UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DISSERTAÇÃO DE MESTRADO

GERAÇÃO DE CENÁRIOS DE TESTE COM BASE EM CASOS DE USO

FÁBIO ROBERTO OCTAVIANO

São Carlos/SP

Janeiro/2011

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DISSERTAÇÃO DE MESTRADO

GERAÇÃO DE CENÁRIOS DE TESTE COM BASE EM CASOS DE USO

Dissertação 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, área de concentração: Engenharia de Software.
Orientadora: Dra. Sandra Carmargo Pinto Ferraz Fabbri

São Carlos/SP

Janeiro/2011

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária da UFSCar

O21gc

Octaviano, Fábio Roberto.
Geração de cenários de teste com base em casos de uso / Fábio Roberto Octaviano. -- São Carlos : UFSCar, 2011.
131 f.

Dissertação (Mestrado) -- Universidade Federal de São Carlos, 2011.

1. Validação, verificação e teste. 2. UML (Linguagem de modelagem unificada). I. Título.

CDD: 005.14 (20ª)

Universidade Federal de São Carlos

Centro de Ciências Exatas e de Tecnologia

Programa de Pós-Graduação em Ciência da Computação

"Geração de Cenários de Teste com Base em Casos de Uso"

FÁBIO ROBERTO OCTAVIANO

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:

Profa. Dra. Sandra C. Pinto Ferraz Fabbri
(Orientadora - DC/UFSCar)

Prof. Dr. Valter Vieira de Camargo
(DC/UFSCar)

Profa. Dra. Silvia Regina Vergilio
(UFPR)

São Carlos
Fevereiro/2011

DEDICATÓRIA

A toda a minha família, amigos e em especial à minha esposa,
pela força, ajuda e compreensão de vocês em todos os momentos.

AGRADECIMENTOS

Primeiramente, e acima de tudo, agradeço a Deus por permitir a realização deste trabalho.

À minha orientadora, Profª. Drª. Sandra C. P. F. Fabbri, pessoa que admiro muito, pela confiança e constante ajuda na orientação, além da paciência e dedicação.

Em especial, aos meus pais Clóvis e Wayni, que sempre me apoiaram nessa caminhada. Aos meus irmãos Paulo e Carlos, que são pessoas muito queridas e sempre estão ao meu lado. À minha esposa Cassiana, pela força e paciência, além da compreensão nos inúmeros momentos de ausência que este trabalho exigiu-me.

Ao grande amigo André, pela colaboração, força e incentivo nos momentos difíceis, além do precioso auxílio técnico em relação ao ambiente COCAR.

Ao pessoal da Comunidade Hesed, pela amizade.

À comissão da PPG-CC e à secretária Cristina, que colaborou sempre que necessário.

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

RESUMO

Cerca de 37% dos casos de insucesso em projetos de software têm sua causa relacionada a problemas com a coleta e manipulação dos requisitos do software. Atividades de Gerência de Requisitos ajudam a melhorar esse quadro, mas é preciso validar constantemente o software sendo produzido quanto à sua fidelidade aos requisitos. Nesse sentido, as atividades de Garantia de Qualidade de Software (GQS) são muito importantes para encontrar defeitos no software sendo desenvolvido, antes mesmo que haja uma versão executável do produto. As principais atividades de GQS são Teste e Inspeção e, apesar do Teste se aplicar, geralmente, quando se está na fase de codificação, essa atividade pode e deve se iniciar o quanto antes, pois ela precisa de planejamento para ser bem sucedida. Além disso, muitos dos Cenários e Casos de Teste relevantes são determinados logo no início do desenvolvimento, ainda em fase de levantamento e modelagem dos requisitos. Assim, considerando esse contexto e considerando também que uma das técnicas mais usadas para a modelagem dos requisitos é o Modelo de Casos de Uso, este trabalho teve por objetivo implementar um módulo para geração de Cenários de Teste a partir de Casos de Uso. Esse módulo corresponde a uma funcionalidade adicional no ambiente COCAR, o qual apóia outras atividades do desenvolvimento de software com base nos modelos de casos de uso. O módulo implementado no ambiente contribuiu para a automatização da atividade de Teste com a geração de cenários de teste unitários, que se referem aos casos de uso individualmente, e cenários de teste de integração, que exercitam as relações de dependência existentes entre os casos de uso de um sistema, sempre com base em suas especificações. A avaliação do trabalho foi feita com base em um sistema real que já estava desenvolvido e para o qual foram comparados os cenários de teste usados quando ele foi desenvolvido e os cenários gerados pelo ambiente COCAR. Os resultados mostraram que os cenários gerados contemplam quase que totalmente os cenários definidos manualmente e, portanto, podem ser usados como suporte à definição do plano de teste e aos casos de teste.

ABSTRACT

Around 37% of unsuccessfully software projects have their cause related to issues on how to software requirements are collected and manipulated. Requirement Management activities help on getting this scenario better, however it is necessary to validate the software under construction on its right meaning frequently. Thus, Software Quality Assurance (SQA) activities are very important to find defects on softwares under development, even before having an executable product version. The main SQA activities are Test and Inspection and, in spite of Test activity generally occurs during coding phase, this activity may and should be applied as soon as possible, as it needs planning to achieve its goal. Besides, many important Test Cases and Scenarios are specified once software development starts, during requirement specification and modeling phases. Thus, considering this scenario and also that Use Case Model is one of the most used techniques of requirement modeling, this work has the objective of implementing a module to generate Test Scenarios based on Use Cases. The module is a new added functionality to COCAR environment, which supports other software development activities based on use case model. This new module helps on Test activity automation, generating unity testing scenarios, related to each use case, and integration testing scenarios, that use the relationships among use cases, based on their specifications. The work evaluation was done based on real developed software, comparing the used test cases with the test scenarios generated by COCAR environment. The results have shown that the generated test scenarios have almost all of the used test cases matching. Thus, they can be used to support test plan and test cases definition.

LISTA DE FIGURAS

Figura 1: Quantidade de defeitos encontrados com e sem o processo de Inspeção (adaptada de [Wheeler, 1996]) 20
Figura 2: Derivação de Cenário para Leitura Baseada em Perspectiva 22
Figura 3: Generalização de Atores 33
Figura 4: Interação entre Atores e Casos de Uso 34
Figura 5: Exemplo de associação com o estereótipo <<extend>> 35
Figura 6: Exemplo de associação com o estereótipo <<include>> 36
Figura 7: Especificação do Caso de Uso "Alugar Carro" 38
Figura 8: Técnicas de leitura que compõem a TUCCA 40
Figura 9: Descrição do Sistema de Biblioteca 44
Figura 10: Diagrama de Dependência para o exemplo da Biblioteca 44
Figura 11: Diagrama de Atividades dos casos de uso para o Sistema de Biblioteca (adaptada de [Briand & Labiche, 2002]) 47
Figura 12: Grafo correspondente ao Diagrama de Atividades da Figura (adaptada de [Briand & Labiche, 2002]) 48
Figura 13: Árvore derivada do grafo da Figura 12 (adaptada de [Briand & Labiche, 2002]) 49
Figura 14: Diagrama da ferramenta UCT (adaptada de [Carniello, 2003b]) 53
Figura 15: Abordagem para melhoria de teste de sistema (adaptado de [Hartmann et al., 2005]) 55
Figura 16: Metodologia de 2 fases para geração de cenários de teste (adaptado de [Nebut et al., 2006]) 57
Figura 17: Processo de aplicação da TUCCA, com a marcação de atores, funções e restrições (adaptado de [Thommazo, 2007]) 61
Figura 18: Relatório de geração de Pontos de Casos de Uso (adaptado de [Martins, 2007]) 62
Figura 19: Exemplo de visualização da rastreabilidade entre requisito e Caso de Uso (adaptado de [Thommazo, 2007]) 63
Figura 20: Tela de Geração de Casos de Teste Unitários e de Integração 70
Figura 21: Exemplo de Geração de Cenário de Teste Unitário 72
Figura 22: Grafo de Dependência de Casos de Uso 73
Figura 23: Exemplo de Cenário de Integração Gerado 74
Figura 24: Opções do Menu para acesso ao cadastro manual de Caso de Uso 75
Figura 25: Tela para cadastro manual de Caso de Uso 76
Figura 26: Tela de Cadastro de Curso Normal do Caso de Uso 78
Figura 27: Tela de Cadastro de Curso Alternativo do Caso de Uso 79
Figura 28: Tela de Modelagem do Caso de Uso na ferramenta ArgoUML 82
Figura 29: Exemplo de arquivo XMI em que foram seguidas as regras estabelecidas para integração da ArgoUML com o COCAR 83
Figura 30: Tela de Importação de Caso de Uso por Arquivo XMI 84
Figura 31: Cenários de Teste Unitários gerados pelo COCAR para o Caso de Uso “Realizar Atendimento” 89
Figura 32: Cenários de Teste de Integração gerados pelo COCAR para os Casos de Uso registrados no ambiente 90
Figura 33: Parte da especificação do Caso de Uso “Visualizar Atendimento” que foi inserido manualmente no COCAR 92
Figura 34: Tela do COCAR usada para inserção manual de Caso de Uso 93
Figura 35: Tela do COCAR para inserção do curso normal do Caso de Uso 94
Figura 36: Tela do COCAR para inserção do curso alternativo do Caso de Uso 95
Figura 37: Arquivo XMI do Caso de Uso “Gerar Relatório” 96
Figura 38: Tela do COCAR usada para importação de casos de uso por meio de arquivo XMI 97
Figura 39: Exemplo de Documentação dos Cursos do Caso de Uso “Realizar Atendimento” 98
Figura 40: Grafo dos cursos normal e alternativos do Caso de Uso “Realizar Atendimento” 100
Figura 41: Exemplo de Caso de Teste extraído do Plano de Teste da Empresa X para o Caso de Uso “Realizar Atendimento” 101
Figura 42: Grafo de dependência dos casos de uso utilizados no estudo de caso 107
Figura 43: Cenários de teste de integração dos casos de uso utilizados no estudo de caso 108

LISTA DE TABELAS

Tabela 1: Histórico da conclusão de projetos de software 13
Tabela 2: Principais fatores de insucesso de um projeto de software 14
Tabela 3: Comparação dos Cenários de Teste Unitários e dos Casos de Teste para o Caso de Uso "Realizar Atendimento" 102
Tabela 4: Comparação dos cenários de teste unitários e dos casos de teste utilizados no estudo de caso 103
Tabela 5: Relação entre os cenários de teste de integração gerados pelo COCAR e utilizados no plano de teste da Empresa X 109

SUMÁRIO

Capítulo 1 - Introdução *12*
1.1. Contexto 12
1.2. Motivação e Objetivos 15
1.3. Organização do Trabalho 16
Capítulo 2 – Atividades de VV&T *18*
2.1 Considerações Iniciais 18
2.2. Inspeção de Software 19
2.3. Teste de Software 24
2.4. Considerações Finais 29
Capítulo 3 – Modelagem de Requisitos com Casos de Uso *30*
3.1 Considerações Iniciais 30
3.2 Casos de Uso 31
3.2.1. Diagrama de Casos de Uso 32
3.2.2. Especificação de Casos de Uso 36
3.3. A Técnica de Inspeção TUCCA 39
3.4. Considerações Finais 41
Capítulo 4 – VV&T e Casos de Uso *42*
4.1 Considerações Iniciais 42
4.2. Método SCENT 43
4.3. Uma Abordagem Baseada em UML para Teste de Sistema 47
4.4. Teste Baseado na Estrutura de Casos de Uso 50
4.5. Teste de Sistema com base na UML 54
4.6. Automação da Atividade de Teste com base em Modelos UML 55
4.7. Geração Automática de Teste com base em Casos de Uso 56
4.8. Considerações Finais 57
Capítulo 5 – O Ambiente COCAR *59*
5.1. Considerações Iniciais 59
5.2. O ambiente COCAR 59
5.3. Considerações Sobre a Implementação da Versão Inicial 64
5.4. Considerações Finais 65
Capítulo 6 – Cenátios de Teste com base em Casos de Uso no Ambiente COCAR *67*
6.1. Considerações Iniciais 67
6.2. Geração de Cenários de Teste com base em Casos de Uso 68
6.2.1. Cenários de Teste Unitário 70
6.2.2. Cenários de Teste de Integração 72
6.3 Módulo de Inserção Manual de Casos de Uso 74
6.3.1. Cadastrar Caso de Uso 75
6.3.2. Cadastrar Curso Normal 77
6.3.3. Cenários de Teste de Integração 78
6.4. Integração do COCAR com Outras Ferramentas 79
6.5. Considerações Finais 85
Capítulo 7 – Estudo de Caso *86*
7.1. Considerações Iniciais 86
7.2. Como Gerar Cenários de Teste no Ambiente COCAR 88
7.2.1. Geração de Cenários de Teste Unitários 88
7.2.2. Geração de Cenários de Teste de Integração 89
7.3. Inserindo no Ambiente COCAR Casos de Uso desenvolvidos externamente a ele 91
7.3.1. Inserção Manual do Caso de Uso 91
7.3.2. Importação de Caso de Uso a partir de XMI 95
7.4. Análise de Resultados 97
7.5. Considerações Finais 111
Capítulo 8 - Conclusões *112*
8.1. Contribuições e limitações deste trabalho 113
8.2. Lições Aprendidas 115
8.3. Trabalhos Futuros 115

Referências *117*
Anexo I *125*

CAPÍTULO 1

INTRODUÇÃO

1.1. Contexto

O produto de software tem assumido as mais diversificadas funções, que vão desde prover entretenimento até o gerenciamento de sistemas críticos, garantindo a preservação de patrimônios e de vidas humanas. Por causa de todas essas atribuições que são conferidas a um software, a garantia da qualidade desse produto torna-se uma atividade cada vez mais crítica. A qualidade de um produto pode ser percebida de diversas maneiras: pela combinação dos elementos visuais utilizados na interface do software, pela facilidade de sua utilização ou ainda pela quantidade de erros que apresenta, para citar alguns exemplos.

