Calling Cards	Recarregar Cartao	recarga

Requisito

Para que o **sistema** possa **enviar** a senha para o usuário este deverá digitar o nº do seu cartão e solicitar ao **sistema** o envio da senha. O **sistema** **consulta** o cadastro do cartão e caso o cartão seja válido e possua um email cadastrado ele envia a senha para o email, caso contrário exibe mensagem de erro para o usuário.

Ator	Objetivo	Referência
<input checked="" type="checkbox"/> Usuário Calling Cards	Enviar Senha	enviar
<input checked="" type="checkbox"/> sistema	Consulta Cartao	consulta

Continuar

Figura 22 – Determinação dos atores e objetivos

iii) Resolução de problemas de redundância

Uma vez determinados os atores e os objetivos do sistema, é necessário retirar as possíveis redundâncias que possam existir, tanto ao que diz respeito aos objetivos identificados como também aos atores que interagem com o sistema. Deve-se atentar para dois tipos de redundância.

A primeira, chamada redundância intra-atores, refere-se ao caso de dois objetivos de um mesmo ator serem exatamente os mesmos, apesar de terem denominação diferente no documento de requisitos. Isso significa que, apesar de sintaticamente diferentes, semanticamente eles são iguais. Como exemplo, tem-se “Cadastrar Cliente” e “Salvar Cliente” que, em uma determinada situação, para um determinado documento de requisitos, podem significar a mesma funcionalidade e, portanto, devem ser fundidos em um único objetivo.

O segundo tipo de redundância é chamada de redundância inter-atores. Nesse caso, temos objetivos sintaticamente diferentes relacionados a atores distintos, mas contendo o mesmo teor semântico. Como exemplo, podemos ter o objetivo “Cadastrar Pedido” sendo realizado pelo ator “Cliente” e o objetivo “Armazenar Pedido” sendo realizado pelo ator “Usuário”. Os dois objetivos, apesar de estarem escritos de forma diferente, tem o mesmo significado e devem ser agrupados em apenas um objetivo.

Na Figura 23, é mostrada a interface disponibilizada pelo ambiente COCAR para resolver as redundâncias intra-atores e inter-atores. No exemplo em questão, não há nenhuma redundância desse tipo a ser resolvida.

Para cada ator, a ferramenta disponibiliza os seus objetivos e permite que o usuário agrupe nas caixas de seleção os objetivos que são semanticamente iguais

Todos os objetivos são listados nas caixas de seleção permitindo que o usuário possa agrupar objetivos semanticamente iguais e que são realizados por atores diferentes. Caso os objetivos agrupados sejam de um mesmo ator, a ferramenta acusa um erro

FAO: Redundâncias Intra-Atores	
Ator: Sistema	
Redundâncias intra-atores: <input type="text" value="Selecione"/> <input type="text" value="Selecione"/> <input type="button" value="Agrupar"/>	
Consulta Cartao	
Ator: Usuário Calling Cards	
Redundâncias intra-atores: <input type="text" value="Selecione"/> <input type="text" value="Selecione"/> <input type="button" value="Agrupar"/>	
Autenticar Usuario	FP
Listar Atividades	FP
Recarregar Cartao	FP
Relacionar Cartao a Usuario	RF0
Autenticar Usuario	RF1
Listar Atividades	RF2
Recarregar Cartao	RF3
Enviar Senha	RF4
FAO: Redundâncias Inter-Atores	
<input type="text" value="Selecione"/> <input type="text" value="Selecione"/> <input type="button" value="Agrupar"/>	

Figura 23 - Exemplo de redundância intra-atores

iv) Criação do Formulário Ator X Objetivo (FAO)

Depois de executados os passos anteriores, tem-se o FAO concluído, uma vez que ele vai sendo elaborado à medida que os outros passos são realizados. Esse formulário apresenta os atores e todos os seus objetivos no sistema. Um exemplo do FAO é mostrado na Figura 24.

The diagram shows a table titled "Formulário Ator X Objetivo" with three columns: "Ator", "Objetivo", and "Referência". The table is enclosed in a red dashed border. Three grey callout boxes with red arrows point to the table: the first points to the "Ator" column, the second to the "Objetivo" column, and the third to the "Referência" column.

Ator	Objetivo	Referência
sistema	Consulta Cartao	RF4
Usuário Calling Cards	Autenticar Usuario	FP
	Recarregar Cartao	FP
	Relacionar Cartao a Usuario	RF0
	Autenticar Usuario	RF1
	Listar Atividades	FP
	Recarregar Cartao	RF3
	Enivar Senha	RF4
	Listar Atividades	RF2

Figura 24 - Exemplo do FAO gerado pela aplicação da AGRT no ambiente COCAR

6.5.2 - Use Case Reading Technique (UCRT)

A UCRT utiliza como entrada o FAO gerado pela AGRT e os requisitos do sistema. Como saída são gerados o Diagrama de Casos de Uso e a Especificação de Casos de Uso.

A UCRT é composta por duas etapas: Criação dos Casos de Uso Preliminares e Criação das Especificações e Possíveis Relacionamentos entre os Casos de Uso. A seguir, é mostrado como cada uma dessas etapas foi implementada no ambiente COCAR.

i) Criação de Casos de Uso Preliminares (Etapa I)

O intuito principal dessa etapa é identificar os Casos de Uso preliminares usando para isso os objetivos existentes no FAO. Além disso, busca-se nessa fase a identificação dos possíveis relacionamentos e associações existentes entre esses Casos de Uso preliminares.

Para tanto, o primeiro passo dessa etapa consiste em tornar cada objetivo do FAO, que tem como referência a seção “Funções do Produto”, em um caso de uso preliminar, pois como essa seção do

documento de requisitos apresenta as principais funcionalidades do sistema, elas são fortes candidatas a tornarem-se Casos de Uso. Após isso, devem-se percorrer os objetivos do FAO em busca daqueles que tiverem o mesmo nome dos objetivos já transformados em Casos de Uso Preliminares, com o intuito de agrupar as referências associadas a eles, uma vez que esses objetivos foram detalhados em outros requisitos do documento de requisitos, além de serem citados na seção “Funções do Produto”. Com isso, facilita-se a especificação do caso de uso, uma vez que todos os pontos em que esse objetivo foi citado estão agora agrupados. O ambiente COCAR faz essa operação automaticamente, como pode ser percebido na Figura 25.

Primeiro Passo
 1) Objetivos com Referência FP transformados em Casos de Uso
 2) Funcionalidades Idênticas: agrupar atores e referências

		Caso de Uso	Referência	Relacionamento	Include
1	Usuário Calling Cards Usuário Calling Cards	Autenticar Usuario	FP, RF1		
2	Usuário Calling Cards Usuário Calling Cards	Recarregar Cartao	FP, RF5		
3	Usuário Calling Cards Usuário Calling Cards	Listar Atividades	FP, RF4		
4	sistema	Consulta Cartao	RF2	6	
5	Usuário Calling Cards	Relacionar Cartao a Usuario	RF3		
6	Usuário Calling Cards	Enivar Senha	RF2	4	

Segundo Passo
 Possibilidade de agrupar objetivos que possuem o mesmo conjunto de referências, caso esses tenham o mesmo significado semântico.

Especificar

Figura 25 – Primeira Etapa: Formulário de Casos de Uso Preliminares

No segundo passo dessa etapa, analisa-se a possibilidade de agrupamento ou não de todos os objetivos que possuem o mesmo conjunto de referências (daqueles ainda não transformados em Casos de Uso Preliminares). No exemplo da Figura 25, o requisito 4 tem a mesma referência

“RF2” e os dois requisitos aparecem no campo “Fundir” dessa tela. No exemplo em questão, “consultar cartão” e “enviar senha” são objetivos que, apesar de terem origem no mesmo requisito, são semanticamente diferentes um do outro e, por isso, não serão fundidos.

O terceiro passo dessa etapa corresponde a transformar em Casos de Uso independentes todos os objetivos restantes no FAO. O ambiente COCAR realiza esse passo automaticamente.

ii) Criação das Especificações e possíveis relacionamentos entre os Casos de Uso (Etapa II)

Nessa etapa, o objetivo principal é especificar os Casos de Uso definidos no Formulário de Casos de Uso Preliminares gerados na etapa anterior.

Para tanto, o ambiente COCAR oferece uma interface que disponibiliza os Casos de Uso Preliminares para a especificação, sendo que os Casos de Uso que contêm apenas referência para a seção de “Funções do Produto” são disponibilizados primeiro, seguidos dos Casos de Uso que tiverem menos referências, conforme o definido na técnica UCRT [Belgamo, 2004]. O objetivo dessa ordem é especificar primeiro os Casos de Uso que têm probabilidade de serem mais simples do que aqueles que estão associados com vários requisitos do documento de requisitos. Esses que estão associados com vários requisitos, além de normalmente serem mais complexos, ainda têm a probabilidade de envolver Casos de Uso que já tenham sido especificados e que podem ter uma associação do tipo <<include>> ou <<extend>> com o Caso de Uso mais complexo.

Para facilitar a elaboração da especificação do caso de uso, o ambiente COCAR disponibiliza em sua interface a facilidade de mostrar os requisitos que estão relacionados com o caso de uso que está sendo especificado, possibilitando ao usuário ler e aproveitar o texto do requisito para poder compor os cursos normal e alternativo do Caso de Uso.

Além disso, é possível incluir cursos alternativos por meio do botão “Incluir” da linha de curso alternativo, ou ainda, definir associações entre Casos de Uso com estereótipos <<includes>> ou <<extends>> por meio da lista de Casos de Uso relacionados.

A Figura 26 mostra a interface oferecida pelo ambiente COCAR para especificação dos Casos de Uso, de acordo com os detalhes discutidos.

Inclusão de Casos de Uso através de <<includes>> ou <<extends>> no curso normal

Inclusão de Casos de Uso através de <<includes>> ou <<extends>> nos cursos alternativos

Caso de uso	
Nome	Enivar Senha
Descrição	Este caso de uso é responsável por usuário que a perdeu ou a esqueceu
Atores	Usuário Calling Cards
Pré-condição	Nenhuma
Curso Normal	<p>Incluir Casos de uso relacionados: <input type="text" value="Selecione"/></p> <ol style="list-style-type: none"> 1. Usuario informa o numero do seu cartao e solicita o envio da senha 2. Sistema verifica que o cartao existe e há um email associado a ele 3. Sistema envia a senha para o email cadastrado
Curso Alternativo	<p>Incluir</p> <p>2 a.</p> <ol style="list-style-type: none"> 1. Sistema verifica que o cartão não existe 2. Sistema emite msg de erro
Evento Disparado	Usuário solicita envio de senha
Autor	admin
Data	14/12/2006
Versão	1

Requisitos relacionados

Solicitar senha - Enviar email contendo a senha do usuário

Para que o sistema possa enviar a senha para o usuário este deverá digitar o nº do seu cartão e solicitar ao sistema o envio da senha. O sistema consulta o cadastro do cartão e caso o cartão seja válido e possua um email cadastrado ele envia a senha para o email, caso contrário exibe mensagem de erro para o usuário.