Dessa forma, torna-se essencial que um conjunto de atividades de apoio, denominado Garantia de Qualidade de Software, ocorra ao longo de todo o processo de desenvolvimento de software. Essas atividades se resumem, basicamente, a atividades de teste e inspeção que têm por finalidade avaliar os diversos artefatos de software que são elaborados ao longo do processo, como por exemplo, o documento de requisitos, os modelos de análise e projeto, o plano de teste, entre outros.

Os documentos relacionados aos requisitos merecem uma atenção especial, pois durante a fase de Engenharia de Requisitos são enfrentadas as maiores dificuldades de um processo de desenvolvimento de software; conseqüentemente, a maior parte dos defeitos tem origem nessa fase [Standish Group, 2006]. Duas grandes causas para essa geração de defeitos são: a dificuldade que os engenheiros de software têm para compreender os requisitos apresentados pelos usuários e a incerteza a respeito dos requisitos desejados por parte dos próprios usuários. Os defeitos então gerados, quanto mais demoram a ser tratados, mais

provocam desvios de orçamento e prazo, além de gerar defeitos no produto final, podendo levar até ao cancelamento do projeto.

Pesquisas do instituto Standish Group [Standish Group, 2006] mostram que no ano de 2006, 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, foi de apenas 35%. Mesmo sendo um número baixo, representa uma melhora com relação aos anos anteriores. A Tabela 1, com dados do mesmo instituto, ilustra essa realidade.

Tabela 1: Histórico da conclusão de projetos de software
[Standish Group, 2006; Standish Group, 2004; Standish Group, 2000]

	Situação dos projetos de softwares		
Ano	**Bem Sucedido**	**Falho**	**Problemático**
2006	35 %	19 %	46 %
2004	29 %	18 %	53 %
2000	28 %	23 %	49 %
1998	26 %	28 %	46 %
1996	27 %	40 %	33 %
1994	16 %	31 %	53 %

Nessa tabela, um projeto Falho é aquele que foi interrompido antes de ser concluído ou que tenha sido entregue ao usuário, mas nunca tenha sido utilizado; um projeto Problemático, por sua vez, é aquele que não respeitou prazo, custo, e/ou tenha sido entregue sem alguma característica solicitada pelo cliente. Como um complemento à Tabela 1, apresenta-se a Tabela 2, que aponta os fatores que mais colaboram para o insucesso de um projeto, sustentando a problemática relacionada à Engenharia de Requisitos, citada anteriormente.

A partir da Tabela 2, pode-se perceber que os fatores determinantes para o sucesso de um projeto de desenvolvimento de software estão intimamente relacionados com a habilidade de compreender as necessidades do usuário e garantir que o produto que está sendo desenvolvido seja adequado aos seus desejos.

Tabela 2: Principais fatores de insucesso 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%

Nesse sentido, a compreensão dos requisitos do usuário torna-se um fator crítico para o sucesso de um Projeto de Software, assim como a correta transcrição desses requisitos no Documento de Requisitos e, posteriormente, em algum modelo, como por exemplo, o modelo de caso de uso. Por isso, é importante que existam atividades de validação dos artefatos gerados nessa fase, como pode ser observado em alguns trabalhos já propostos na literatura, que utilizam Técnicas de Leitura para realizar essa tarefa.

Um exemplo dessas técnicas é a TUCCA (Technique for Use Case Construction and Construction-Based Requirements Analysis) [Belgamo, 2004a; Belgamo, 2004b; Belgamo, 2005a; Belgamo 2005b], que tem por objetivo principal auxiliar na elaboração de Modelos de Casos de Uso, fazendo com que essa atividade fique mais sistematizada e independente da experiência do desenvolvedor. Dado que essa técnica se mostrou bastante efetiva tanto para avaliar o Documento de Requisitos como para ajudar na construção de Modelos de Caso de Uso mais padronizados, trabalhos anteriores de mestrado, do mesmo grupo de pesquisa, implementaram o ambiente COCAR [Martins, 2007; Thommazo, 2007], que dá suporte

automatizado à TUCCA, além de apoiar outras etapas do desenvolvimento de software: o rastreamento dos requisitos e a geração de pontos de casos de uso.

Considerando a importância das atividades de Garantia da Qualidade de Software, em particular das atividades de VV&T (Validação, Verificação e Teste), e considerando que na fase de Engenharia de Requisitos vários cenários de teste podem ser definidos para serem usados posteriormente, pode-se observar na literatura diversos trabalhos que procuram contribuir nessa área, como os trabalhos de Ryser & Glinz (2000), de Briand & Labiche (2002), de Carniello (2003), de Hartmann et al. (2005), de Vieira et al. (2006) e de Nebut et al. (2006). Todos eles têm por objetivo gerar cenários de teste com base no modelo de caso de uso e foram usados como referência para o trabalho aqui apresentado.

1.2. Motivação e Objetivos

Dado o contexto apresentado anteriormente, a grande motivação deste trabalho é contribuir com a validação dos requisitos do usuário e dar continuidade ao desenvolvimento do ambiente COCAR, fruto de trabalhos de mestrado anteriores. Essa contribuição está relacionada à atividade de Teste que é uma das atividades de Garantia de Qualidade de Software (GQS), disponibilizando nesse ambiente recursos de geração de cenários de teste com base em casos de uso, os quais podem ser usados como subsídios para a elaboração do plano de teste e da definição de casos de teste.

Assim, o principal objetivo deste trabalho é combinar as propostas de alguns autores [Ryser & Glinz, 2000; Briand & Labiche, 2002; Carniello, 2003; Hartmann et al., 2005; Vieira et al., 2006; Nebut et al., 2006] e colocar na prática um suporte automatizado, com base nesses estudos, tendo como fruto a implementação de um módulo no ambiente COCAR para a geração automatizada de cenários de teste com base em casos de uso, procurando com isso um aumento de qualidade na atividade de Teste, uma vez que o propósito é gerar cenários de teste ainda na fase de Engenharia de Requisitos, conforme comentado na seção anterior.

O objetivo secundário é permitir que o ambiente COCAR possa ser utilizado independentemente da aplicação da técnica TUCCA, possibilitando que sistemas já desenvolvidos possam fazer uso dos recursos de teste. Nesse sentido, o objetivo é tornar possível que sistemas já desenvolvidos e cuja documentação esteja em formato manual ou em

formatos gerados por ferramenta de modelagem, possam utilizar o ambiente para gerar cenários de teste.

1.3. Organização do Trabalho

Este trabalho está dividido em sete capítulos, da seguinte forma:

Neste Capítulo 1 apresentou-se o contexto em que o trabalho está inserido, o qual é uma área bastante importante da engenharia de software, pois trata da atividade de teste com base nas fases iniciais de desenvolvimento, mais particularmente, com base na modelagem dos requisitos por meio de casos de uso.

O Capítulo 2 comenta a importância das atividades de verificação, validação e testes no contexto da Garantia da Qualidade de Software, explorando o processo de inspeção com a apresentação das técnicas de leitura PBR e OORTs; ainda nesse capítulo, são apresentados os conceitos e técnicas de teste.

O Capítulo 3 apresenta a técnica para modelagem de requisitos com casos de uso, uma vez que a principal proposta deste trabalho é a geração automatizada de cenários de teste com base em casos de uso.

O Capítulo 4 mostra as atividades de VV&T para casos de uso, dando uma visão geral sobre a geração de cenários e casos de teste a partir de casos de uso e são comentados os trabalhos relevantes para este trabalho por possuírem contribuições interessantes nessa área.

O Capítulo 5 apresenta o ambiente COCAR, descrevendo seu surgimento e os trabalhos de mestrado que contribuíram para seu desenvolvimento.

O Capítulo 6 apresenta os frutos deste trabalho de mestrado, comentando sobre cada uma das funcionalidades implementadas no ambiente COCAR.

No Capítulo 7 é apresentado um estudo de caso para análise e verificação dos resultados obtidos comparando cenários de teste gerados automaticamente no ambiente com documentações reais de casos de uso e de casos de teste fornecidas para esta pesquisa.

O Capítulo 8 apresenta a conclusão deste trabalho, dando ênfase às vantagens, desvantagens e limitações do mesmo.

O trabalho ainda contém um anexo (Anexo A) que contém exemplos de documentos fornecidos pela empresa para viabilização do estudo de caso.

CAPÍTULO 2

Atividades de VV&T

O objetivo deste capítulo é apresentar os principais conceitos sobre teste e inspeção de software, que são duas atividades de garantia de qualidade de software. A importância da atividade de inspeção para o contexto do trabalho se dá pelo fato de o ambiente COCAR ter sido construído com base na técnica de leitura TUCCA (Technique for Use Case Construction and Construction-Based Requirements Analysis). A importância da atividade de teste se dá pelo fato da proposta deste trabalho estar direcionada justamente para a geração de casos de teste com base em casos de uso.

2.1 Considerações Iniciais

A Garantia da Qualidade de Software (GQS) pode ser definida como a conformidade de requisitos funcionais e de desempenho explicitamente declarados, padrões de desenvolvimento explicitamente documentados, e características implícitas que são esperadas de todo software desenvolvido profissionalmente [Pressman, 2006]. Cada vez mais a GQS vem sendo considerada para possibilitar maior qualidade aos produtos que são desenvolvidos. Nesse sentido, mais recentemente, grande ênfase tem sido dada aos modelos de qualidade de processo, como por exemplo, o CMMI (Capability Maturity Model Integration), no qual a Garantia da Qualidade de Software é pré-requisito para que a empresa saia do nível 1, considerado caótico, e passe para o nível 2, no qual é dado início à melhoria do processo.

Com a crescente evolução e demanda para construção de software, várias pesquisas vêm sendo realizadas para garantir a qualidade do software desenvolvido. As atividades de Garantia da Qualidade de Software se resumem, basicamente, em atividades de Verificação, Validação e Testes (VV&T).

As atividades de VV&T são conceituadas da seguinte forma:

- Verificação refere-se ao conjunto de atividades que garante que o software implementa corretamente uma função específica [Pressman, 2006].

- Validação refere-se a um conjunto de atividades diferente que garante que o software construído corresponde aos requisitos do cliente [Pressman, 2006].
- Teste refere-se ao processo de execução de um produto para verificar se esse satisfaz os requisitos especificados ou para identificar as diferenças entre resultados esperados e reais [IEEE, 1990].

Com relação à atividade de Teste, três conceitos importantes e freqüentemente confundidos são conceituados pela [IEEE, 1990] como:

- **Defeito:** é o passo, processo ou definição de dados incorretos, como por exemplo, uma instrução ou comando incorreto;
- **Erro:** é a diferença entre o valor obtido e o valor esperado, ou seja, qualquer estado intermediário incorreto ou resultado inesperado na execução do programa constitui um erro;
- **Falha:** é a produção de uma saída incorreta com relação à especificação.

Deve-se salientar a importância das atividades de VV&T durante todo o processo de desenvolvimento de software, garantindo assim tanto atividades de verificação e validação quanto atividades de teste, de forma sistemática [Andriole, 1986].

Para apoiar as atividades de Inspeção e Teste, várias técnicas foram propostas na literatura. Nas seções seguintes tanto as atividades como as técnicas mais relevantes ao contexto deste trabalho são comentadas. Assim, a Inspeção de Software é explorada na Seção 2.2 e a Atividade de Teste na Seção 2.3. Na Seção 2.4 apresentam-se as Considerações Finais.

2.2. Inspeção de Software

O processo de Inspeção de software foi criado em 1972 por Fagan, para a IBM, com o objetivo de melhorar a qualidade de software e aumentar a produtividade dos programadores. Trata-se de um método de análise estática para verificar se os produtos gerados pelo processo de desenvolvimento de software satisfazem o cliente/usuário [Fagan, 1976; Fagan, 1986].

O processo de Inspeção parte do princípio que a detecção de defeitos logo nas primeiras fases de desenvolvimento do software pode garantir um menor tempo gasto em re-trabalho, assegurando um software que tenha uma maior probabilidade de atender aos

requisitos do usuário, de ser entregue no prazo estabelecido, de cumprir a estimativa de custo estipulada, etc. A Figura 1 apresenta os dados de uma situação relatada em [Wheeler et al., 1996], na qual os números entre parênteses representam a quantidade de defeitos passados para a próxima fase sem o processo de Inspeção e os números fora dos parênteses mostram a quantidade de defeitos passados para a próxima fase utilizando-se Inspeção. De acordo com esse exemplo, pode-se perceber que, quando se usou o processo de Inspeção, o re-trabalho deve ter sido reduzido, pois a quantidade de defeitos ao longo do processo também foi reduzida.