CursoAlternativo

Curso Alternativo de:

Include:

 Extend:

Incluir

1. Sistema verifica que o cartão não existe
2. Sistema emite msg de erro

Figura 26 – Interface para a especificação dos Casos de Uso e Janela com a janela dos requisitos relacionados e janela de especificação de curso alternativo

6.6 – Aspectos de Rastreabilidade

Como visto no capítulo anterior, o ambiente COCAR implementa quatro funcionalidades relacionadas com a rastreabilidade, sendo elas:

- Geração da Matriz de Rastreabilidade de Requisitos.
- Criação dos Indicadores de Estabilidade.
- Rastreabilidade entre Documento de Requisitos e Modelo de Casos de Uso.
- Consultas sobre os Requisitos.

Cada uma delas será abordada na seqüência, exibindo-se exemplos de uso.

6.6.1 - Geração da Matriz de Rastreabilidade de Requisitos

Como visto anteriormente, a matriz de Rastreabilidade de Requisitos indica o relacionamento entre os requisitos do software. Sommerville (2003) enfatiza a importância dessa matriz para o gerenciamento de requisitos e também alerta para a dificuldade em obtê-la e mantê-la ao longo do desenvolvimento de software. Nesse sentido, o ambiente COCAR buscou uma solução para determinação automática dessa matriz, indicando ainda qual o percentual de dependência entre os requisitos.

Para a determinação da dependência entre os requisitos, o ambiente faz a comparação entre os dados de entrada de cada um dos requisitos. Caso dois requisitos manipulem os mesmos dados, a dependência entre eles tende a ser alta. Se eles estirem manipulando parcialmente os mesmos dados, a dependência entre esses requisitos deve ser menor. Caso dois requisitos de um mesmo sistema não possuam nenhum dado de manipulação em comum, não se indica nenhuma dependência entre eles na matriz. Na Figura 27 é exemplificada a Matriz de Rastreabilidade de Requisitos.

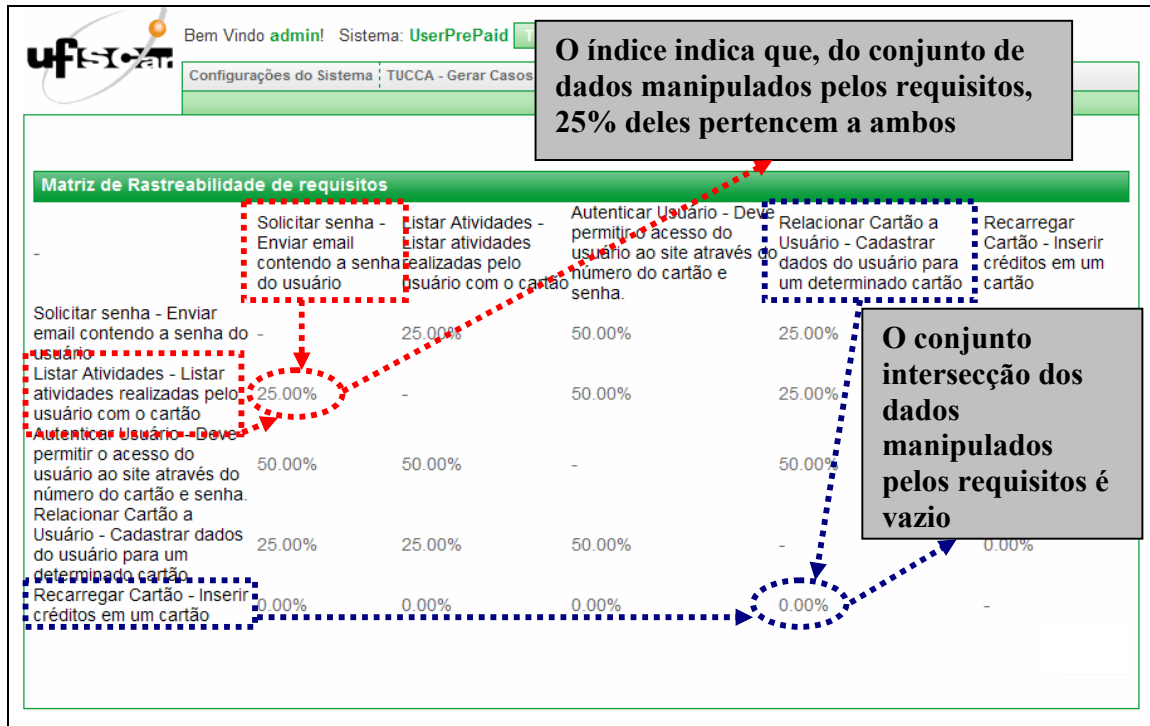


Figura 27 - Exemplo de Matriz de Rastreabilidade de Requisitos

Com o intuito de apresentar de forma mais didática o relacionamento entre os requisitos, considere um sistema de uma locadora de veículos, com os seguintes requisitos:

1. Cadastrar cliente.
2. Alterar cliente.
3. Cadastrar veículo.
4. Devolver veículo.
5. Cadastrar funcionário.

Os requisitos “Cadastrar cliente” e “Alterar cliente” manipulam os mesmos tipos de dados relacionados aos atributos de cliente (por exemplo: nome, telefone, endereço). Nesse caso, o relacionamento entre os requisitos é classificado como sendo de 100%. Isso significa que qualquer mudança em um dos requisitos representa um impacto grande no outro. Exemplificando essa dependência, caso seja inserido um novo dado para a funcionalidade “Cadastrar cliente”, por exemplo, o registro da Carteira Nacional de Habilitação (CNH), o requisito “Alterar cliente” deve ser observado, pois, provavelmente, deve sofrer algum impacto já que a dependência entre eles é alta.

Outro exemplo é a comparação do requisito “Cadastrar veículo” - com os dados de entrada de placa do veículo, modelo, ano e cor - com o requisito “Devolver veículo” - com os dados de entrada de placa do veículo, data, situação do carro e CNH do cliente. Nesse caso, a dependência entre os requisitos é de 25%, uma vez que conjunto intersecção dos dados entre os requisitos possui um quarto (25%) do conjunto de todos os dados de entrada do requisito. Nessa situação, uma mudança em um desses requisitos tende a representar um impacto pequeno no outro requisito.

Existem casos em que a dependência é nula, como no caso dos requisitos “Cadastrar funcionário” e “Alterar cliente”. Nesse caso, o conjunto intersecção dos dados pessoais do funcionário e dos dados de cliente é vazio. Para essa situação, o ambiente COCAR indica dependência de 0%.

Na Figura 28 segue outro exemplo da matriz de rastreabilidade de requisitos, considerando agora o sistema de locadora de veículos.

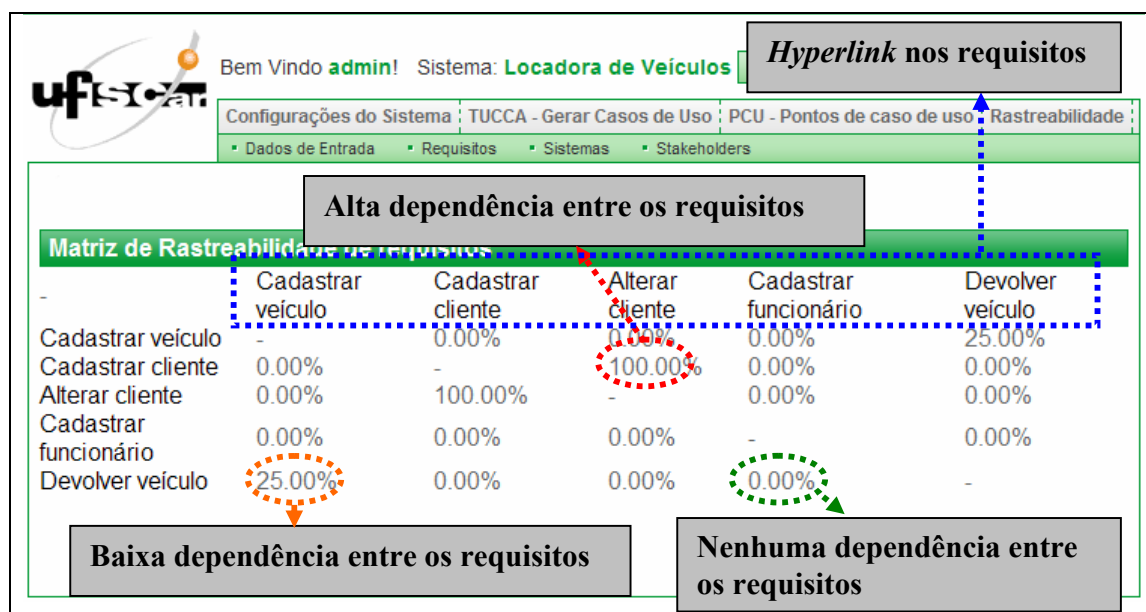


Figura 28 - Exemplo de Matriz de Rastreabilidade de Requisitos

Outro aspecto observado na visualização da Matriz de Rastreabilidade de Requisitos é o uso de *hyperlinks* na representação dos requisitos, favorecendo a rastreabilidade ([Zisman e Spanoudakis, 2004] e [Munson & Nguyen, 2005]). Sendo assim, o usuário pode, a qualquer

momento, clicar sobre os requisitos apresentados na matriz e serão exibidos os detalhes desse requisito.

6.6.2 – Criação dos Indicadores de Estabilidade

O indicador de estabilidade tem como objetivo oferecer uma medida sobre o processo de gerenciamento de requisitos. O usuário da ferramenta pode, a qualquer momento do processo de desenvolvimento, consultar o percentual de requisitos inseridos, alterados ou excluídos dentro do período de tempo que ele determinar. Para isso, basta informar a data inicial e final do período sobre o qual se deseja realizar a pesquisa dos indicadores. Isso é possível, pois cada inserção ou alteração do requisito tem a data de ocorrência registrada. Com relação à exclusão de requisitos, ela é feita logicamente, mantendo-se o requisito armazenado no banco de dados, porém registrado como excluído, salvando-se a data em que essa operação foi feita.