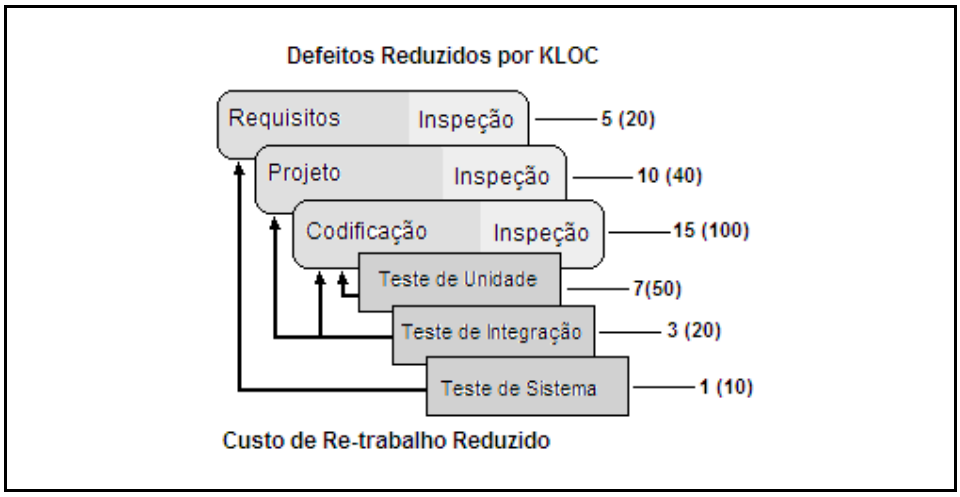


Figura 1: Quantidade de defeitos encontrados com e sem o processo de Inspeção (adaptada de [Wheeler, 1996])

Embora o custo de implantação e execução do processo de Inspeção seja alto, o custo de re-trabalho necessário tende a diminuir, devido ao fato dos defeitos serem encontrados logo no início do processo de desenvolvimento de software. Já a não utilização do processo de Inspeção resulta em mais re-trabalho o que faz com que o custo de reparo do defeito detectado seja de 10 a 100 vezes mais alto e, além disso, isso acontece em fases mais adiantadas do processo de desenvolvimento [Fagan, 1976].

O processo de Inspeção pode ser aplicado nos diversos artefatos desenvolvidos, como por exemplo, especificações de requisitos, documentação, planos e casos de teste, melhorando a qualidade do produto final (software). Segundo Fagan, o fato de a Inspeção poder ser aplicada logo no início do processo de desenvolvimento contribui muito para a qualidade

desse processo, pois é na etapa inicial que é injetada a maior quantidade de defeitos [Fagan, 1976; Fagan, 1986].

O mesmo não acontece com a atividade de teste, pois como ela implica na execução do produto, os defeitos só são detectados após sua codificação. No entanto, a Inspeção também pode ser aplicada durante a elaboração dos planos de testes, melhorando a eficiência na detecção de defeitos, uma vez que algumas possibilidades que nunca seriam testadas são identificadas nessa atividade de inspeção dos planos de teste [Fagan,1986].

Ressalta-se que as atividades de Inspeção não substituem as atividades de teste; elas devem ser combinadas. Quando a Inspeção é combinada com testes, a detecção de defeitos é muito mais ampla porque a Inspeção tende a encontrar erros difíceis de serem descobertos com testes [Russel, 1991].

Como foi mencionado anteriormente, durante o processo de Inspeção, utilizam-se algumas técnicas que auxiliam na detecção dos defeitos. Essas técnicas são técnicas de leitura que fornecem diretrizes de como os inspetores devem direcionar sua leitura para que mais defeitos possam ser encontrados. A seguir será realizada uma breve revisão sobre técnicas de leitura, comentando de forma mais detalhada as duas técnicas de leitura que foram utilizadas como base para o trabalho de Belgamo (2004a), que definiu a técnica TUCCA, implementada no ambiente COCAR.

Técnicas de Leitura

Os vários documentos de trabalho gerados durante o desenvolvimento de software (por exemplo, requisitos, projeto, código e plano de testes) freqüentemente exigem entendimento, revisão e modificação contínua ao longo do processo de desenvolvimento. Dessa forma, a leitura de software é uma atividade chave em muitas tarefas de Engenharia de Software – verificação, validação, manutenção, evolução e reuso [Basili et al., 1996a].

As técnicas de leitura PBR (Perspective Based Reading) e OORTs/ProDeS (Object Oriented Reading para o Processo ProDes) adotam procedimentos específicos e bem definidos para se realizar uma leitura de um documento. A PBR apóia a leitura de documentos de requisitos. A OORTs/ProDeS apóia a leitura de diagramas UML. Essas duas técnicas foram utilizadas na definição da técnica TUCCA, que está implementada no ambiente COCAR, e que dá suporte à elaboração de modelos de casos de uso.

- **Técnica de Leitura Baseada em Perspectiva: PBR – Perspective Based Reading**

A PBR é considerada uma técnica de leitura baseada em cenários, os quais denotam um procedimento que o leitor de um documento de requisitos deve seguir durante o processo de inspeção e pode ser caracterizada como uma fusão de questões e modelos [Runeson et al., 1999]. A Figura 2 mostra a relação entre as questões e os modelos, formando cenários que são utilizados na revisão do documento.

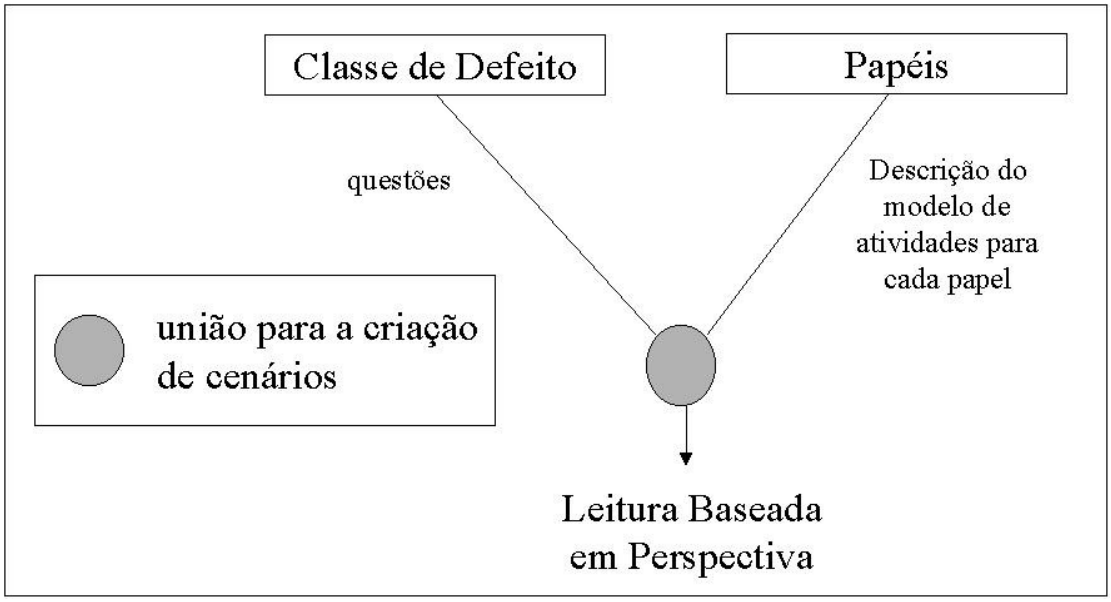


Figura 2: Derivação de Cenário para Leitura Baseada em Perspectiva (adaptado de [Basili et al., 1998])

Considerando que o documento de requisitos é fundamental para certas classes de pessoas que participam do desenvolvimento do software, as perspectivas inicialmente definidas para PBR foram do usuário, do projetista e do testador. Com isso, a idéia é que cada indivíduo se concentre em encontrar defeitos relacionados somente a um desses papéis. Ao fazer isso, a classe de defeitos encontrada é mais específica daquela perspectiva em particular e, com o uso da PBR, a pessoa tem as diretrizes necessárias para se concentrar no papel específico que lhe foi atribuído.

Vale salientar que a PBR pode ser adaptada para cada organização. Essa característica se deve ao fato de que a PBR, com o uso de suas perspectivas, trabalha com procedimentos bem definidos, os quais podem ser instanciados para o contexto da organização. Com esses procedimentos, a experiência pode ser repassada a novos revisores por meio de treinamento [Shull et al., 2000].

- **Técnicas de Leitura Orientada a Objetos: OORTs/ProDes – Object Oriented Reading Techniques para o Processo ProDes**

As técnicas OORTs/ProDeS correspondem a uma extensão das técnicas OORTs. Estas, por sua vez, foram propostas por Travassos et al (1999a) para leitura de artefatos produzidos segundo o paradigma orientado a objetos e foram definidas para serem usadas com um subconjunto da notação UML (diagrama de casos de uso, diagrama de classes acompanhado de sua descrição, diagramas de estados e diagramas de seqüência) [Rational, 1999] versão 1.3 padronizada pela OMG. [Travassos et al., 1999a; Travassos et al., 1999b; Travassos et al., 2000; Travassos et al., 2002].

As OORTs compreendem sete técnicas de leitura divididas em Leitura Horizontal e Leitura Vertical. Na Leitura Horizontal, os artefatos comparados pertencem à mesma fase do projeto, enquanto que na Leitura Vertical, os documentos do projeto são comparados com o documento de requisitos e casos de uso, que pertencem à fase inicial do projeto.

Cada documento que é inspecionado é avaliado quanto a dois aspectos:

- Aspecto Sintático: são feitas marcações de conceitos a serem observados nos artefatos que estão sendo inspecionados,
- Aspecto Semântico: mediante os conceitos marcados no aspecto sintático, os passos das técnicas ajudam o leitor a procurar pelas discrepâncias nos artefatos.

Diante do fato de que as OORTs, propostas por Travassos et al., são técnicas utilizadas para avaliar documentos UML da fase de projeto do processo de desenvolvimento de software e que a notação UML não possui um processo associado a ela, de forma a padronizar quais devem ser os documentos UML elaborados na fase de projeto e como esses documentos devem ser elaborados, Marucci propôs algumas técnicas de leitura baseadas nas OORTs, considerando a utilização de um processo de desenvolvimento bem definido [Marucci, 2002a; Marucci et al., 2002b]. Esse processo, denominado ProDeS/UML, foi proposto por Colanzi (1999) e está dividido em quatro fases: Engenharia de Requisitos, Análise, Projeto e Implementação. Além disso, para definir uma estratégia de inspeção para o ProDes/UML, Marucci considerou também a abordagem de VV&T proposta por Andriole (1986), que recomenda que todo sistema deve ser desenvolvido gradativamente, de forma que ele possa

ser avaliado por etapas e que também essa avaliação contemple atividades de validação e verificação.

Foi com base na PBR-Usuário e em uma das técnicas de leitura que compõem as OORTs/ProDeS que a técnica TUCCA, proposta por Belgamo et al. (2004b), foi definida. Lembra-se que a TUCCA é uma das técnicas apoiada pelo ambiente COCAR e que, com base nos Modelos de Casos de Uso decorrentes de sua aplicação, o ambiente COCAR permite a rastreabilidade dos requisitos e a geração dos Pontos de Casos de Uso.

2.3. Teste de Software

Nesta seção são apresentados os principais conceitos sobre a atividade de teste e as técnicas que dão suporte à mesma.

Como mencionado anteriormente, a atividade de Teste também é umas das atividades de Garantia da Qualidade de Software que possui como principal objetivo executar um programa com a finalidade de encontrar erros, implicando, em geral, numa mudança de ponto de vista, já que é comum se pensar que um teste bem-sucedido é aquele no qual não são encontrados erros [Pressman, 2006].

Mesmo aplicando a atividade de teste, ainda assim não se pode garantir que o produto esteja livre de erros. Exceto em algumas circunstâncias, a única maneira de fazer essa afirmação seria com a aplicação do Teste Exaustivo, no qual o produto é testado com todas as possíveis entradas. Como esse tipo de teste é inviável na grande maioria dos casos, foram definidos técnicas e critérios de teste que ajudam o testador a definir casos de teste mais eficazes. Entende-se por caso de teste a definição de um conjunto de entradas e o conjunto de saídas esperadas. Assim diz-se que um bom caso de teste é aquele que tem alta probabilidade de encontrar um erro ainda não descoberto e que um teste bem-sucedido é aquele que descobre um erro ainda não descoberto.

A aplicação da atividade de Teste pode ser considerada como uma série de quatro passos que são executados seqüencialmente. Os passos de aplicação e seus objetivos são explicados a seguir [Pressman, 2006]:

1. **Teste de Unidade**: tem por objetivo identificar erros de lógica e implementação em cada unidade do software separadamente. Esse teste tem seu foco na menor unidade de projeto do software – o componente ou módulo de software.
2. **Teste de Integração**: é uma estratégia sistemática para construir a estrutura do programa conduzindo-se testes para descobrir erros associados às interfaces de comunicação entre as partes componentes. Essa estratégia é utilizada após o teste de unidade.
3. **Teste de Validação**: tem por objetivo validar se o software funciona de um modo que contemple as expectativas do cliente.
4. **Teste de Sistema**: é uma série de diferentes testes cuja finalidade principal é exercitar por completo o sistema, explorando problemas de desempenho, segurança, etc.

Para que esses passos sejam executados, eles são apoiados por técnicas e critérios de teste que têm por objetivo auxiliar na definição dos casos de teste. Os critérios estão agrupados em três técnicas, a Funcional, a Estrutural e a Baseada em Erros as quais são consideradas complementares pelo fato de identificarem tipos diferentes de defeitos. Além dessas técnicas que são clássicas na literatura e que foram desenvolvidas para o paradigma procedimental, existe muita pesquisa e vários critérios de teste definidos para o teste de Máquina de Estados Finitos [Chow, 1978]. A seguir, as três técnicas clássicas são descritas, bem como os principais critérios pertencentes a elas.

A. Técnica de Teste Funcional

Essa técnica é também conhecida como Teste Caixa Preta, pois o software é tratado como uma caixa preta na qual seu conteúdo é desconhecido, conhecendo-se apenas o exterior do software. O Teste Funcional permite ao Engenheiro de Software utilizar um conjunto de valores de entrada que exercite todos os aspectos funcionais do sistema [Pressman, 2006].

Um dos principais objetivos dessa técnica é mostrar que a implementação do software está de acordo com os requisitos especificados no documento de requisitos. Nesse tipo de teste, a estrutura lógica interna do sistema não é levada em consideração e sim, as funções que são identificadas no documento de requisitos [Pressman, 2006].

Para a aplicação desse tipo de teste, é necessário entender os relacionamentos entre cada parte do software sendo desenvolvido. Por isso, geralmente, o primeiro passo é desenvolver um grafo em que cada nó representa um objeto (aqui um objeto seria um módulo do sistema) e suas arestas representem as relações entre os objetos, contendo informações como peso e restrições. Então, pode-se utilizar o grafo gerado de diversas maneiras para testar o sistema, avaliando-se a forma como as informações fluem entre eles.

Além do uso de grafos, outros critérios que auxiliam na elaboração dos casos de teste podem ser utilizados, sendo que os mais conhecidos são: Particionamento em Classes de Equivalência e Análise do Valor Limite [Pressman, 2006].

- Particionamento em Classes de Equivalência

Este Teste Funcional divide o domínio de entrada em classes de dados. Seu objetivo é encontrar grupos de dados de entrada, denominados Classes de Equivalência, de tal forma que o teste do sistema para qualquer dado dentro de uma classe tenha um resultado equivalente. Assim, reduz-se o número de casos de teste que devem ser gerados para o número de Classes de Equivalência encontradas.

- Análise do Valor Limite

Por razões que não são completamente claras, um grande número de erros tende a ocorrer nas fronteiras do domínio de entrada, ao invés do "centro" [Pressman, 2006]. Por isso, é interessante construir casos de teste que avaliem os limites do domínio de entrada e é nesse sentido que o teste por Análise do Valor Limite é executado.

A Análise do Valor Limite é complementar à do Particionamento em Classes de Equivalência. Enquanto que na técnica por Classes de Equivalência é selecionado um valor aleatório dentro da classe para a realização do teste, na Análise do Valor Limite são selecionados os limites de cada Classe de Equivalência.

Outra abordagem interessante da Análise do Valor Limite é que é possível analisar os limites do domínio de saída além do domínio de entrada. Ou seja, a partir do conhecimento do domínio de saída, é possível selecionar valores de entrada que gerem resultados nos limites do domínio de saída ao final da execução dos testes.

B. Técnica de Teste Estrutural

Ao contrário da Técnica de Teste Funcional, o Teste Estrutural é baseado na estrutura interna da implementação do software, o que faz com que ele também seja chamado de Teste Caixa Branca.

A maioria dos critérios dessa técnica utiliza uma estrutura de grafo para representar o programa testado denominada grafo de fluxo de controle. Esse grafo é orientado, com um único nó de entrada e um único nó de saída. Os nós representam uma seqüência de comandos do programa enquanto que os arcos representam os desvios de controle entre as seqüências de comandos.

Os critérios dessa técnica são divididos em três categorias: Critérios Baseados na Complexidade, Critérios Baseados em Fluxo de Controle e Critérios Baseados em Fluxo de Dados.

- Critérios Baseados na Complexidade

Utilizam informações obtidas de uma medida quantitativa da complexidade lógica de um programa para derivar casos de teste. Um critério bastante conhecido é o Critério de McCabe, também chamado Caminhos Básicos, que estabelece grupo de caminhos a serem executados com base na complexidade lógica de execução de cada caminho. O critério garante que casos de teste que sejam construídos para executar este grupo base de caminhos irão executar todos os arcos existentes no programa pelo menos uma vez durante a execução do teste. A métrica de software utilizada para calcular a complexidade lógica do programa em teste, de maneira quantitativa, é chamada de complexidade ciclomática.

Esse critério inspirou decisões tomadas neste trabalho para definir os cenários de teste com base nos casos de uso, como é apresentado no Capítulo 6.

- Critérios Baseados em Fluxo de Controle

Utilizam as informações de controle da execução do programa para derivar os casos de teste. Dessa forma, cada critério tem o objetivo de exercitar por completo uma das estruturas do grafo que modela a estrutura do sistema. Os critérios mais conhecidos são: Todos os Nós, Todos os Ramos e Todos os Caminhos [Pressman, 2006].

- Critérios Baseados em Fluxo de Dados

Utilizam informações do fluxo de dados do programa para derivar os casos de teste. Eles consideram a identificação da declaração, definição e uso das variáveis. Os critérios mais conhecidos são: Potencias-Usos [Maldonado, 1991], Todas-Definições e Todos-Usos [Rapps & Weyuker, 1982].

C. Técnica de Teste Baseado em Erros

Essa técnica tem por objetivo verificar se o software está livre dos erros mais comuns cometidos durante o seu desenvolvimento. Portanto, a ênfase da técnica está nos erros que o programador pode cometer durante o desenvolvimento e nas abordagens que podem ser usadas para detectar a sua ocorrência. Os principais critérios dessa técnica são: Semeadura de Erros (*Error Seeding*) [Budd, 1981] e Análise de Mutantes (*Mutation Analysis*) [DeMillo, 1980].

O critério de Semeadura de Erros baseia-se na inserção de erros no software e avaliação de quantos desses erros foram identificados.

Já o critério de Análise de Mutantes baseia-se na inserção de alterações sintáticas do código fonte em teste, produzindo novas versões do programa, denominadas mutantes. Então, são gerados casos de teste com o objetivo de distinguir as versões mutantes da versão original. Quando isso acontece, sabe-se que o programa em teste está livre dos erros caracterizados pelas alterações sintáticas.

D. Técnica de Teste Baseado em Máquina de Estados Finitos

Esse tipo de teste tem por objetivo determinar seqüências de teste para avaliar um sistema que esteja modelado pela técnica Máquina de Estados Finitos (MEF). Alguns dos critérios dessa técnica são: W [Chow, 1978], DS [Gönenç, 1970], UIO [Sabnani & Dahbura, 1988], TT [Naito & Tsunoyama, 1981] e Wp [Fujiwara et al., 1991]. Esses critérios têm por objetivo testar algumas propriedades da MEF. Por exemplo, o critério W tem como objetivo identificar todos os erros de seqüenciamento de uma MEF. Esse critério checa somente a estrutura de controle do projeto e não exige uma especificação executável. Para que este critério seja aplicado, assim como acontece com vários outros critérios citados acima, a MEF

deve satisfazer alguns requisitos como minimalidade, determinismo, especificação completa, etc. O critério TT (*Transition Tour*), por sua vez, é o menos rigoroso deles e garante apenas que todas as possíveis seqüências de transições da MEF sejam exercitadas.

2.4. Considerações Finais

Neste capítulo foram abordados os principais conceitos sobre as atividades de garantia de qualidade de software relevantes para este trabalho, que são a atividade de inspeção e a atividade de teste.

Os conceitos de inspeção e técnicas de leituras são relevantes porque o ambiente COCAR foi criado para automatizar a técnica de leitura TUCCA, que apóia a inspeção de documentos de requisitos e a construção de casos de uso. Já os conceitos sobre as atividades de teste de software são relevantes porque o objetivo principal deste trabalho é a implementação do módulo de geração de Cenários de Teste com base em Casos de Uso.

O próximo capítulo apresenta a modelagem de requisitos com casos de uso, que é fundamental para este trabalho, uma vez que os cenários de teste são gerados com base nos casos de uso.

CAPÍTULO 3

MODELAGEM DE REQUISITOS COM CASOS DE USO

Este capítulo apresenta conceitos sobre casos de uso e sobre a técnica TUCCA, que auxilia na construção de casos de uso ao mesmo tempo em que faz uma inspeção no Documento de Requisitos. Essa técnica está implementada no ambiente COCAR, a partir do qual são gerados os cenários de teste como proposto neste trabalho.

3.1 Considerações Iniciais

Requisitos são as necessidades que motivam o usuário a buscar no software uma ferramenta de auxílio. Essas necessidades devem ser coletadas de clientes e usuários para se saber "o quê" o software a ser idealizado fará, sem se preocupar em "como" ele fará o que deve ser feito.

Com base nos requisitos, importantes cenários de teste podem ser construídos de forma a tornar o produto gerado mais confiável. Muitos desses cenários são estabelecidos após a modelagem dos requisitos em alguma técnica de modelagem, dentre as quais, o Modelo de Casos de Uso [Jacobson et al., 1992], técnica utilizada neste trabalho.

Embora muito usado, o Modelo de Casos de Uso não possui diretrizes para sua elaboração. Assim, a técnica TUCCA (*Technique for Use Case Model Construction and Construction-based Requirements Document Analysis*) [Belgamo et al., 2005a] foi criada para auxiliar na padronização de geração de casos de uso, e é base para a implementação do ambiente COCAR.

Neste capítulo explora-se o Modelo de Casos de Uso na Seção 3.2, a técnica de inspeção TUCCA na Seção 3.3 e as Considerações Finais na Seção 3.4.

3.2 Casos de Uso

Os casos de uso são representações pictóricas do documento de requisitos que têm como objetivo mostrar a funcionalidade do sistema, bem como a interação dos usuários com o mesmo. Os casos de uso possuem ainda a característica de facilidade de entendimento por parte dos usuários, contribuindo assim, no auxílio à elicitação e validação dos requisitos junto aos mesmos.

O caso de uso foi proposto por Ivar Jacobson em 1992 em seu modelo de processo denominado Objectory [Jacobson et al., 1992]. O caso de uso é utilizado hoje em dia pela UML e o modelo Objectory foi comprado pela Rational Software, posteriormente adquirida pela IBM. O Objectory, juntamente com os casos de uso, tornaram-se parte do Rational Unified Process (RUP) [Furlan, 1998; Schneider & Winters, 2001; Rational, 2009]. Os Diagramas de Caso de Uso mostrados neste trabalho são baseados na notação UML (Unified Modeling Language) mantida pelo OMG – *Object Management Group* [OMG, 2010]. Em Julho de 2005 a versão 2.0 da UML foi adotada formalmente pela OMG como versão atual.

De acordo com Jacobson et al. (1992), a abordagem dirigida ao caso de uso é importante para a elicitação e validação dos requisitos junto aos usuários, além de servir 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 são utilizados nas fases de Análise, Projeto, Implementação, Testes e Manutenção de acordo com as necessidades correspondentes a estas. Além dessas características, os casos de uso podem ser usados tanto para uma abordagem orientada a objetos quanto para qualquer outra abordagem de desenvolvimento.

Neste trabalho, o termo Modelo de Casos de Uso corresponde ao Diagrama de Casos de Uso (representação pictórica) e às Especificações de Casos de Uso que descrevem as interações que ocorrem durante a execução do Caso de Uso. A seguir, esses dois itens são comentados.

3.2.1. Diagrama de Casos de Uso

Para elaboração do Diagrama de Caso de Uso, os seguintes itens devem ser definidos [Jacobson et al., 1992]:

- **Fronteira do Sistema:** representa a linha imaginária que separa o sistema – representado por seus casos de uso – e o mundo exterior. Dessa maneira, o escopo do sistema é estabelecido tendo-se em mente as restrições de prazo e custo para a execução do projeto; auxiliam na identificação da fronteira, os atores e casos de uso que são definidos, de forma que toda essa definição ocorre de maneira iterativa. 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.
- **Ator:** especifica um papel assumido por um usuário ou qualquer sistema que realize alguma interação com o sistema. A fronteira do sistema auxilia na identificação dos atores uma vez que todo ator deve ser uma entidade externa ao sistema. É importante ter em mente que um ator não representa uma entidade física, mas sim o papel representado por ela em uma interação com o sistema; sendo assim, uma mesma entidade física pode assumir diferentes papéis ao interagir com o sistema fazendo uso de diferentes funcionalidades, ou seja, possivelmente tornando-se mais de um ator. Da mesma forma, um ator pode ser um papel assumido por mais de uma entidade física.