Na Figura 29 é apresentado o Indicador de Estabilidade. O período é determinado pelo usuário e, posteriormente, os indicadores são disponibilizados. Como exemplo, considere os cinco requisitos apresentados na Figura 27 - Exemplo de Matriz de Rastreabilidade de Requisitos. Como dois desses cinco requisitos foram alterados durante o desenvolvimento do produto, o percentual de requisitos alterados é de 40%. Caso mais um desses cinco requisitos sofra alterações, esse percentual passará a ser 60%. Com relação aos requisitos inseridos no sistema, durante o período pesquisado nenhum requisito foi acrescentado, uma vez que o percentual de requisitos inseridos é 0%. No que diz respeito aos requisitos excluídos, é possível saber que anteriormente existiam seis requisitos e um foi excluído, pois o percentual de requisitos excluídos indica que 20% de cinco requisitos, ou seja, um requisito foi excluído durante o período pesquisado.

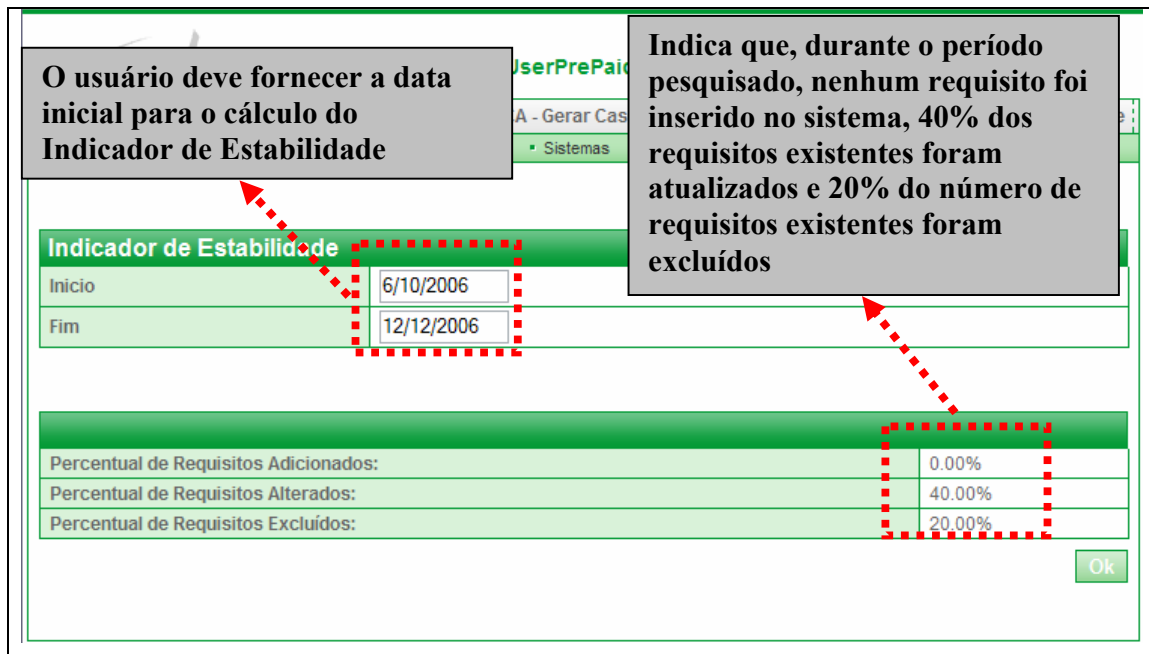


Figura 29 - Exemplo do Indicador de Estabilidade

6.6.3 – Rastreabilidade entre Documento de Requisitos e Modelo de Casos de Uso

A ligação entre os requisitos e os Casos de Uso é determinada pelo cadastramento de requisitos e aplicação da TUCCA, conforme detalhados anteriormente. Dessa forma, a dependência entre esses artefatos é obtida de forma automática. A visualização dessa dependência é exemplificada conforme exibido na Figura 30.

Bem Vindo **admin!** Sistema: **UserPrePaid** [Trocar Sistema](#)

Configurações do Sistema : TUCCA - Gerar Casos de Uso : PCU - Pontos de caso de uso : Rastreabilidade :

Rastreabilidade - Caso de Uso X Requisito	
Caso de Uso	Requisito
Relacionar Cartao a Usuario	- Relacionar Cartão a Usuário - Cadastrar dados do usuário para um determinado cartão
Recarregar Cartao	- Recarregar Cartão - Inserir créditos em um cartão
Listar Atividades	- Relacionar Cartão a Usuário - Cadastrar dados do usuário para um determinado cartão - Listar Atividades - Listar atividades realizadas pelo usuário com o cartão
Enivar Senha	- Solicitar senha - Enviar email contendo a senha do usuário - Listar Atividades - Deve exibir todas as ligações efetuadas pelo cartão exibindo os seguintes dados: Data, Hora, Dia da Semana, ANI, ToNumber, T
Consulta Cartao	- Solicitar senha - Enviar email contendo a senha do usuário

Nessa situação, o caso de uso “Listar Atividades” está relacionado com os dois requisitos da célula ao lado

Figura 30 - Exemplo de visualização da rastreabilidade entre requisito e Caso de Uso

Para o acompanhamento da evolução dos requisitos e dos Casos de Uso, é feito o controle baseado no rótulo de tempo das ligações e das próprias versões dos requisitos e Casos de Uso. Sendo assim, se a versão do requisito que se relaciona com o Caso de Uso for mais recente que o Caso de Uso, indica que o mesmo deve ser revisto, pois, como o requisito que o originou foi alterado, é possível que o Caso de Uso também necessite de atualização. Nesses casos, a dependência fica registrada no sistema com a cor vermelha, conforme pode ser visto na Figura 31. Nesse caso, o requisito “Relacionar Cartão a Usuário - Cadastrar dados do usuário para um determinado cartão”, passou por uma alteração. Sendo assim, seu rótulo de tempo foi atualizado e é mais recente que o dos Casos de Uso, indicando que os Casos de Uso “Relacionar Cartão a Usuário” e “Listar Atividades” - que surgiram da aplicação da TUCCA – devem ser observados para possível atualização.

Vale notar que tanto os requisitos como os Casos de Uso são disponibilizados com *hyperlinks*. Dessa forma, conforme comentado na Seção 6.6.1, a qualquer momento o usuário poderá clicar nesses *hyperlinks* para exibição dos detalhes inerentes a cada um deles.

Bem Vindo **admin!** Sistema: **UserPrePaid** Trocar Sistema

Configurações do Sistema : TUCCA - Gerar Casos de Uso : PCU - Pontos de caso de uso : Rastreabilidade :

Rastreabilidade - Caso de Uso X Requisito	
Caso de Uso	Requisito
Relacionar Cartao a Usuario	- Relacionar Cartão a Usuário - Cadastrar dados do usuário para um determinado cartão
Recarregar Cartao	- Recarregar Cartão - Inserir créditos em um cartão
Listar Atividades	- Relacionar Cartão a Usuário - Cadastrar dados do usuário para um determinado cartão - Listar Atividades - Listar atividades realizadas pelo usuário com o cartão
Enviar Senha	- Solicitar senha - Enviar email contendo a senha do usuário - Listar Atividades - Deve exibir todas as ligações (Data, Hora, Dia da Semana, ANI, ToNumber, T...
Consulta Cartao	- Solicitar senha - Enviar email contendo a senha

Como o requisito marcado foi alterado na ferramenta, os casos de uso aos quais ele está relacionado são marcados na cor vermelha

Figura 31 – Exemplo de indicação de alterações de em requisitos

6.6.4 - Pesquisas sobre os Requisitos.

Essa funcionalidade do ambiente COCAR refere-se à possibilidade de efetuar consultas sobre os requisitos, com o intuito de levantarem-se os requisitos que são responsabilidade de determinado gerente ou que foram solicitados por determinado *stakeholder*, além dos requisitos que são de algum sistema em especial. A Figura 32 ilustra as possibilidades de pesquisa que podem ser feitas, com filtros de sistema, gerente de requisito e solicitante de requisito.

Bem Vindo admin! Sistema: UserPreP

Configurações do Sistema : TUCCA - Gera

Pesquisa

Filtros para Pesquisa:

Sistema:

Gerente do Requisito:

Solicitante do Requisito:

Requisitos Cadastrados

	Descrição	Data	Gerente
<input type="checkbox"/>	Autenticar Usuário - Deve permitir o acesso do usuário ao site através do número...	2006-12-12	Inezita Barroso
<input type="checkbox"/>	Solicitar senha - Enviar email contendo a senha do usuário	2006-12-12	Inezita Barroso
<input type="checkbox"/>	Recarregar Cartão - Inserir créditos em um cartão	2006-12-12	Zeca Baleiro
<input type="checkbox"/>	Listar Atividades - Listar atividades realizadas pelo usuário com o cartão	2006-12-12	Inezita Barroso
<input type="checkbox"/>	Relacionar Cartão a Usuário - Cadastrar dados do usuário para um determinado ca...	2006-12-12	Gilberto Gil

O usuário pode realizar pesquisa compondo filtros por meio da seleção dos sistemas, gerentes ou solicitantes cadastrados

Os requisitos que atendem às restrições determinadas anteriormente são exibidos

Figura 32 – Possibilidades de pesquisas sobre os requisitos

A Figura 33 ilustra o filtro sobre o gerente do requisito. Nesse caso, é possível buscar todos os requisitos que estão sob a supervisão do gerente “Gilberto Gil”.

A pesquisa sobre os solicitantes de requisito é mostrada na Figura 34 . Nesse exemplo é possível listar os requisitos que foram solicitados por “Caetano Veloso”.

Bem Vindo **admin!** Sistema: **UserPrePaid** Trocar Sistema

Configurações do Sistema : TUCCA - Gerar Casos

Pesquisa

Filtros para Pesquisa:

Sistema:

Gerente do Requisito:

Solicitante do Requisito:

Requisitos Cadastrados

	Descrição	Data	Gerente
<input type="checkbox"/>	Relacionar Cartão a Usuário - Cadastrar dados do usuário para um determinado ca...	2006-12-12	Gilberto Gil

Após a seleção de um filtro de gerente de requisito, são recuperados os requisitos que atendem à restrição

Figura 33 - Consulta sobre os requisitos de determinado Gerente

Bem Vindo **admin!** Sistema: **UserPrePaid** Trocar Sistema

Configurações do Sistema : TUCCA - Gerar Casos

Pesquisa

Filtros para Pesquisa:

Sistema:

Gerente do Requisito:

Solicitante do Requisito:

Requisitos Cadastrados

	Descrição	Data	Gerente
<input type="checkbox"/>	Autenticar Usuário - Deve permitir o acesso do usuário ao site através do número...	2006-12-12	Inezita Barroso
<input type="checkbox"/>	Recarregar Cartão - Inserir créditos em um cartão	2006-12-12	Zeca Baleiro
<input type="checkbox"/>	Relacionar Cartão a Usuário - Cadastrar dados do usuário para um determinado ca...	2006-12-12	Gilberto Gil

Após a seleção de um filtro de solicitante de requisito, são recuperados os requisitos que atendem à restrição

Figura 34 - Consulta sobre os requisitos de determinado Solicitante

6.7 - Considerações Finais

Nesse capítulo foi exemplificado o uso do ambiente COCAR, desenvolvido como produto deste trabalho de mestrado. Os requisitos que foram abordados no Capítulo 5, focando-se a justificativa e a importância dos mesmos, foram aqui apresentados com exemplos práticos. Mostrou-se o uso do ambiente num exemplo prático e real, passando-se pela criação de um sistema no ambiente, cadastramento de requisitos e aplicação da TUCCA para a geração do Modelo de Casos de Uso. Por fim, foram apresentados os aspectos de rastreabilidade.

Como pôde ser observado pelos exemplos dados, as informações relativas ao gerenciamento de requisitos são, certamente, de grande valia para o gerenciamento do projeto, uma vez que com elas o gerente pode, por exemplo, ter subsídios para estimar custo e tempo associados ao desenvolvimento do produto, por meio do impacto das mudanças, favorecendo a mudança racional nos requisitos. O Indicador de Estabilidade permite avaliar como evoluiu a inserção, a alteração e a remoção de requisitos em qualquer período do desenvolvimento, indicando a qualidade do processo de levantamento, especificação e validação de requisitos. Os *links* de rastreabilidade entre os requisitos e o Modelo de Casos de Uso ajudam a garantir a consistência entre esses artefatos, valendo-se ainda do indicativo de que requisitos merecem ou não serem analisados. As consultas indicam quais requisitos estão sob gerência de determinada pessoa ou qual *stakeholder* deve ser consultado caso um requisito sofra alterações; recuperação do histórico dos requisitos.

No próximo capítulo serão apresentadas as conclusões e contribuições deste trabalho, bem como propostas de trabalhos futuros.

Capítulo 7

Conclusões

Este trabalho apresentou uma versão inicial do ambiente COCAR, desenvolvido para dar suporte ao processo de desenvolvimento de software, tendo como requisitos funcionais o registro dos requisitos de um sistema, segundo diretrizes estabelecidas no trabalho de Kawai (2005); a elaboração do Modelo de Casos de Uso, a partir desses requisitos, de acordo com a TUCCA [Belgamo, 2004]; a geração de Pontos de Casos de Uso [Martins, 2007] e o suporte ao gerenciamento de requisitos, funcionalidade essa correspondente ao foco principal deste trabalho.

Embora esta primeira versão contemple apenas essas quatro funcionalidades, o objetivo é que o ambiente seja extensível para dar suporte a várias outras atividades do processo de desenvolvimento de software que possam estar apoiadas no modelo de Casos de Uso, como é o caso da geração de casos de teste com base nesse modelo e que está, atualmente, sendo desenvolvida por outro mestrando desse grupo de pesquisa. Busca-se, dessa forma, a criação de um ambiente capaz de agregar outros trabalhos acadêmicos e que seja passível de uma utilização prática.

Como a implementação do ambiente COCAR não havia sido iniciada e todas as funcionalidades a serem disponibilizadas no ambiente trabalham com o Modelo de Casos de Uso, as diretrizes para definição de requisitos e a TUCCA foram o ponto inicial dessa implementação. Essa atividade foi compartilhada com outro mestrando que desenvolveu a geração dos Pontos de Casos de Uso [Martins, 2007]. Exatamente por isso, algumas das conclusões aqui apresentadas são similares àquelas apresentadas em Martins (2007).

Para o escopo específico deste trabalho, foram implementados os aspectos de gerenciamento de requisitos tratados com detalhes no Capítulo 5. No Capítulo 6, essas funcionalidades foram exemplificadas com telas do próprio ambiente COCAR. A primeira funcionalidade específica que tratou o gerenciamento de requisitos foi a Geração da Matriz de Rastreabilidade de Requisitos. Diversos trabalhos faziam menção à necessidade de determinação dessa matriz para a análise de impacto da mudança de um requisito sobre os demais. Neste trabalho foi proposta uma forma automática de determinação dessa matriz, baseada na análise dos dados de entrada que os requisitos manipulam. Além da determinação da dependência ou não entre os requisitos, é apresentado o percentual de dependência, também baseado no conjunto intersecção dos dados de entrada dos requisitos.

Outra funcionalidade implementada foi o Indicador de Estabilidade, proposto por Hazan e Leite (2003), com o objetivo de fornecer dados para que os responsáveis pelo desenvolvimento tenham subsídios para as tomadas de decisão. O uso de indicadores integrado com a ferramenta visa proporcionar uma maneira prática de avaliar a evolução dos requisitos, sinalizando o percentual de requisitos alterados, excluídos ou inseridos no sistema em um determinado período.

A rastreabilidade entre o documento de requisitos e o Modelo de Casos de Uso também foi disponibilizado no ambiente. A determinação dos *links* de rastreabilidade é feito de forma automática, valendo-se do formalismo da TUCCA e das diretrizes propostos por Kawai (2005). Esse formalismo é apontado como necessário por Salem (2006) para o processo de rastreabilidade. Com relação à determinação automática da rastreabilidade - tanto para a relação entre o documento de requisitos e o Modelo de Casos de Uso e para a Matriz de Rastreabilidade - Zisman e Spanoudakis (2004) afirmam ser esse o melhor caminho para as ferramentas que tratam da rastreabilidade. Apesar de Alexander (2003) defender o uso de métodos semi-automáticos, no caso do ambiente COCAR, como a associação entre um caso de uso e os requisitos que deram origem a ele é feita automaticamente, devido ao formalismo da TUCCA, o risco de ter-se o *link* gerado equivocadamente é minimizado.

Completando as funcionalidades do ambiente proposto, estão as consultas que podem ser realizadas sobre os requisitos, permitindo a pré-rastreabilidade. Para que essa funcionalidade

pudesse ser inserida no sistema, as diretrizes propostas por Kawai (2005) foram adaptadas, agregando-se os seguintes elementos: solicitante e gerente do requisito.

Todas essas funcionalidades referidas acima foram demonstradas neste trabalho por meio de um exemplo real, implementado e acompanhado com o ambiente COCAR. Esse sistema pôde fornecer idéia da aplicabilidade da ferramenta num ambiente cooperativo.

7.1 – Contribuições e limitações deste trabalho

A seguir relacionam-se, resumidamente, algumas contribuições deste trabalho:

- Criação de uma primeira versão do ambiente COCAR, para fornecer suporte a algumas atividades do desenvolvimento de software, com base em Casos de Uso, abrangendo inserção de requisitos, criação do Modelo de Casos de Uso e gerenciamento de requisitos, além de suporte para a geração de Pontos de Caso de Uso e geração de Casos de Teste baseado em Casos de Uso.
- Implementação das diretrizes para elaboração de documento de requisitos com ênfase nos requisitos funcionais propostas por Kawai (2005). A implementação da proposta favorece seu uso e avaliação.
- Implementação da TUCCA [Belgamo, 2004] para a geração do Modelo de Casos de Uso. Por ser composta de uma série de passos que compõe um algoritmo, ela pôde ser implementada, tendo ainda alguns passos automatizados. A aplicação da técnica com suporte de uma ferramenta oferece ao usuário um caminho único a ser seguido passo a passo, minimizando a possibilidade de erros e facilitando sua aplicação.
- Criação de um ambiente extensível para agregar futuras funcionalidades baseadas em Casos de Uso que possam contribuir para melhorar a qualidade do desenvolvimento de software.
- Proposta de uma forma automática para a geração da Matriz de Rastreabilidade de Requisitos, indicando-se, além do relacionamento entre os requisitos, um percentual para essa dependência. A geração automática dessa matriz facilita sua manutenção ao longo do processo de desenvolvimento de software e também favorece a mudança, de forma racional, por meio da análise de impacto da mesma.

A seguir, relacionam-se, resumidamente, algumas limitações deste trabalho:

- Apesar da proposta do ambiente COCAR ser de um ambiente que abranja algumas fases do processo de desenvolvimento de software, a ferramenta deve ser capaz de interagir com ferramentas existentes no mercado, por meio de importação/exportação de arquivos XML, por exemplo.
- O ambiente não provê, atualmente, a possibilidade de geração de relatórios impressos ou exportáveis para arquivos texto.
- Como os aspectos de rastreabilidade baseiam-se em técnicas automáticas, deve haver a possibilidade de edição, tanto da Matriz de Rastreabilidade de Requisitos como dos *links* de rastreabilidade entre os requisitos funcionais e os Casos de Uso. Dessa forma, alguma inconsistência gerada pela automatização desses processos poderia ser corrigida pelo usuário.
- Apesar de apresentar bons resultados em alguns exemplos conduzidos durante a elaboração deste trabalho, a proposta de utilizar os dados de entrada, manipulados em um requisito, para determinar a relação deste com os demais requisitos do software, a análise da Matriz de Rastreabilidade gerada merece uma maior validação por meio de experimentos práticos.
- O ambiente trata dos requisitos funcionais, uma vez que o trabalho de Kawai (2005) também foi focado nesses requisitos.

7.2 – Lições aprendidas

Durante o desenvolvimento deste trabalho de mestrado diversas lições foram aprendidas, algumas das quais estão relatadas na seqüência.

Como duas das funcionalidades do ambiente COCAR – inserção de requisitos de software e geração do Modelo de Casos de Uso – eram parte comum deste trabalho e do mestrado de Martins (2007), tanto a documentação, o projeto e a implementação dessas funcionalidades, bem como a descrição das mesmas nos trabalhos, foi realizada em conjunto. Esse trabalho em equipe exigiu organização e sinergia dos envolvidos, sendo necessária a discussão das decisões de projeto e dos aspectos de implementação. Dessa forma, o compartilhamento das experiências dos envolvidos favoreceu para a qualidade do trabalho e para a garantia do caráter extensível da

ferramenta. Entretanto, a necessidade da implementação de partes em comum ocasionou erros e atrasos no desenvolvimento. Caso a ferramenta fosse atender apenas às funcionalidades específicas deste trabalho de mestrado, sua especificação, implementação e teste teriam sido simplificados.

Ainda sobre a criação do ambiente, vale ressaltar que a especificação do mesmo foi criada utilizando-se os moldes implementados neste trabalho, ou seja, as diretrizes propostas por Kawai (2005) e disponíveis no Apêndice I. Outro artefato criado para a implementação foi o modelo de Casos de Uso gerado a partir da especificação citada anteriormente com a aplicação da TUCCA e disponível no Apêndice II. Dessa forma, procurou-se avaliar os benefícios dos trabalhos científicos que seriam incorporados no ambiente COCAR na própria criação do ambiente, além de documentar a criação do mesmo. Como não havia forma de fazer o gerenciamento de requisitos, não se pode avaliar o grau de mudanças que ocorreram durante o desenvolvimento ou mesmo prever o impacto dessas mudanças.