Uma vez identificados os atores, é possível classificá-los, de forma a categorizar e priorizar a implementação dos casos de uso de acordo com os atores que interagem com eles. Jacobson classifica os atores da seguinte forma [Jacobson et al., 1992]:

- **Atores Primários:** são aqueles que se utilizam do sistema diretamente (talvez em seus trabalhos diários). 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, pois não faz sentido existir um ator secundário se não existir um ator primário interagindo com o sistema. Por exemplo, em um Sistema de Vendas, um ator com o papel de Operador é responsável pela manutenção do sistema e

por relatórios diários. Já o ator Cliente solicita pedidos de produtos no sistema. Nessa situação, o ator primário é o cliente, já que sem ele não seria necessário, por exemplo, a manutenção do sistema e relatórios de vendas. Desse modo, não existe Operador sem Cliente.

Como citado anteriormente, essa classificação em atores primários e secundários é útil para estruturar a funcionalidade do sistema em termos das principais funcionalidades [Jacobson et al., 1992]. Ou seja, pode-se iniciar o processo de desenvolvimento do software projetando, implementado e testando os casos de uso que são mais importantes de acordo com a interação dos mesmos com os atores primários.

Além da classificação dos atores quanto a primários e secundários, é possível estabelecer uma relação de generalização de atores, classificando-os em atores genéricos e específicos, como ilustrado pela Figura 3. Esse artifício permite que interações que deveriam ser modeladas de maneira igual para os atores mais específicos possam ser modeladas uma única vez para o ator mais geral.

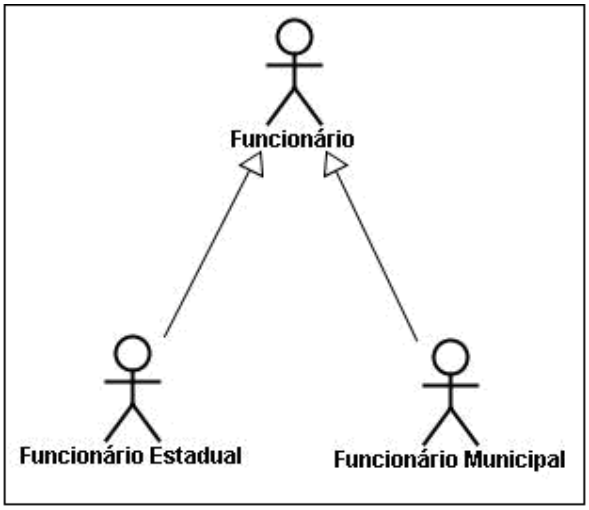


Figura 3: Generalização de Atores

- **Casos de Uso:** este item representa uma parte do comportamento provido pelo sistema. Após a definição do que está do lado de fora do sistema, pode-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 dentro ou logo abaixo da mesma, conforme pode ser observado na Figura 4.

Na Figura 4 é exibida a interação do ator primário "Cliente" com o Caso de Uso "Alugar Carro". Tal interação é representada por uma seta que leva informações do ator para o Caso de Uso e, nesse caso, por outra seta, responsável por entregar a mensagem de resposta emitida pelo Caso de Uso para o ator.

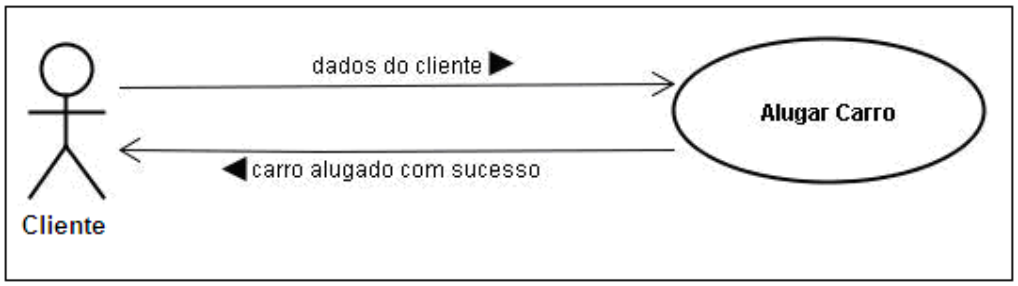


Figura 4: Interação entre Atores e Casos de Uso

Vale observar na Figura 4 que as interações entre ator e caso de uso podem conter informações explícitas. Essas informações são úteis para o entendimento de quais são as entradas para o caso de uso, bem como suas saídas.

Além das interações com atores, os casos de uso também podem interagir com outros casos de uso, utilizando associações identificadas por meio dos estereótipos <<extend>> e <<include>>.

As associações de casos de uso com o estereótipo <<extend>> são usadas para estender 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, 20001]. Jacobson menciona alguns exemplos de quando o <<extend>> é usado [Jacobson et al., 1992]:

- Para modelar partes opcionais de casos de uso, ou seja, existe algum comportamento no caso de uso que não ocorre freqüentemente; portanto, este comportamento deve ser modelado em novo caso de uso.
- Para modelar cursos complexos e alternativos, os quais raramente ocorrem.
- Para modelar sub-cursos separados, os quais são executados somente em certos casos.

Na Figura 5 é mostrada a associação com o estereótipo <<extend>> entre casos de uso. Observa-se que o "Caso de Uso Original" é estendido com a criação do "Novo Caso de Uso". O "Novo Caso de Uso" pode ou não ser usado pelo "Caso de Uso Original".

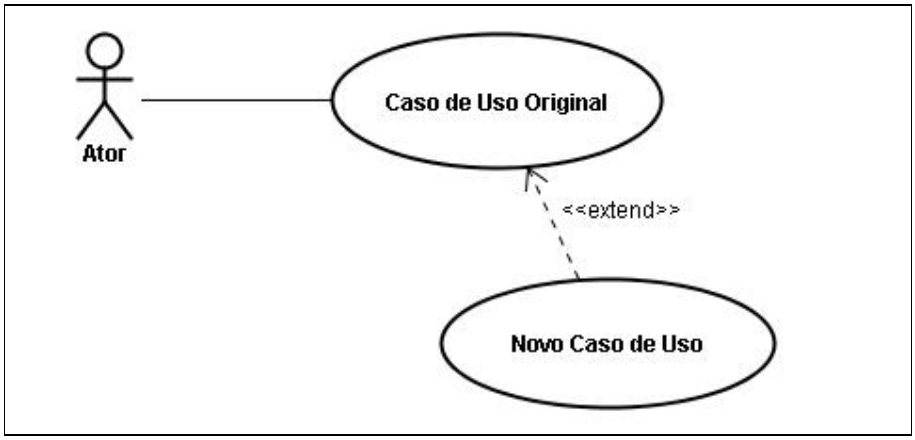


Figura 5: Exemplo de associação com o estereótipo <<extend>>

As associações de casos de uso com o estereótipo <<include>> são usadas para evitar a duplicação de passos através de múltiplos casos de uso. Esse tipo de estereótipo é empregado como uma estratégia de reuso para a funcionalidade representada pelo caso de uso. Ao invés de colocar os passos em múltiplos documentos de especificação de caso de uso, criam-se casos de uso que possuem associação com o estereótipo <<include>> [Kulak & Guiney, 2000]. Este tipo de associação garante uma melhor consistência entre os casos de uso, pois evita a possibilidade da mesma funcionalidade ser especificada de forma inconsistente. Além disso, modificações na forma de execução do caso de uso em questão teriam que ser duplicadas em todos os casos de uso que tivessem a mesma função, caso o estereótipo <<include>> não estivesse sendo utilizado, o que prejudicaria a manutenibilidade da documentação, acabando por gerar inconsistências.

A associação com o estereótipo <<include>> entre casos de uso é exibida na Figura 6. O "Caso de Uso B" foi identificado como um caso de uso com comportamento comum e o "Caso de Uso A" utiliza esse comportamento.

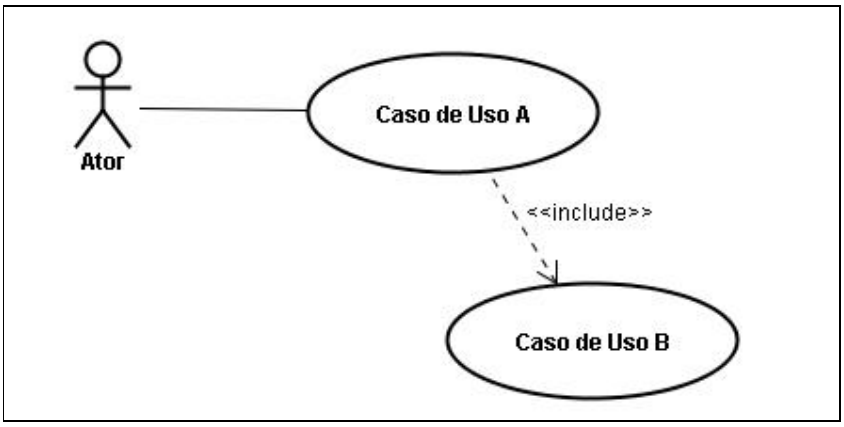


Figura 6: Exemplo de associação com o estereótipo <<include>>

Salienta-se que o sentido das setas de associações quando é usado o estereótipo <<extend>> (Figura 5) e <<include>> (Figura 6) diferem. No estereótipo <<extend>> a seta da associação "sai" no caso de uso estendido e "chega" no caso de uso que o estende, enquanto que no estereótipo <<include>> a seta da associação "sai" do caso de uso que inclui o outro e "chega" no caso de uso incluído.

3.2.2. Especificação de Casos de Uso

Identificados os principais casos de uso que compõem o sistema e desenvolvido o Diagrama de Casos de Uso, inicia-se a documentação dos mesmos. A Documentação dos Casos de Uso, ou Especificação, serve para descrever o que o caso de uso em questão faz, podendo-se utilizar linguagem natural ou alguma 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 *templates* na literatura que podem ser utilizados para a Especificação de Casos de Uso, como por exemplo, [Kulak & Guiney, 2000; Ryser & Glinz, 2000; Schneider & Winters, 2001]. Alguns desses *templates* utilizam algumas características comuns que são necessárias para o entendimento dos casos de uso, relacionadas a seguir:

- **Nome do Caso de Uso:** todo caso de uso deve possuir um nome único que o identifique.
- **Descrição ou Resumo**: serve para declarar o propósito e os 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**: nome dos atores que interagem com o caso de uso, podendo ser especificadas as responsabilidades de cada ator.
- **Pré-Condição:** condições que devem ser verdadeiras para que o caso de uso possa ser realizado.
- **Pós-Condição:** assim como a pré-condição, a pós-condição deve declarar em qual estado o sistema vai estar após a realização do caso de uso. O *template* definido em [Ryser & Glinz, 1999a] sugere a divisão da pós-condição quando um fluxo de eventos encerra-se com sucesso e quando o fluxo encerra-se em conseqüência de alguma falha.
- **Curso Normal:** é também chamando de curso básico e corresponde a um fluxo de eventos, ou seja, uma série de passos que representa o caminho correto do caso de uso de acordo com o usuário. Os passos do caso de uso devem ser numerados e a descrição dos passos deve ser curta. É importante salientar que existe somente um curso normal para cada caso de uso.
- **Curso Alternativo:** também corresponde a um fluxo de eventos, porém mostra os caminhos menos comuns que podem acontecer. Esse tipo de curso corresponde a uma seqüência diferente de eventos da que foi usada para o curso normal.
- **Caminhos de Exceção**: diferentemente de cursos alternativos, caminhos de exceção mostram as interações que ocorrem quando um erro ocorre.
- **Evento Disparador:** descreve o critério de entrada para o caso de uso sendo especificado. Um evento disparador pode descrever uma necessidade de negócio, pode estar relacionado ao tempo ou pode ser a finalização de um outro caso de uso.
- **Requisitos Não-Funcionais:** referem-se às restrições de tempo, requisitos de interface, segurança e etc.
- **Autor:** a pessoa que criou a especificação e é responsável pela sua manutenção.

- **Data:** a data da criação da especificação.
- **Versão:** corresponde a um código associado à versão da especificação, para qual pode ser utilizado para propósitos de rastreabilidade.

A Figura 7 exemplifica a descrição do caso de uso "Alugar Carro". Vale ressaltar que algumas partes (Caminhos de Exceção e Requisitos Não-Funcionais) da especificação do caso de uso em questão não estão apresentadas.

Nome do Caso de Uso	Alugar Carro
Descrição ou Resumo	O Caso de Uso tem como funcionalidade alugar 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	1. Cliente escolhe o carro desejado. 2. Cliente fornece seu RG. 3. Executar o Caso de Uso "Validar Cliente" 4. Sistema efetua o aluguel do carro.
Curso Alternativo	4. Sistema não efetua o aluguel ao cliente. Cliente com pendência financeira.
Caminhos de Exceção	não disponível
Evento Disparador	Cliente deseja alugar um carro.
Requisitos Não-Funcionais	não disponível
Autor	Fábio Roberto Octaviano
Data	18/03/2010
Versão	3

Figura 7: 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 *template* de especificação de requisitos. Conforme explicado, o curso normal é caracterizado pelos eventos que ocorrem de forma comum e natural no sistema. Por exemplo, em um sistema de aluguel de carros, pode ser considerado que no curso normal de um caso de uso "Alugar Carro" não seja necessário o cadastro do cliente, pois na maioria das vezes o cliente já está cadastrado. 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. Já os cursos alternativos de um caso de uso são variações dos cursos normais. Normalmente, cada caso de uso tem somente um curso normal e zero ou mais cursos alternativos.

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 este finaliza, é retornado ao curso normal do caso de uso original, como se nada tivesse acontecido. Na Figura 7, observa-se essa situação quando o sistema se encontra 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". Neste ponto, encerra-se a execução do caso de uso "Alugar Carro".

Para finalizar, observa-se que a especificação dos casos de uso é importante para as demais fases do processo de desenvolvimento, pois ela é usada durante a elaboração de outros diagramas, bem como para a elaboração de casos de testes.

A seguir comenta-se sobre a técnica de inspeção TUCCA, cujo principal objetivo é uma maior padronização e eficiência na geração de casos de uso, sendo de grande importância no contexto deste trabalho, uma vez que ela está implementada no ambiente COCAR.

3.3. A Técnica de Inspeção TUCCA

O trabalho de mestrado de Belgamo (2004a) propôs uma técnica de leitura, denominada TUCCA (*Technique for Use Case Model Construction and Construction-based Requirements Document Analysis*), cujo objetivo principal é auxiliar na elaboração de Modelos de Casos de Uso, fazendo com que essa atividade fique mais sistematizada e independente da experiência do desenvolvedor, como também menos suscetível a maus entendimentos que levem a 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. Ao utilizar essa técnica consegue-se gerar Modelos de Casos de Uso padronizados, atentando para a clareza e completude dos requisitos funcionais e não-funcionais que são retratados nos modelos. 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.

A TUCCA é composta de duas técnicas de leitura: AGRT – Actor Goal Reading Technique e UCRT – Use Case Reading Technique. Elas fornecem diretrizes para a elaboração do Modelo de Casos de Uso de forma que, à medida que o modelo é construído, essas diretrizes provocam uma inspeção no Documento de Requisitos, dando suporte à detecção de vários defeitos no mesmo. A Figura 8 mostra as entradas esperadas e saídas geradas pela TUCCA.

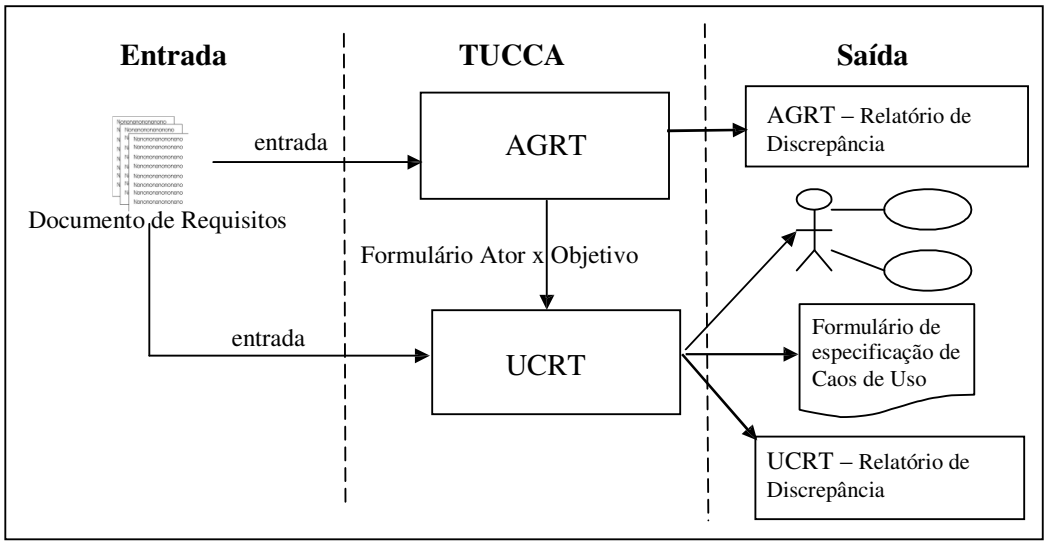


Figura 8: Técnicas de leitura que compõem a TUCCA

Como pode ser observado na Figura 8, a primeira técnica a ser aplicada é a AGRT. A entrada para essa técnica é o Documento de Requisitos (que pode ter sido inspecionado ou não), o qual deve estar no padrão IEEE [IEEE, 1998] ou possuir, ao menos, seções equivalentes às seções Definições, Funções do Produto, Características do Usuário, Requisitos Funcionais e Requisitos Não-Funcionais desse padrão. Essa técnica gera como saída o Formulário Ator X Objetivo e o Relatório de Discrepâncias decorrente da aplicação dos seus passos. Em seguida, a técnica UCRT é aplicada, tendo por entrada o Documento de Requisitos e o Formulário Ator x Objetivo gerado com a aplicação da AGRT. As saídas geradas pela UCRT são o Modelo de Casos de Uso e outro Relatório de Discrepâncias contendo as discrepâncias decorrentes da aplicação da técnica UCRT.

Ao final da aplicação da TUCCA têm-se as Especificações dos Casos de Uso, sendo que as informações para compor o diagrama estão contidas nas próprias especificações.

Vale ressaltar que a aplicação da TUCCA não descarta a necessidade de uma atividade de inspeção no Documento de Requisitos, uma vez que essa tarefa é uma decorrência da aplicação da técnica, pois seu intuito principal é a construção de Modelos de Casos de Uso. No entanto, como se sabe que, mesmo com a aplicação de atividades de garantia de qualidade de software, alguns defeitos ainda podem permanecer, a TUCCA provê uma oportunidade adicional de detectar esses defeitos que tenham permanecido mesmo que o Documento de Requisitos tenha passado por uma inspeção anteriormente. Por outro lado, se não foi realizada uma inspeção prévia no Documento de Requisitos, ao elaborar o Modelo de Casos de Uso, uma revisão que contempla as características da PBR-Usuário será realizada.

3.4. Considerações Finais

Apresentaram-se neste capítulo alguns conceitos que são importantes no contexto deste trabalho. Como se trata da geração de cenários de teste com base em casos de uso, apresentou-se o modelo de casos de uso, tanto no que se refere ao diagrama utilizado, como à sua especificação. Além disso, apresentou-se uma visão geral sobre a técnica TUCCA, que é base para a geração de casos de uso estruturados no ambiente COCAR, no qual o novo módulo de geração de cenários de teste com base em casos de uso foi implementado.

No capítulo seguinte, serão abordados conceitos de verificação, validação e teste para casos de uso, uma vez que esse é o foco deste trabalho.

CAPÍTULO 4

VV&T E CASOS DE USO

O objetivo deste capítulo é apresentar algumas abordagens encontradas na literatura para a geração de cenários de teste com base em casos de uso. Essas abordagens são essenciais para este trabalho, pois com base nelas é que a nova funcionalidade foi adicionada no ambiente COCAR.

4.1 Considerações Iniciais

A geração automática de cenários e casos de teste é um assunto que tem sido foco de muitos trabalhos de pesquisa, uma vez que visa à melhoria na qualidade da atividade de teste. Como a atividade de teste já pode ser planejada desde as primeiras fases do ciclo de vida de um software, com base nos requisitos de um sistema, vários trabalhos têm por objetivo auxiliar na geração de cenários de teste a partir do Documento de Requisitos e de modelos da UML utilizados para a modelagem desses requisitos.

Neste capítulo são comentados trabalhos de outros autores que deram subsídios a este trabalho. Todos eles tratam da execução de testes relacionados com casos de uso e auxiliaram na definição da abordagem aqui proposta. O primeiro, denominado SCENT, utiliza casos de uso para criação de Diagramas de Dependência e statecharts, que por sua vez, servem para a geração de cenários de teste. Esse trabalho é comentado na Seção 4.2. O segundo, uma abordagem baseada na UML, utiliza também os casos de uso para a geração de casos de teste a partir de uma árvore que representa as dependências entre os casos de uso, e é tratado na Seção 4.3. O terceiro trabalho, Teste Baseado na Estrutura de Casos de Uso, propõe critérios para exercitar as estruturas de relacionamento entre atores e casos de uso, com o objetivo de melhorar a qualidade da atividade de construção de casos de uso e é comentado na Seção 4.4. O quarto trabalho, Teste de Sistema com base na UML, propõe a elaboração de diagramas de atividades modificados a partir do Modelo de Casos de Uso, inserindo requisitos de teste por

meio de estereótipos propostos pelos autores, e é apresentado na Seção 4.5. O quinto trabalho, Automação da Atividade de Teste com base em Modelos UML, propõe a elaboração de diagramas de atividades individuais, para cada caso de uso, com a finalidade de mapear seus cursos normal e alternativos, e é tratado na Seção 4.6. O sexto trabalho, Geração Automática de Teste com base em Casos de Uso, também propõe o uso de diagramas de atividades modificados e o uso de estereótipos para tratar dos requisitos de teste chamados "requisitos de contrato", e é comentado na Seção 4.7. As Considerações Finais são apresentadas na Seção 4.8.

4.2. Método SCENT

O método SCENT foi desenvolvido no Instituto de Informática da Universidade de Zurich [Ryser, 1995; Ryser & Glinz, 1999a; Ryser & Glinz, 1999b; Ryser & Glinz, 1999c; Ryser & Glinz, 2000]. SCENT é o acrônimo para ***SCEN****arios-Based Validation and* ***T****est of Software*.

O método SCENT tem por objetivo ajudar desenvolvedores a criarem cenários de teste utilizando artefatos criados logo no início do processo de desenvolvimento. Os artefatos utilizados no método são o Diagrama de Casos de Uso e suas respectivas especificações em linguagem natural. Com base no Diagrama de Casos de Uso, o método propõe a representação de cada caso de uso por um statecharts e a elaboração de um Diagrama de Dependência que estabelece as dependências entre os casos de uso desde que se conheça a ordem de execução dos mesmos. Assim, o Diagrama de Dependência modela a seqüência, iteração, concorrência, dependências relacionadas ao tempo, entre outras características e, com base nele, alguns cenários de teste podem ser estabelecidos, os quais vão contemplar situações operacionais do sistema, estando, portanto, mais direcionados ao teste em nível de integração e/ou sistema.

Para exemplificar o método SCENT, considere a descrição do Sistema de Biblioteca extraído de [Ryser & Glinz, 1999a] apresentado na Figura 9. Como pode ser observado, essa descrição está redigida de uma forma em que os casos de uso já foram identificados e destacados pelos números entre parênteses. Assim, em uma situação nova, o primeiro passo seria a elaboração do Documento de Requisitos e o segundo passo seria a identificação dos

casos de uso. Após isso, os Casos de Uso são especificados e usados nos passos de criação do Diagrama de Dependência e dos statecharts.

> Em uma biblioteca, o usuário pode utilizar o seu cartão para emprestar livros e pesquisar por livros. No exemplo, há dois atores: o Usuário de Biblioteca e o Bibliotecário. Há cinco Casos de Uso no qual o Usuário é o ator: (01) Solicitar Cartão da Biblioteca, (02) Consultar Catálogo, (03) Emprestar Livros (04) Devolver Livros e (05) Solicitar Exclusão. Os Casos de Uso para o ator Bibliotecário são: (11) Registrar Usuário, (12) Excluir Usuário, (13) Atualizar Dados do Usuário, (14) Catalogar Livro, (15) Remover Livro, (16) Manter Catálogo da Biblioteca, (17) Consultar Dados e Status do Usuário, (18) Consultar Status do Livro, e (19) Cobrar Livros Atrasados.

Figura 9: Descrição do Sistema de Biblioteca
(adaptado de [Ryser & Glinz, 2000])

Na Figura 10 apresenta-se o Diagrama de Dependência correspondente ao sistema descrito na Figura 9. É importante salientar que o método não possui diretrizes bem especificadas para a elaboração do mesmo.