Outro aspecto a ser considerado como lição aprendida é o da revisão bibliográfica que, apesar de ser iniciada de forma *ad-hoc*, já no final do trabalho tomou-se conhecimento da abordagem de Revisão Sistemática ([Kitchenham, 2004] e [Biolchini et al, 2005]), fazendo-se uma aplicação da mesma para atualizar o conjunto de artigos lidos. Embora esse processo não tenha sido conduzido com o rigor exigido, conclui-se que esse tipo de revisão permite organização e sistematização do trabalho, agregando qualidade nesta etapa da dissertação.

7.3 – Trabalhos futuros

A seguir, relacionam-se os aspectos que fornecem perspectivas de continuidade deste trabalho:

- Realização de experimentos para avaliar a aplicação da TUCCA de forma manual e de forma assistida pelo ambiente.
- Realização de experimentos para analisar os resultados gerados pela Matriz de Rastreabilidade de Requisitos, avaliando-se, de forma sistemática, o impacto indicado pelo ambiente COCAR e o impacto indicado por especialistas.

-
- Permitir a criação de atributos para os *links* de rastreabilidade, conforme proposta de Marcus et al. (2005). Dessa forma, o usuário do ambiente poderia criar os atributos que achasse conveniente para atribuir aos *links*. Algumas sugestões de tipos de *links* poderiam ser feitas, seguindo as propostas da equipe de Marcus et al (2005).
 - Implementar no ambiente a possibilidade de o usuário alterar as sugestões que foram geradas automaticamente pela ferramenta, tanto para a dependência entre os requisitos na Matriz de Rastreabilidade de Requisitos, como para os *links* de rastreabilidade entre requisitos e Casos de Uso. Nesse caso, para avaliar a qualidade das sugestões feitas pela ferramenta, pode-se adaptar o indicador de corretitude, proposto por Cleland-Huang et al. (2006), gerado pela divisão do número de sugestões corretas feita pela ferramenta pelo total de sugestões feitas.
 - Avaliar a utilização do ambiente COCAR na indústria para que sejam utilizados em sistemas complexos, de diferentes portes dos utilizados neste trabalho.
 - Agregar a geração do relatório de discrepâncias proposta por Belgamo (2005) como funcionalidade do ambiente COCAR.

Referências Bibliográficas

- [Abbott,1986] ABBOTT, R.J.; An Integrated Approach to Software Development; Nova Yorque, John Wiley, 1986, 334 p.
- [Alexander, 2002] ALEXANDER, I. Towards Automatic Traceability in Industrial Practice . In: **First International Workshop on Traceability**, Edinburgh, Setembro 2003. Disponível em <http://www.soi.city.ac.uk/~zisman/WSTPapers/Alexander.pdf>. Acesso em 15/07/2006.
- [Alexander, 2003] ALEXANDER, I. SemiAutomatic Tracing of Requirement Versions to Use Cases Experiences & Challenges. In: **Second International Workshop on Traceability**, Montreal, Outubro 2003. Disponível em <http://www.soi.city.ac.uk/~gespan/paper6.pdf>. Acesso em 10/07/2006.
- [Anda & Sjoberg, 2002] Bente, Anda e Sjoberg, Dag. “Towards an Inspection Technique for Use Case Models”, In: **Fourteenth IEEE Conference on Software Engineering and Knowledge Engineering (SEKE)**, 2002.
- [Apache, 2006] Apache Software Foundation. Disponível em : <http://www.apache.org/>. Acesso em: 13/10/2006

-
- [Belgamo & Fabbri, 2004a] BELGAMO, A., FABBRI, S.C.P.F. Constructing Use Case Model by Using a Systematic Approach: Description of a Study. In: **WER – Workshop em Engenharia de Requisitos**, Tandil, Argentina, p. 251 – 262, 2004.
- [Belgamo & Fabbri, 2004b] BELGAMO, A., FABBRI, S.C.P.F. GUCCRA: Contribuindo para a Identificação de Defeitos em Documentos de Requisitos Durante a Construção de Modelos de Casos de Uso. In: **WER – Workshop em Engenharia de Requisitos**, Tandil, Argentina, p. 100 – 111, 2004.
- [Belgamo & Fabbri, 2005] BELGAMO, A., FABBRI, S.C.P.F. GUCCRA: Técnica de Leitura para apoiar a Construção de Modelos de Caso de Uso e a Análise de Documentos de Requisitos. In: **SBES – 19º Simpósio Brasileiro de Engenharia de Software**, Uberlândia, MG, 3-7 out., 2005.
- [Belgamo et al., 2005] BELGAMO, A., FABBRI, S.C.P.F., MALDONADO, J.C. Avaliando a Qualidade da Técnica GUCCRA com Técnica de Inspeção. In: **WER – Workshop em Engenharia de Requisitos**, Porto, Portugal, 13-14 jun., 2005.
- [Belgamo, 2004] BELGAMO, A. GUCCRA: Técnicas de Leitura para Construção de Modelos de Casos de Uso e Análise do Documento de Requisitos. 2004. 157 f. Tese (Mestrado em Ciências da Computação) - Universidade Federal de São Carlos, São Carlos, 2004.
- [Biolchini et al., 2005] BIOLCHINI, J.; MIAN, P. ; NATALI, A. ; TRAVASSOS, G. . Systematic Review in Software Engineering, Relatório Técnico RT - ES 679 - 05, 2005.

-
- [Castor et al.,2004] CASTOR, A; PINTO, R; SILVA, C; CASTRO, J. Towards Requirement Traceability in TROPOS, In: **WER – Workshop em Engenharia de Requisitos**, Tandil, Argentina, p. 100 – 111, 2004.
- [Cleland-Huang et al., 2006] CLELAND-HUANG, J; ZEMONT, G.; LUKASIK, W. A heterogeneous solution for improving the return on investment of requirements traceability. In: **12th IEEE International Requirements Engineering Conference (RE'04)** , Kyoto, Japão, 2005
- [Cordeiro, 2002] CORDEIRO, M. A. Uma Ferramenta Automatizada de Suporte ao Processo de Gerenciamento de Requisitos, Dissertação de Mestrado. Centro de Ciências Exatas e de Tecnologia – CCET, Pontifícia Universidade Católica do Paraná – PUCPR, Curitiba, 2002. 182p.
- [Dalal et al, 1999] DALAL, S.R. JAIN, A. KARUNANITHI, N. Leaton, J.M. Lott, C.M. Patton, G.C. Horowitz, B.M. Model-based testing in practice; In: **Proceedings of the 1999 International Conference on Software Engineering**, Los Angeles, USA, 1999.
- [Dano et al., 1997] DANO, B.; BRIAND, H.; BARBIER F. An Approach Based on the Concept of Use Case to Produce Dynamic Object-Oriented Specifications; IEEE Computer Society, Annapolis, MD, USA, p. 54-64. 1997.
- [Davis & Zowghi, 2005] DAVIS A. M ; ZOWGHI D; Good requirements practices are neither necessary nor sufficient. Requirements Engineering - Volume 11 - Springer-Verlag New York, 2005

-
- [Eclipse, 2006] Eclipse. Disponível em: <http://www.eclipse.org/>. Acesso em: 13/08/2006
- [Egyed & Grünbacher, 2002] EGYED, A.; GRÜNbacher, P. Automating Requirements Traceability: Beyond the Record & Replay Paradigm; In: **17th IEEE International Conference on Automated Software Engineering (ASE'02)**, Edinburgh, UK, 2002
- [Furlan, 1998] FURLAN, J. D. **Modelagem de Objetos através da UML**. São Paulo: Makron Books, 1998, 329 p.
- [Gotel & Finkelstein, 1997] GOTEL, O.; FINKELSTEIN, A. Extended Requirements Traceability: Results of an Industrial Case Study. In: **ISRE'97 Third International Symposium on Requirements Engineering**. 1ed. USA:, Los Alamitos, CA. Proceedings, 1997.
- [Gotel, 1996] GOTEL, O., Contribution Structures for Requirements Engineering. Ph.D Thesis. Department of Computing, Imperial College of Science, Technology, and Medicine, London, U.K., 1996.
- [Gregolin, 2006] GREGOLIN, B.T. Uma Proposta de Ferramenta para a Geração de Casos de Teste a partir de Casos de Uso. 83 f. (Exame de Qualificação de Mestrado em Ciências da Computação) - Universidade Federal de São Carlos, São Carlos, 2006.

-
- [Hammer, 1997] HAMMER, T.; ROSENBERG, L.; HUFFMAN, L.; HYATT, L. Requirement Metrics – Value Added. In: **ISRE'97 Third International Symposium on Requirements Engineering**. 1ed. USA: IEEE Comput. Soc, Los Alamitos, CA. Proceedings, 1997.
- [Hazan & Leite, 2003] Hazan, C.; Leite, J.C.S.P. Indicadores para a Gerência de Requisitos. In: **WER - Workshop em Engenharia de Requisitos**; Piracicaba, Brasil, 2003.
- [IEEE, 1990] IEEE, Institute of Electrical and Electronics Engineers. Standard Computer Dictionary: A Compilation of IEEE Standard Computer, Glossaries. New York, NY: 1990.
- [IEEE, 1998] IEEE Standard Glossary of Software Engineering Terminology, Nova Iorque: IEEE, ANSI/IEEE Std 729.1983, 1983.
- [IncoSE, 2005] INCOSE Requirements Management Tools Survey, Disponível em:
<http://www.paper-review.com/tools/rms/read.php>. Acesso em: 12/03/2005
- [Jacobson et al., 1992] JACOBSON, I. **Object-Oriented Software Engineering - A Use Case Driven Approach**, Addison-Wesley Publish Company, 1992, 528 p.

-
- [Java, 2006] Java - Core J2EE Pattern Catalog. Disponível em: <http://java.sun.com/blueprints/corej2eepatterns>. Acessado em: 30/10/2006.
- [Jones, 1994] JONES, T. C. Assessment and Control of Software Risks. Prentice Hall, 1994. 464 p
- [Kawai, 2005] KAWAI, K.K. Diretrizes para elaboração de Documento de Requisitos com ênfase nos Requisitos Funcionais. 2005. 170 f. Tese (Mestrado em Ciências da Computação) - Universidade Federal de São Carlos, São Carlos, 2005.
- [Kelleher, 2005] KELLEHER, J. A reusable traceability framework using patterns. In: **Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering- Automated Software Engineering**. Long Beach, USA, 2005.
- [Kitchenham, 2004] KITCHENHAM, B Procedures for Performing Systematic Reviews, Joint Technical Report, Keele University TR/SE-0401 e NICTA 0400011T.1, 2004
- [Kulak & Guiney, 2000] KULAK, D.; GUINEY, E. **Use Cases: Requirements in Context**. Addison-Wesley, 2000. 329 p.