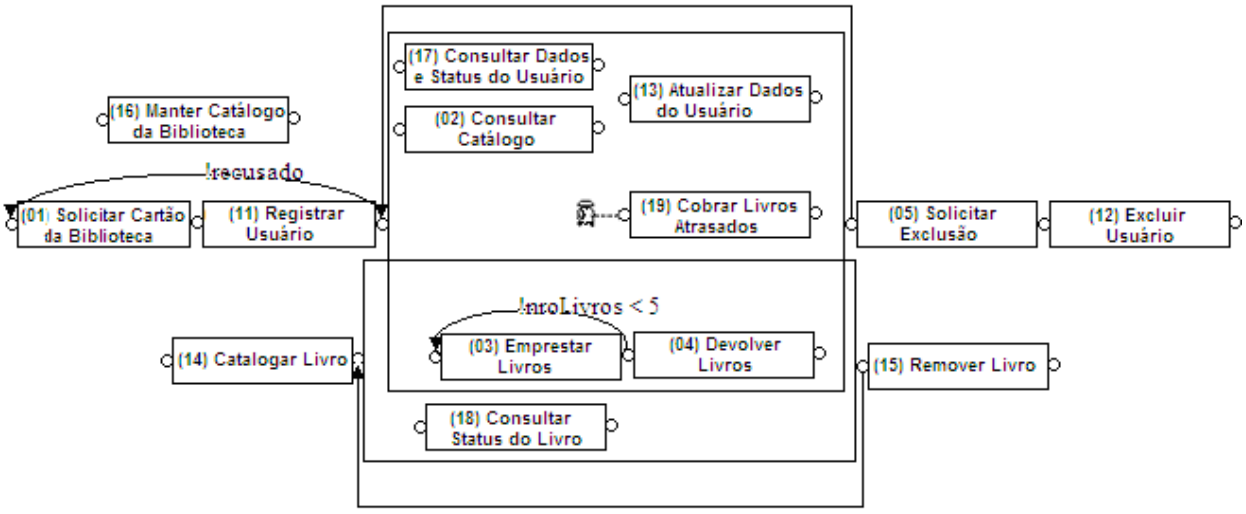


Figura 10: Diagrama de Dependência para o exemplo da Biblioteca
(adaptada de [Ryser & Glinz, 2000])

Cada caso de uso é representado por um retângulo com conectores em seus lados. Quando um caso de uso depende da execução de outro, como por exemplo, os Casos de Uso

(03) e (04), eles são associados seqüencialmente pelo conector. Nesse exemplo, o caso de uso (04) Devolver Livros só pode ser executado para um determinado usuário, se e somente se, ele tenha emprestado algum livro - caso de uso (03).

Observa-se ainda que o caso de uso (03) só pode ser executado no máximo quatro vezes, pois o usuário tem um limite de livros que ele pode emprestar. Outra característica importante no Diagrama de Dependência é o fato de que somente são executados os casos de uso (17), (02), (13), (19), (03), (04), (05), (12) se os casos de uso (01) e (11) forem executados anteriormente. Os casos de uso que não estão conectados uns aos outros podem ser executados em qualquer ordem, como por exemplo, os casos de uso (17) e (02). Quando existe um relógio associado a um caso de uso, como por exemplo, o caso de uso (19), implica que ele possui uma restrição de tempo. Nesse caso, significa que a cobrança dos livros em atraso será feita quando o prazo de devolução do livro tiver vencido.

Depois de elaborado o Diagrama de Dependência, as seqüências de teste são geradas com base nas dependências apresentadas no mesmo. Algumas das seqüências são bastante óbvias, como por exemplo, a situação dos usuários terem que ser cadastrados antes de realizar um empréstimo. Outro exemplo de seqüência de teste, utilizando a mesma notação dos autores, pode ser:

$$(01)\ (11)\ (03)^4$$

Nessa seqüência, observa-se que a situação explorada é a seguinte: o usuário solicita o cartão da biblioteca (01), é registrado (11) e pode realizar o empréstimo (03) de, no máximo, quatro livros, o que é denotado na seqüência de teste pelo número quatro sobrescrito ao lado do caso de uso (03).

Além do Diagrama de Dependência, acima citado, o SCENT propõe que, a partir do Diagrama de Caso de Uso e suas especificações sejam elaborados statecharts que serão utilizados para a criação de outros casos de teste. Nesse caso, os autores recomendam a utilização de critérios de teste para Máquinas de Estados Finitos, como por exemplo, o método W [Chow, 1978], citado na seção 2.3.

Segundo o método SCENT, o passo de transformação dos casos de uso não é formal, tanto para o Diagrama de Dependência quanto para statecharts, embora o método SCENT defina algumas heurísticas que auxiliam nessa transformação:

- Assim que os casos de uso tenham sido desenvolvidos, devem ser criados os statecharts correspondentes.

- Criar um statechart para cada caso de uso. Tanto o curso normal, quanto os cursos alternativos e as exceções devem ser capturados em um único statechart.
- Modelar primeiro o curso normal do caso de uso. Integrar os cursos alternativos posteriormente. Analisar se todos os eventos que podem ocorrer a partir de um estado estão modelados. Se isso não ocorrer, então significa que uma transição está faltando e, em geral, uma ação ou curso alternativo do caso de uso está faltando também.

Assim, percebe-se que o fato de se tentar seguir essas heurísticas já implica em realizar uma validação e verificação dos requisitos, pois durante a criação do statecharts pode-se observar que cursos alternativos não foram modelados e que, possivelmente, não foram elicitados ou foram omitidos pelos clientes/usuários ou pelos desenvolvedores do artefato em questão. Outra possibilidade é que o requisito tenha sido elicitado e que durante a modelagem dos casos de uso este requisito tenha sido omitido.

Um ponto interessante no método SCENT é a utilização do Diagrama de Dependência para a modelagem da seqüência de execução dos casos de uso. Porém, os autores do método não fornecem diretrizes que auxiliem a geração do Diagrama de Dependência a partir dos casos de uso. Com base nesse diagrama podem-se visualizar facilmente os principais cenários de operação do sistema. Assim, no contexto deste trabalho foi proposto um mecanismo que auxilia essa transformação, fornecendo essas diretrizes e possibilitando a automatização da geração de cenários de teste com base no Diagrama de Dependência. Tal mecanismo é descrito no Capítulo 6 deste trabalho.

Outra característica importante no método SCENT é a proposta de transformação da especificação dos casos de uso, em linguagem natural, para statecharts, pois, com base nos statecharts, casos de teste podem ser elaborados pelo uso de várias técnicas de geração de casos de teste baseadas em MEF. Porém, assim como para o Diagrama de Dependência, os autores não fornecem regras para esse mapeamento, sendo apenas consideradas algumas heurísticas que podem auxiliar o mapeamento da especificação em statecharts.

4.3. Uma Abordagem Baseada em UML para Teste de Sistema

Esta abordagem foi desenvolvida por Briand & Labiche (2002) no Laboratório de Engenharia de Qualidade de Software da Universidade de Carleton, no Canadá.

O objetivo dessa abordagem é apoiar a derivação de cenários de teste e de casos de teste, possibilitando também a automação dessa atividade. Essa abordagem inicia-se com a elaboração do diagrama de atividades, de acordo com a notação UML [Rational, 1999], o qual contém os casos de uso associados com cada um dos atores do sistema. Esses diagramas representam a divisão dos casos de uso de acordo com as classes que serão criadas para sua implementação como é apresentado na Figura 11.

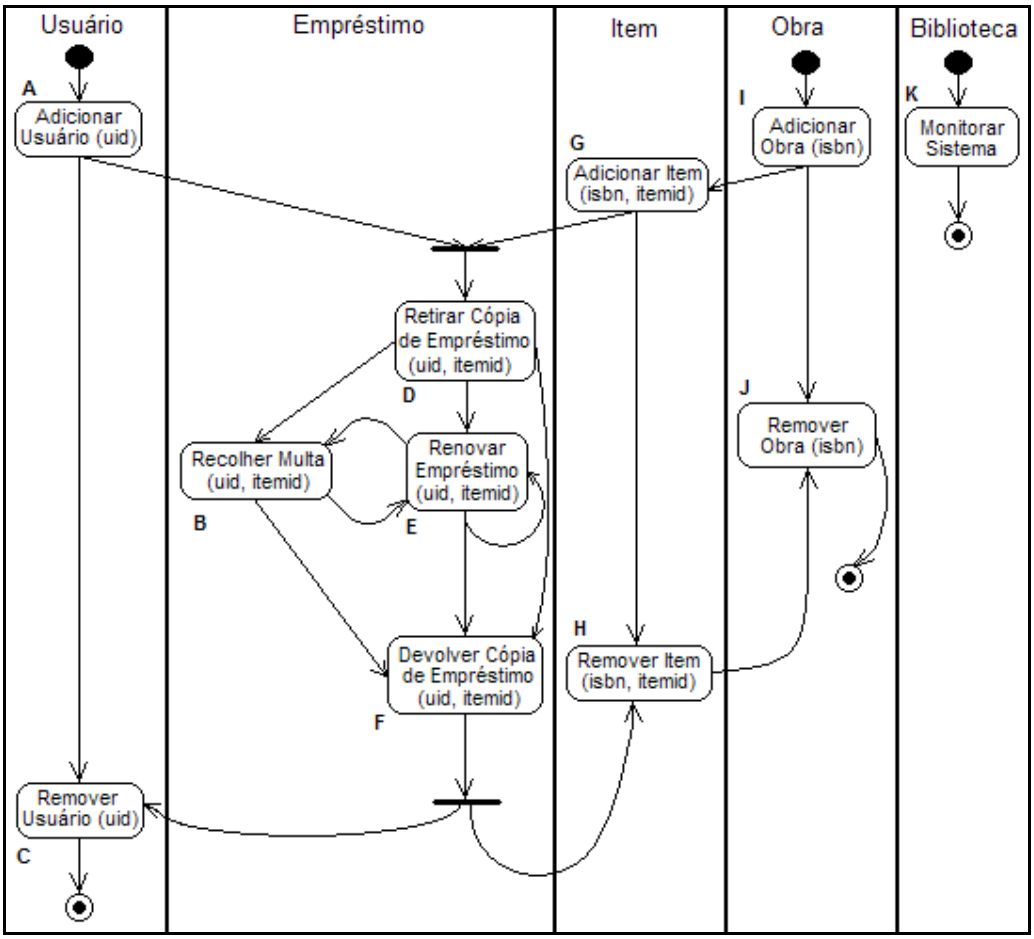


Figura 11: Diagrama de Atividades dos casos de uso para o Sistema de Biblioteca (adaptada de [Briand & Labiche, 2002])

O próximo passo do método é a geração das seqüências de casos de uso do Diagrama de Atividades, o qual é feito transformando o Diagrama de Atividades em um grafo, como mostra a Figura 12.

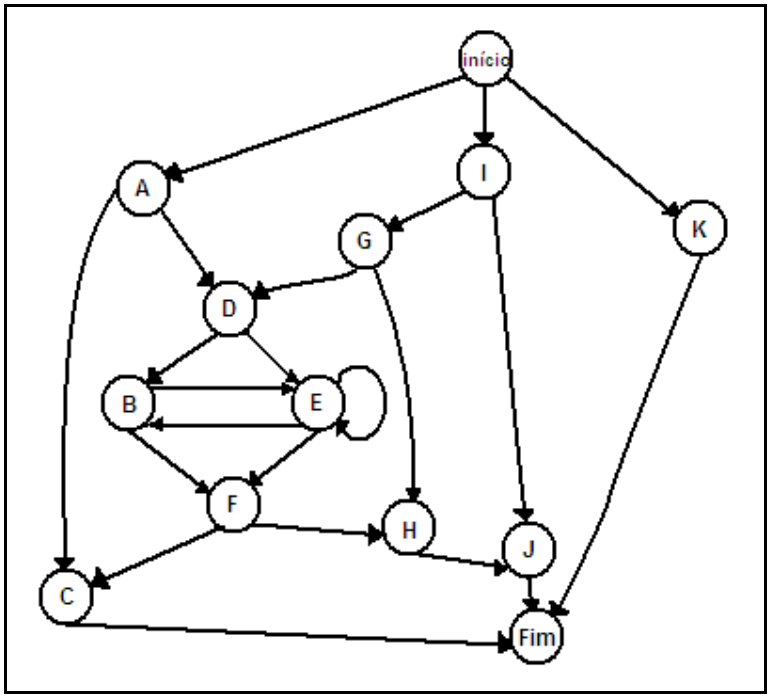


Figura 12: Grafo correspondente ao Diagrama de Atividades da Figura (adaptada de [Briand & Labiche, 2002])

Do grafo é possível a derivação de seqüências de cenários de teste. Este passo é feito por meio de uma pesquisa em profundidade no grafo criado anteriormente, gerando-se a partir dessa pesquisa, uma árvore que possui como nós os casos de uso. Desse modo, todas as possibilidades de execução dos casos de uso são mapeadas, restando apenas a criação dos casos de teste. A Figura 13 mostra a árvore derivada do grafo da Figura 12.