-
- [Lam, 1999] LAM, W.; LOOMES, M.; SHANKARARAMAN, V. Managing Requirements Change Using Metrics and Action Planning. In: **Third European Conference on Software Maintenance and Reengineering**, London, U.K., 1999.
- [Leite, 1994] LEITE. Engenharia de Requisitos: Notas de Aula – Parte IV, PUC-Rio, 1994.
- [Letelier, 2002] LETELIER, P. A Framework for Requirements Traceability in UML-based Projects. In: **1st International Workshop on Traceability in Emerging Forms of Software Engineering**, Edinburgh, U.K, p32-41, 2002
- [Lott et al, 2005] LOTT, C.; JAIN, A.; DALAL, S.R. Modeling requirements for combinatorial software testing; In: **Proceedings of the first international workshop on Advances in model-based testing - International Conference on Software Engineering**, St. Louis, USA, 2005.
- [Marcus et al. , 2005] MARCUS A.; XIE, X.; POSHYVANYK, D. When and how to visualize traceability links? In: **Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering- Automated Software Engineering**, Long Beach, USA, 2005
- [Martins, 2007] MARTINS, M.D.C. Geração De Pontos De Casos De Uso No Ambiente Scorpion. 2007. 180 f. Tese (Mestrado em Ciências da Computação) - Universidade Federal de São Carlos, São Carlos, 2007. Aguardando defesa.

-
- [Munson & Nguyen, 2005] MUNSON, E. V.; NGUYEN, T. N. Concordance, conformance, versions, and traceability; In: **Proceedings of the 3rd international workshop on Traceability in emerging forms of software engineering**, Long Beach, California, 2005.
- [Pinheiro, 1996] PINHEIRO, F.A.C.; GOGUEN, J. A. An Object Oriented Tool for Tracing Requirements. In : **2nd International Conference on Requirements Engineering (ICRE '96)**. Colorado Springs, Colorado, USA, IEEE Comput. Soc, 1996.
- [Pinheiro, 1996a] PINHEIRO, F. A. C. Design of a Hyper-Environment for Tracing Object-Oriented Requirements. Tese de Doutorado - Department of Computing, University of Oxford, Oxford, U.K., 1996.
- [Pohl, 1996] POHL, K. Process-Centered Requirements Engineering. Advanced Software Development Series, John Wiley and Sons Ltd., 1996.
- [Power & Moyniahn, 2003] POWER, N.; MOYNIHAN, T. A Theory of Requirements Documentation Situated in Practice. In: **21st annual international conference on Documentation**, San Francisco, CA, USA, 2003
- [Pressman, 2006] PRESSMAN, R, S. **Engenharia de Software**. 6a. edição. USA. McGrawHill, 2006, 720 p.
- [Ramesh & Jarke, 2001] RAMESH, B.; JARKE, M. Towards Reference Models For Requirements Traceability, IEEE Transactions on Software Eng., vol. 27, pp. 58-93, Jan. 2001

-
- [Rational , 2006] Rational Rose. Disponível em: <http://www-306.ibm.com/software/rational/>. Acesso em: 28/08/2006
- [Rocha, 2001] ROCHA, A.C. ; MALDONADO, J.C.; WEBER, K.C. **Qualidade de Software – teoria e prática**. Prentice Hall, 1a. edição, p.320, 2001.
- [Ryan, 98] [47] RYAN, K. Requirements Engineering – getting value for money. In : **SBES'98, XII Simpósio Brasileiro de Engenharia de Software**, Maringá, Paraná. 1ed. Brasil: SBC Sociedade Brasileira de Computação, 1998.
- [Ryser & Glinz, 1999] RYSER, J.; GLINZ, M. SCENT: A Method Employing Scenarios to Systematically Derive Test Cases for System Test. University of Zurich, Insitut für Informatik, Zürich, 1999a. Disponível em: <http://www.ifi.unizh.ch/groups/req/ftp/SCENT/SCENT.pdf>. Acesso em: 07/11/2004
- [Ryser & Glinz, 2000] RYSER, J.; GLINZ, M. Using Dependency Charts to Improve Scenario-Based Testing. In: **17th International Conference on Testing Computer Software – TCS'2000**, Washington, D.C., 2000

-
- [Salem,2006] SALEM, A.M. Improving Software Quality through Requirements Traceability Models; In: **The 4th ACS/IEEE International Conference on Computer Systems and Applications (AICCSA 2006)**, Dubai, Sharjah, UAE, 2006.
- [Sarkar, 2005] SARKAR, A. Requirement Management in Testing, - Infosys Technologies. Disponível em : http://www.softwareiodioxide.com/Channels/events/testing2001/Proceedings/anindita_infosys.pdf - Acesso em: 12/02/2005
- [Schneider & Winters, 2001] SCHNEIDER, G.; WINTERS, J. P. Applying Use Cases, A Practical Guide. Second Edition, Addison-Wesley, 2001. 245 p.
- [SEI, 2002] Artigo produzido pela SEI (Software Engineering Institute), pelo grupo CMMI Product Team. CMMISM for Systems Engineering, Software Engineering, Integrated Product and Process Development, and Supplier Sourcing. Disponível em: http://seir.sei.cmu.edu/GDMforCMMI/CMMI_HTML_Files%5CREQM.htm#sp131 . Acesso em: 21/11/2004
- [Serpro,2002] SERPRO. Processo SERPRO de Desenvolvimento de Soluções (PSDS). 2002 . Disponível em : <http://www.serpro.gov.br/publicacao/tematec/2002/ttec60> Acesso em: 04/02/2005
- [Sommerville, 1997] [37] SOMMERVILLE, I. **Requirements Engineering – A good practice guide**, John Wiley, 1997.

-
- [Sommerville, 2003] SOMMERVILLE, I. **Engenharia de Software**. 6a. edição. – São Paulo - Addison Wesley, 2003, 580 p.
- [Standish Group, 1994] The Chaos Report – 1994 - Standish Group. Disponível em:
http://www.standishgroup.com/sample_research/chaos_1994_2.php. Acesso em: 12/04/2005
- [Standish Group, 1999] The Chaos Report – 1999 - Standish Group. Disponível em:
http://www.standishgroup.com/sample_research/PDFpages/chaos1999.pdf . Acesso em: 12/04/2005
- [Standish Group, 2004a] The Chaos Report – 2004 - Standish Group. Disponível em:
<http://www.stsc.hill.af.mil/crosstalk/2004/10/0410Jacobs.pdf>. Acesso em: 12/04/2005
- [Standish Group, 2004b] 2004 Third Quarter Research Report - Standish Group. Disponível em:
http://www.standishgroup.com/sample_research/PDFpages/q3-spotlight.pdf. Acesso em: 12/04/2005
- [Sun, 2006a] SUN. Disponível em:
<http://www.sun.com/java/everywhere/#facts>.
Acessado em: 15/11/2006
- [Sun, 2006b] SUN. Disponível em:
<http://www.sun.com/java/everywhere/#awards>
Acessado em: 15/11/2006

-
- [SWEBOK, 2004] IEEE Computer Society Professional Practices Committee. Guide to the Software Engineering Body of Knowledge SWEBOK. Los Alamitos, California: BOURQUE, P., DUPUIS, R., 2004. 202 p.
- [Toranzo et al., 2002] TORANZO, M.; CASTRO, J.F.B.; MELLO, E. Uma Proposta para Melhorar o Rastreamento de Requisitos; WER
- [UML, 2003] UML. OMG Unified Modeling Language Specification. Versão 1.5, março, 2003, 736 p. Disponível em: <http://www.omg.org/technology/documents/formal/uml.htm>. Acesso em: 26/11/2004.
- [Wieggers, 1999a] WIEGERS, K. **Software Requirements**, Microsoft Press, 1999. 1ª edição.
- [Wieggers, 1999b] WIEGERS, K. Automating Requirements Management. Disponível em: http://www.processimpact.com/articles/rm_tools.html. Acesso em: 21/04/2005

[Zisman & Spanoudakis,
2004]

ZISMAN, A.; SPANOUDAKIS, G. Software Traceability: Past, Present, and Future. In : **The Newsletter of the Requirements Engineering Specialist Group of the British Computer Society** – Disponível em : <http://www.resg.org.uk/archive/rq33.pdf>. Acesso em: 05/03/2005

[Zisman et al., 2003]

ZISMAN, A.; SPANOUDAKIS, G.; PEREZ-MIÑANA, E.; KRAUSE, P. Tracing Software Requirements Artifacts. In: **The 2003 International Conference on Software Engineering Research and Practice**, Las Vegas, EUA, 2003

Apêndice I

Exemplos de Requisitos Funcionais do Ambiente COCAR

Neste apêndice apresentam-se exemplos de alguns requisitos funcionais do documento de requisitos do ambiente COCAR. Ressalta-se que esse documento foi elaborado de acordo com as diretrizes de Kawai (2005). Por ser um documento extenso, para cada uma das funcionalidades contempladas no ambiente, selecionaram alguns exemplos para serem apresentados.

Exemplo de requisito do módulo de Inserção dos Requisitos de um Sistema

Requisito: 001

Descrição: Cadastrar Sistema

Agente Fornecedor/Receptor: Responsável

Agente Executor: Responsável

Entrada:

nro	campo	formatação	restrição	descrição
1	Nome do sistema	String	64 caracteres	Nome do sistema
2	Descrição do sistema	String	512 caracteres	Descrição do sistema
3	Cliente	String	Deve estar num conjunto de clientes cadastrados	Cliente que solicitou o sistema
4	Data de cadastro	Data	Formato dd/mm/aaaa	Data que foi realizado o cadastro
5	Responsável pelo sistema	String	Deve estar num conjunto de responsáveis cadastrados	Responsável pelo sistema
6	Funções do Produto	String	4096 caracteres	Funções que o sistema deve apresentar

Processamento: O Responsável deve solicitar à ferramenta a inclusão de um novo Sistema. O Responsável deve preencher todos os campos de entrada exibidos na interface e mostrados no quadro de entradas (Nome do sistema, Descrição do sistema, Cliente, Data de cadastro, Responsável pelo sistema, Funções do Produto). O Responsável salva os dados. A ferramenta deve atribuir, automaticamente, um número de identificação auto-incremental ao sistema. A qualquer momento o Responsável pode escolher a opção de Cancelar o cadastro de um Sistema. Quando isso ocorrer, deve ser exibida a interface inicial da ferramenta.

Condição/Restrição: O campo “nome do sistema” deve ser único.

Saída: Dados do Sistema salvos no banco de dados da ferramenta.

Solicitante do Requisito: André

Gerente do Requisito: André

Data: 28/11/2005

Exemplo de requisito do módulo de Inserção dos Requisitos de um Sistema

Requisito: 005

Descrição: Cadastrar Requisito

Agente Fornecedor/Receptor: Responsável

Agente Executor: Responsável

Entrada:

nro	campo	formatação	restrição	descrição
1	Descrição	String	256 caracteres	Descrição do requisito
2	Agente Fornecedor/Receptor	String	Devem estar num conjunto de agentes cadastrados	Agentes fornecedor/receptor envolvidos no requisito
3	Agente Executor	String	Devem estar num conjunto de agentes cadastrados	Agentes executores envolvidos no requisito
4	Entrada	String	Devem estar num conjunto de Tipos de Entrada cadastrados	Conjunto de dados de entrada do requisito que está

				sendo especificado
5	Processamento	String	4096 caracteres	Explicação do requisito como uma sequência de passos
6	Condição/Restrição	String	4096 caracteres	Restrições do requisito
7	Saída	String	4096 caracteres + conjunto de dados cadastrados no conjunto de dados cadastrados	Qualquer dado de saída, incluindo mensagens, textos ou dados produzidos e armazenados no sistema.
8	Stakeholder	String	Devem estar num conjunto de <i>stakeholders</i> cadastrados	Quem propôs o requisito
9	Data	Data	Formato dd/mm/aaaa	Data de entrada do requisito na ferramenta
10	Responsável	String	Deve estar num conjunto de responsáveis cadastrados	Responsável pelo cadastramento do requisito

Processamento: A ferramenta lista os sistemas cadastrados conforme o requisito 002. O Responsável solicita a inclusão de um novo Requisito para o sistema escolhido. O Requisito representa uma funcionalidade que o Sistema deve oferecer. O Responsável deve preencher todos os dados apresentados no quadro de entradas. Para salvar os dados do requisito o Responsável deve escolher a opção Salvar. A ferramenta deve atribuir, automaticamente, dois números ao requisito: um número de identificação auto-incremental e um número de versão. A qualquer momento o Responsável pode escolher a opção de Cancelar o cadastro de um Requisito no sistema. Quando isso ocorrer, deve ser exibida a interface inicial da ferramenta.

Condição/Restrição: Requisito 002 concluído com sucesso. O campo “descrição” deve ser único.

Saída: Dados do requisito salvos no banco de dados da ferramenta.

Solicitante do Requisito: André

Gerente do Requisito: André

Data: 28/11/2005

Exemplo de requisito do módulo de Elaboração do Modelo de Casos de Uso

Requisito: 029

Descrição: Marcar candidatos a atores, funcionalidades e restrições

Agente Fornecedor/Receptor: Responsável

Agente Executor: Responsável

Entrada:

nro	campo	formatação	restrição	descrição
1	Ator	String	4096 caracteres	Conjunto de atores do requisito ou das funções do produto
2	Função	String	4096 caracteres	Conjunto de Funções do requisito ou das funções do produto
3	Restrição	String	4096 caracteres	Conjunto de Restrições do requisito ou das funções do produto

Processamento: A ferramenta deve exibir os sistemas cadastrados para que o Responsável selecione um deles. O Responsável solicita a aplicação da técnica AGRT para o sistema escolhido. A ferramenta faz sugestões de atores, funcionalidades e restrições, de acordo com os campos Agente Fornecedor/Receptor, Descrição e Condição/Restrição de cada requisito cadastrado no sistema, respectivamente. O Responsável ou aceita cada sugestão ou a edita de acordo com a necessidade. Também é exibida a seção “Funções do produto” do sistema para que o Responsável identifique candidatos a atores, funcionalidades e restrições nesta seção.

Condição/Restrição: ao menos um sistema esteja cadastrado.

Saída: requisitos e seção “Funções do produto” com candidatos a atores, funcionalidades e restrições marcados.

Solicitante do Requisito: Marcos

Gerente do Requisito: Marcos

Data: 24/01/2006

Exemplo de requisito do módulo de Geração de Pontos de Casos de Uso

Requisito: 047

Descrição: Gerar medida de Pontos de Casos de Uso não ajustados para um sistema.

Agente Fornecedor/Receptor: Responsável

Agente Executor: Responsável

Entrada:

nro	campo	formatação	restrição	descrição
1	Tipo de complexidade de cada ator do sistema	Inteiro	1 caracter para cada ator	Classificar os atores como simples (1) médio (2) ou complexo (3)

Processamento: A ferramenta lista os sistemas cadastrados conforme o requisito 002, sendo que a opção Tem diagrama de casos de uso associado é marcada verdadeiro e não pode ser alterada. O Responsável escolhe um sistema e confirma a geração da medida de pontos de casos de uso. O Responsável atribui um tipo de complexidade para cada ator do sistema, ao qual é atribuído um fator de peso, de acordo com a tabela.

Tipo de ator	Descrição	Fator de peso
Simple	Outro sistema interagindo através interface com aplicação definida	1
Médio	Outro sistema interagindo através de protocolo ou linha de comando	2
Complexo	Uma pessoa interagindo através de uma interface gráfica ou página na internet	3

Para cada caso de uso, a ferramenta conta os passos dos cursos normal e alternativos, que representam as transações em cada caso de uso, e então o caso de uso é classificado de acordo com o número de transações e recebe um fator de peso, de acordo com a tabela.

Tipo de caso de uso	Descrição	Fator de peso
Simple	Até 3 transações (até 5 objetos lógicos)	5
Médio	4 a 7 transações (6 a 10 objetos lógicos)	10
Complexo	Mais que 7 transações (mais que 10 objetos lógicos)	15

A medida de pontos de casos de uso não ajustados é obtida pela fórmula $PCU(na) = \sum(PesosDosAtores) + \sum(PesosDosCasosDeUso)$.

Condição/Restrição: Requisito 002 concluído com sucesso.

Saída: Medida de pontos de casos de uso não ajustados.

Solicitante do Requisito: Marcos

Gerente do Requisito: Marcos

Data: 08/12/2005

Exemplo de requisito do módulo de Suporte ao Gerenciamento de Requisitos

Requisito: 051

Descrição: Visualizar Indicador de Estabilidade

Agente Fornecedor/Receptor: Responsável

Agente Executor: Responsável

Entrada:

nro	campo	formatação	restrição	descrição
1	Data inicial	Data	Formato dd/mm/aaaa	Data de início do período de busca para gerar o indicador
2	Data final	Data	Formato dd/mm/aaaa	Data de final do período de busca para gerar o indicador

Processamento: A ferramenta lista os sistemas cadastrados conforme o requisito 002. O Responsável escolhe um sistema e solicita a visualização do Indicador de Estabilidade. Além disso ele deve fornecer o período (data inicial e data final) para o qual deve ser gerado o indicador. O sistema deve exibir os valores do Indicador de Estabilidade, descrevendo:

- Qual o percentual de novos requisitos no período: para a determinação desse indicador deve-se buscar o número de requisitos que existiam na data inicial do período e o número de requisitos que existiam na data final do período. Subtraindo-se o número de requisitos que existiam na data final do período pelo número de requisitos que existiam na data inicial do período obtém-se o número de requisitos que foram inseridos no período. Para a determinação do percentual de novos requisitos, deve-se dividir esse valor obtido pelo número de requisitos da data inicial do período.
- Qual o percentual de requisitos alterados no período: para a determinação desse indicador deve-se buscar o número de requisitos que foram alterados entre o período

compreendido entre a data inicial e data final fornecidas. Para a determinação do percentual de requisitos alterados, deve-se dividir esse valor obtido pelo número de requisitos da data final do período.

- Qual o percentual de requisitos excluídos no período: para a determinação desse indicador deve-se buscar o número de requisitos que foram excluídos entre o período compreendido entre a data inicial e a data final fornecidas. Para a determinação do percentual de requisitos excluídos, deve-se dividir esse valor obtido pelo número de requisitos da data final do período.

Condição/Restrição: Requisito 002 concluído com sucesso.

Saída:

nro	campo	formatação	restrição	descrição
1	Requisitos novos	Float	Valores percentuais com 2 casas decimais	Percentual de requisitos novos no período especificado.
2	Requisitos alterados	Float	Valores percentuais com 2 casas decimais	Percentual de requisitos alterados no período especificado.
3	Requisitos Excluídos	Float	Valores percentuais com 2 casas decimais	Percentual de requisitos excluídos no período especificado.

Solicitante do Requisito: André

Gerente do Requisito: André

Data: 17/12/2005

Apêndice II

Exemplos de Especificação de Caso de Uso do Ambiente COCAR

Na seqüência são disponibilizadas algumas especificações de Casos de Uso do ambiente COCAR construídas a partir da aplicação da técnica TUCCA no documento de requisitos apresentado no Apêndice I.

Exemplo de Especificação de Caso de Uso de Inserção de um Sistema

Especificação do Caso de Uso		Número:	01
Nome do Caso de Uso	Cadastrar sistema		
Descrição ou Resumo	Cadastrar sistema na ferramenta		
Ator Participante	Responsável		
Ator Operador	Responsável		
Ator Genérico	-		
Pré-condição	-		
Curso Normal	<ol style="list-style-type: none"> 1. Ferramenta exibe campos para preenchimento 2. Responsável preenche todos os campos e salva 3. Ferramenta confirma que o nome do sistema ainda não existe 4. Ferramenta atribui um número auto-incremental para o sistema e salva no banco de dados 		
Curso Alternativo	<ol style="list-style-type: none"> 2a. Responsável não preenche todos os campos <ol style="list-style-type: none"> 2a1. Ferramenta informa que é necessário preencher todos os campos 2a2. Voltar ao passo 2 2b. Responsável cancela cadastro <ol style="list-style-type: none"> 2b1. Ferramenta exibe tela inicial 3a. Ferramenta confirma que o nome do sistema já existe <ol style="list-style-type: none"> 3a1. Ferramenta informa que deve ser utilizado outro nome do sistema 3a2. Voltar ao passo 2 		
Evento Disparador	O Responsável seleciona a opção		

Include	
Extend	-
Requisitos Funcionais	001
Requisitos Não funcionais	-

Exemplo de Especificação de Caso de Uso de Inserção dos Requisitos de um Sistema

Especificação do Caso de Uso		Número:	03
Nome do Caso de Uso	Cadastrar requisito		
Descrição ou Resumo	Cadastrar requisito para um determinado sistema		
Ator Participante	Responsável		
Ator Operador	Responsável		
Ator Genérico	-		
Pré-condição	-		
Curso Normal	<ol style="list-style-type: none"> 1. Include: Listar sistemas 2. Responsável seleciona um sistema no qual deseja cadastrar um requisito 3. Ferramenta exibe campos para preenchimento 4. Responsável preenche todos os campos e salva 5. Ferramenta confirma que a descrição do requisito ainda não existe 6. Ferramenta atribui um número auto-incremental para o sistema e o número de versão 1 e salva no banco de dados 		
Curso Alternativo	<ol style="list-style-type: none"> 2a. Responsável cancela alteração <ol style="list-style-type: none"> 2a1. Ferramenta exibe tela inicial 4a. Responsável cancela alteração <ol style="list-style-type: none"> 4a1. Ferramenta exibe tela inicial 5a. Ferramenta confirma que a nova descrição do requisito já existe <ol style="list-style-type: none"> 5a1. Ferramenta informa que deve ser utilizada outra descrição do requisito 5a2. Voltar ao passo 4 		
Evento Disparador	O Responsável seleciona a opção		
Include			
Extend	-		
Requisitos Funcionais	005		
Requisitos Não funcionais	-		

Exemplo de Especificação de Caso de Uso de Aplicação da AGRT

Especificação do Caso de Uso		Número:	09
Nome do Caso de Uso	Aplicar AGRT		
Descrição ou Resumo	Aplicar AGRT num determinado sistema		
Ator Participante	Responsável		
Ator Operador	Responsável		
Ator Genérico	-		
Pré-condição	Deve haver ao menos um sistema cadastrado na ferramenta		
Curso Normal	<ol style="list-style-type: none"> 1. Ferramenta exibe uma lista selecionável com todos os sistemas cadastrados 2. Responsável seleciona um sistema para aplicar a AGRT 3. Ferramenta exibe requisito e faz sugestões de ator, funcionalidade e restrição 4. Responsável confirma as sugestões feitas 5. Ferramenta inclui o ator, a funcionalidade e o número do requisito no FAO 6. Voltar ao passo 3 até que todos os requisitos tenham sido considerados 7. Ferramenta exibe seção 'Funções do Produto' do sistema 8. Responsável identifica funcionalidades na seção 'Funções do Produto' 9. Responsável associa estas funcionalidades à funcionalidades já identificadas 10. Ferramenta exibe lista de funcionalidades de um mesmo ator com possibilidade de seleção múltipla 11. Responsável seleciona funcionalidades com o mesmo significado e seleciona opção combinar 12. Responsável seleciona o termo mais apropriado para as funcionalidades 13. Ferramenta agrupa as referências das funcionalidades selecionadas numa só, utilizando o termo selecionado 14. Ferramenta substitui os termos descartados pelo termo selecionado nos requisitos referenciados pela funcionalidade 15. Responsável preenche campos da discrepância e confirma 16. Ferramenta salva discrepância 17. Voltar ao passo 11 até que não haja mais funcionalidades ambíguas 18. Voltar ao passo 10 até que todos os atores tenham sido considerados 		

	<ol style="list-style-type: none"> 19. Ferramenta exibe lista de funcionalidades dos atores, podendo associar uma letra ou um asterisco a cada funcionalidade 20. Responsável marca cada objetivo do ator 'Sistema' ou <nome do sistema> com uma letra diferente, e marca as funcionalidades com o mesmo significado com a mesma letra 21. Responsável seleciona o termo mais apropriado para as funcionalidades marcadas com uma mesma letra 22. Ferramenta substitui os termos descartados pelo termo selecionado nos requisitos referenciados pelas funcionalidades 23. Ferramenta agrupa as referências das funcionalidades marcadas com a mesma letra numa só e atribui ao ator que não 'Sistema' ou <nome do sistema>, utilizando o termo selecionado 24. Responsável preenche campos da discrepância e confirma 25. Voltar ao passo 21 até que todas as letras utilizadas para marcar as funcionalidades sejam consideradas 26. Responsável associa funcionalidade marcada com asterisco a algum ator 27. Voltar ao passo 26 até que todas as funcionalidades marcadas com asterisco tenham sido consideradas 28. Ferramenta exibe lista de funcionalidades dos atores, podendo associar um número a cada funcionalidade 29. Responsável marca funcionalidades com o mesmo significado com um mesmo número 30. Ferramenta agrupa as referências das funcionalidades selecionadas numa só, utilizando o termo selecionado 31. Responsável seleciona o termo mais apropriado para as funcionalidades marcadas com um mesmo número 32. Ferramenta substitui os termos descartados pelo termo selecionado nos requisitos referenciados pelas funcionalidades 33. Responsável preenche campos da discrepância e confirma 34. Voltar ao passo 30 até que todos os números utilizados para marcar as funcionalidades sejam considerados 35. Ferramenta verifica que todo requisito foi referenciado em alguma funcionalidade
Curso Alternativo	<ol style="list-style-type: none"> 2a. Responsável cancela aplicação da AGRT <ol style="list-style-type: none"> 2a1. Ferramenta exibe tela inicial

	<p>4a. Responsável não confirma sugestões feitas</p> <p>4a1. Responsável altera sugestões de acordo com a necessidade e confirma</p> <p>4a2. Ferramenta inclui o ator, a funcionalidade e o número do requisito no FAO</p> <p>4a3. Voltar ao passo 3</p> <p>4b. Responsável não confirma sugestões feitas</p> <p>4b1. Responsável associa requisito a uma funcionalidade já identificada</p> <p>4b2. Ferramenta inclui número do requisito na funcionalidade identificada</p> <p>4b3. Voltar ao passo 3</p> <p>4c. Responsável não confirma sugestões feitas</p> <p>4c1. Desconsiderar requisito e voltar ao passo 3</p> <p>4d. Responsável não confirma sugestões feitas</p> <p>4d1. Responsável seleciona opção nova funcionalidade</p> <p>4d2. Responsável preenche campos para nova funcionalidade e confirma</p> <p>4d3. Ferramenta inclui o ator, a funcionalidade e o número do requisito no FAO</p> <p>4d4. Voltar ao passo 3</p> <p>9a. Responsável não encontra funcionalidade para associar</p> <p>9a1. Responsável seleciona opção nova funcionalidade</p> <p>9a2. Responsável preenche campos para nova funcionalidade e confirma</p> <p>9a3. Ferramenta inclui o ator, a funcionalidade e a seção 'Funções do Produto' no FAO</p> <p>9a4. Responsável preenche campos da discrepância e confirma</p> <p>9a5. Ferramenta salva discrepância</p> <p>9a6. Voltar ao passo 9</p> <p>11a. Não há funcionalidades com o mesmo significado</p> <p>11a1. Ir para o passo 19</p> <p>20a. Não há funcionalidades com o mesmo significado de uma funcionalidade do ator 'Sistema' ou <nome do sistema></p> <p>20a1. Responsável marca funcionalidade com um asterisco</p> <p>20a2. Voltar ao passo 20</p> <p>26a. Funcionalidade não pode ser associada a nenhum ator</p> <p>26a1. Ferramenta move a funcionalidade para a lista de objetivos não-associados</p> <p>26a2. Voltar ao passo 26</p> <p>29a. Não há funcionalidades com o mesmo significado</p> <p>29a1. Ir para o passo 35</p>
--	--

	35a. Ferramenta verifica que um requisito não foi referenciado em nenhuma funcionalidade 35a1. Responsável preenche campos da discrepância e confirma 35a2. Voltar ao passo 35 e checar demais requisitos
Evento Disparador	O Responsável seleciona a opção
Include	
Extend	-
Requisitos Funcionais	029, 030, 031, 032, 033, 034, 035
Requisitos Não funcionais	-

Exemplo de Especificação de Caso de Uso de Geração de PCU

Especificação do Caso de Uso		Número:	10
Nome do Caso de Uso	Gerar medida de PCU		
Descrição ou Resumo	Gerar medida de pontos de casos de uso para um determinado sistema		
Ator Participante	Responsável		
Ator Operador	Responsável		
Ator Genérico	-		
Pré-condição	-		
Curso Normal	<ol style="list-style-type: none"> 1. Include: Listar sistemas (com a opção 'Tem diagrama de casos de uso associado' marcada verdadeiro) 2. Responsável seleciona um sistema 3. Ferramenta exibe a lista de atores do sistema selecionado 4. Responsável atribui peso a cada um dos atores 5. Ferramenta conta os passos dos cursos normal e alternativos de cada caso de uso do sistema e atribui peso a cada um deles 6. Ferramenta calcula pontos de casos de uso não ajustados para o sistema 7. Ferramenta exibe o questionário de Fator de Complexidade Técnica 8. Responsável atribui pontuação a cada um dos fatores do questionário 9. Ferramenta exibe o questionário de Fator Ambiental 10. Responsável atribui pontuação a cada um dos fatores do questionário 11. Ferramenta calcula os pontos de casos de uso ajustados para o sistema 12. Ferramenta transforma a medida de pontos de casos de uso em medida de pontos por função 13. Ferramenta exibe as medidas de pontos de casos de uso ajustados e a medida de pontos por função do sistema 		

Curso Alternativo	<p>2a. Responsável cancela geração de PCU 2a1. Ferramenta exibe tela inicial</p> <p>4a. Responsável não atribui peso a todos os atores 4a1. Ferramenta informa que o responsável deve atribuir peso a todos os atores do sistema 4a2. Voltar ao passo 4</p> <p>8a. Responsável não atribui pontuação a todos os fatores do questionário 8a1. Ferramenta informa que o responsável deve atribuir pontuação a todos os fatores do questionário 8a2. Voltar ao passo 8</p> <p>10a. Responsável não atribui pontuação a todos os fatores do questionário 10a1. Ferramenta informa que o responsável deve atribuir pontuação a todos os fatores do questionário 10a2. Voltar ao passo 10</p>
Evento Disparador	O Responsável seleciona a opção
Include	
Extend	-
Requisitos Funcionais	047, 048, 049
Requisitos Não Funcionais	-

Exemplo de Especificação de Caso de Uso de Indicador de Rastreabilidade

Especificação do Caso de Uso		Número:	11
Nome do Caso de Uso	Visualizar indicador de estabilidade		
Descrição ou Resumo	Visualizar indicador de estabilidade para um determinado sistema		
Ator Participante	Responsável		
Ator Operador	Responsável		
Ator Genérico	-		
Pré-condição	-		
Curso Normal	<ol style="list-style-type: none"> 1. Include: Listar sistemas 2. Responsável seleciona sistema cujo indicador de estabilidade deseja visualizar 3. Ferramenta exibe campos para data inicial e data final 4. Responsável informa data inicial e data final do período a ser calculado o indicador de estabilidade 5. Ferramenta calcula o indicador de estabilidade do sistema selecionado para o período informado 6. Ferramenta exibe o indicador de estabilidade do sistema no período informado 		
Curso Alternativo	2a. Responsável cancela visualização do indicador de estabilidade		

	2a1. Ferramenta exibe tela inicial 4a. Responsável não informa data inicial e/ou data final 4a1. Ferramenta informa que o responsável deve informar a data inicial e a data final do período 4a2. Voltar ao passo 4
Evento Disparador	O Responsável seleciona a opção
Include	
Extend	-
Requisitos Funcionais	051
Requisitos Não Funcionais	-