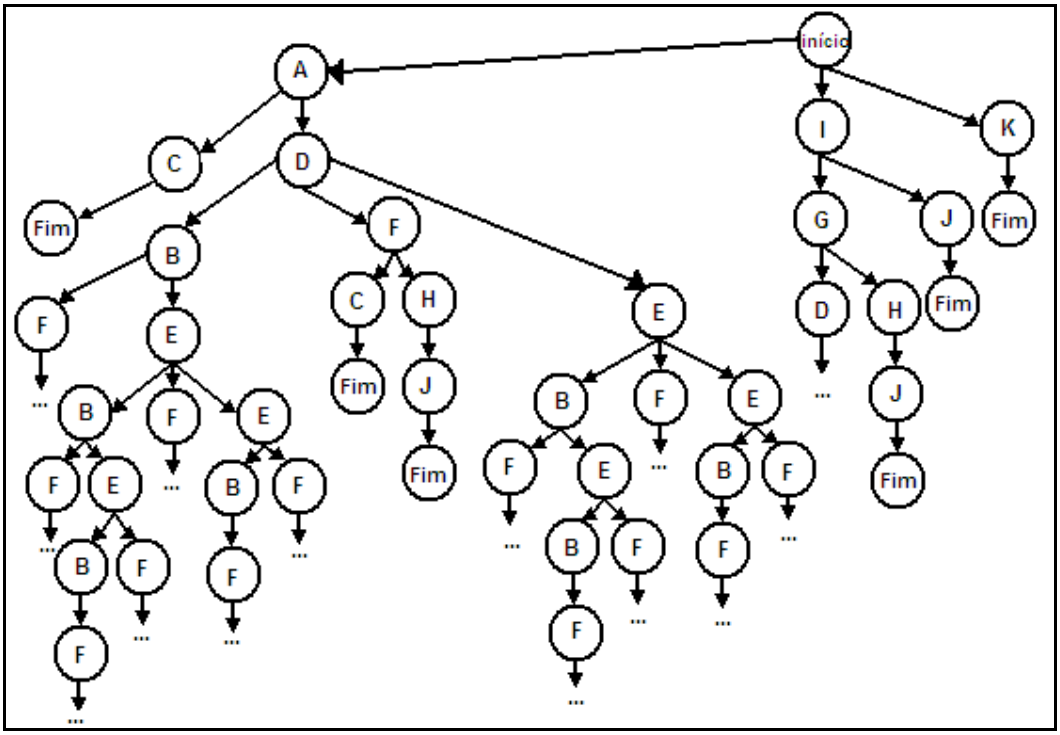


Figura 13: Árvore derivada do grafo da Figura 12 (adaptada de [Briand & Labiche, 2002])

Para a criação dos casos de teste são utilizados os parâmetros especificados nos casos de uso (conforme pode ser observado na Figura 11) obtendo, por exemplo, os seguintes casos de teste:

A(*uid*).C(*uid*)

I(*isbn*).J(*isbn*)

No primeiro exemplo, parte-se do estado inicial para o estado A que utiliza o parâmetro *uid*; em seguida, o sistema passa para o estado C, que também deve usar o parâmetro *uid*, com o mesmo valor utilizado no estado A (e não somente o mesmo tipo) e chega finalmente ao estado final. No segundo exemplo, parte-se do estado inicial, passando pelos estados I e J com a utilização do parâmetro *isbn* até que se chega ao estado final.

Os autores desse trabalho discutem a possibilidade de automação dos passos de transformação do Diagrama de Atividades para grafo, de grafo para árvore e a geração dos casos de teste a partir da árvore. Segundo Briand & Labiche (2002), o último passo é o mais complexo para a automação por causa da análise que precisa ser realizada em função dos atributos que serão utilizados pelos casos de teste. Uma das análises que deve ser realizadas

sobre os parâmetros é a de garantir que um dado parâmetro que representa a mesma informação no sistema seja transmitido com o mesmo valor entre os casos de uso; ou seja, não basta testar o tipo do parâmetro, mas também seu valor.

O trabalho de Briand & Labiche (2002) possui uma abordagem parecida com a proposta de Ryser & Glinz (2000) à medida que se preocupa com a ordem de execução dos casos de uso, representados pelo Diagrama de Atividades e pelo Diagrama de Dependência, respectivamente. Porém, a proposta de Briand & Labiche (2002) apresenta uma preocupação mais clara com a estrutura interna dos casos de uso, chegando ao ponto de avaliar o valor dos parâmetros que são passados entre os casos de uso.

4.4. Teste Baseado na Estrutura de Casos de Uso

Este trabalho foi desenvolvido na Universidade Estadual de Campinas [Carniello, 2003a; Carniello, 2003b]. O trabalho propõe um conjunto de novos critérios para exercitar os relacionamentos entre os casos de uso de um sistema a fim de melhorar a qualidade da atividade de construção de casos de uso. Esses novos critérios propostos requerem o exercício de elementos que fazem parte da estrutura dos casos de uso e diferem dos critérios baseados em especificação pelo fato de realizarem testes a partir da estrutura de uma especificação e não de sua semântica.

A aplicação da técnica de teste estrutural no Diagrama de Casos de Uso possibilita que dados de teste sejam gerados com o objetivo de fazer com que certos elementos sejam exercitados. Assim, o trabalho define critérios de teste estruturais que requerem o exercício de determinados requisitos derivados a partir do Diagrama de Casos de Uso. Esses critérios estabelecem *o que* testar em um Diagrama de Casos de Uso e quando *dar por concluído* o teste, uma vez que os critérios estruturais são capazes de quantificar a atividade de teste.

Entre os elementos estruturais existentes, foram considerados três tipos de relacionamentos a serem exercitados: os relacionamentos de *comunicação,* e os relacionamentos de *inclusão* e de *extensão.* Os relacionamentos de comunicação representam toda comunicação entre casos de uso e atores, enquanto que os relacionamentos de inclusão e extensão são os relacionamentos conhecidos denotados nos Modelos de Casos de Uso por <<include>> e <<extend>>.

A técnica define os seguintes critérios de teste dos elementos estruturais:

1. Critério todas-as-comunicações (c1)
2. Critério todas-as-inclusões (c2)
3. Critério todas-as-extensões (c3)
4. Critério todos-os-estendidos-pares (c4)
5. Critério todos-os-estendidos-combinações (c5)
6. Critério todos-os-extensores-combinações (c6)
7. Critério todas-as-comunicações-inclusões-extensões (c7)

De forma geral, o que cada critério faz é exercitar todas as combinações possíveis dos elementos estruturais existentes em um Modelo de Casos de Uso, restritos a aplicabilidade de cada critério. Isso quer dizer que c1 exercita as combinações de comunicações, sejam elas entre atores e casos de uso ou entre casos de uso; por outro lado, c7 exercita todas as combinações de comunicações, inclusões e extensões entre os elementos do modelo. Essa divisão existe para que seja aplicado o critério mais indicado para o sistema, partindo dos critérios mais simples (c1, c2 e c3), até o critério mais completo (c7).

A partir dessa definição de critérios é possível compreender porque a autora diz que esta técnica estabelece um limite para os testes, pois ao exercitar todos os elementos estruturais existentes no Modelo de Casos de Uso, o teste definido pela técnica está terminado.

Para apoiar a aplicação do conjunto de critérios proposto, foi desenvolvida uma ferramenta de cobertura de teste, denominada Use Case Tester - UCT. Esta ferramenta possui três funcionalidades: levantar os requisitos de teste dos critérios definidos; simular a execução de casos de teste a partir da especificação de casos de uso e de informações de entrada (casos de teste); e realizar a análise de cobertura da simulação da execução dos casos de teste. Cada uma dessas funcionalidades será comentada a seguir:

- *Levantamento dos Requisitos de Teste*

A ferramenta UCT tem como entrada um arquivo texto que descreve os casos de uso de um sistema. Assim, apesar da existência de estruturas de especificação de casos de uso na literatura, um novo formato de especificação de casos de uso é proposto.

Tal proposta se deve à necessidade de uma especificação com um nível de detalhamento conciso, que inclua estritamente as informações necessárias ao entendimento da funcionalidade do caso de uso e à identificação dos requisitos de teste estruturais. Esta estrutura de especificação contém elementos como relacionamentos de inclusão de origem e destino, relacionamentos de extensão de origem e destino e seus estados iniciais, para citar alguns exemplos. É importante ressaltar que, diferentemente das abordagens de Ryser & Glinz (2000) e de Briand & Labiche (2002), esse trabalho não se preocupa com a seqüência de execução entre os casos de uso, uma vez que seu objetivo é avaliar a cobertura de testes com base nos relacionamentos de <<include>> e <<extend>> do Diagrama de Casos de Uso.

Internamente, a ferramenta armazena a estrutura de relacionamentos entre os casos de uso na forma de listas de casos de uso incluídos, estendidos, etc., que são percorridas durante a fase de simulação da execução dos casos de teste.

- *Simulação da Execução dos Casos de Teste*

A representação interna de cada caso de uso contém um grafo que representa seu fluxo de ações. Esse grafo, gerado a partir da especificação do caso de uso, é usado para simular o seu comportamento. A função do grafo de fluxo de ações é semelhante à representação de casos de uso por statecharts do método SCENT, pois ambas as abordagens têm como objetivo a documentação da seqüência de ações que podem ocorrer dentro de um caso de uso.

A simulação consiste em, dado um conjunto de estados iniciais (dados de teste), percorrer o seu grafo de ações de forma a determinar a seqüência de acionamentos das inclusões e/ou extensões estabelecidas com outros casos de uso. Esse processo de simulação é considerado um ponto chave da ferramenta UCT, pois identifica os requisitos de teste exercitados para cada conjunto de dados de teste.

Outro aspecto importante da simulação está no fato de permitir verificar o comportamento de um caso de uso analisando-se a seqüência de ações simuladas. Além disso, quando a simulação é automatizada, permite que testes de regressão de versões anteriores do

diagrama possam ser executados automaticamente. O teste de regressão visa assegurar que a adição de uma nova capacidade não danifique as versões anteriores do modelo.

- *Análise de Cobertura*

Depois de gerada a seqüência das inclusões e/ou extensões acionadas durante a simulação, essa seqüência de acionamentos é comparada com os requisitos de teste dos critérios aplicados, gerando a cobertura desses requisitos durante o teste (simulação).

A seguir, na Figura 14, é apresentado um diagrama da estrutura da ferramenta UCT.

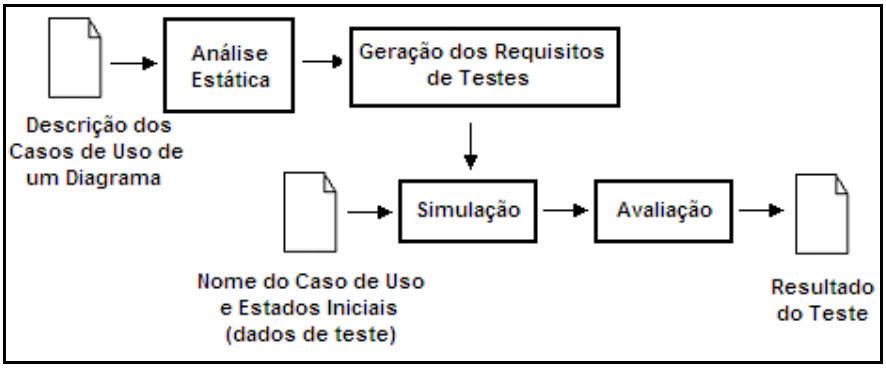


Figura 14: Diagrama da ferramenta UCT (adaptada de [Carniello, 2003b])

Em resumo, ressalta-se que o trabalho de Carniello (2003a; 2003b) realiza a análise de cobertura de casos de testes com base nos critérios propostos e, além disso, a autora não se preocupa diretamente com a seqüência de execução dos casos de uso ou regras de precedência entre eles, que estão diretamente relacionadas às necessidades de negócio do sistema sendo desenvolvido. Por outro lado, sua proposta estabelece um conjunto de critérios para exercitar os relacionamentos de comunicação entre atores e casos e uso e relacionamentos do tipo <<include>> e <<extend>> entre casos de uso.

Um aspecto interessante do trabalho de Carniello (2003a; 2003b) é a forma como são coletadas as especificações dos casos de uso, através de um *template*. Esse *template* armazena em listas os conjuntos de relacionamentos de comunicação com os atores e os relacionamentos de <<include>> e <<extend>>; além disso, o *template* define uma estrutura para armazenar o fluxo de ações internas a um caso de uso de tal forma que, posteriormente, seja montado um grafo que contenha o fluxo de ações para cada caso de uso.

A autora apresenta ainda a ferramenta UCT que é responsável por armazenar a especificação de cada caso de uso, bem como suas listas de relacionamentos. A ferramenta permite também a execução de casos de teste especificados pelo usuário e depois realiza a etapa de avaliação, comparando os casos de uso que foram exercitados com aqueles que deveriam ser exercitados segundo o critério selecionado.

4.5. Teste de Sistema com base na UML

Hartmann et al. (2005) propuseram a geração e execução de casos de teste de sistema de maneira automatizada, tendo por base modelos comportamentais de uma aplicação, com apoio da UML (Unified Modeling Language). Para a execução dos testes de sistema os autores construiram uma ferramenta gráfica de execução de testes de sistema.

Primeiramente, o projetista de teste modela os requisitos por meio do Modelo de Casos de Uso. Em seguida, com o uso do Diagrama de Atividades da UML modificado para contemplar casos de uso, modela o curso normal e os cursos alternativos dos casos de uso. Os requisitos de testes são inseridos no diagrama de atividades criado por meio de estereótipos pré-definidos. Por fim, por meio da ferramenta gráfica de geração de casos de teste de sistema, um conjunto de casos de teste ou scripts de teste é gerado automaticamente.

A Figura 15 mostra a abordagem utilizada para a melhoria da atividade de teste de sistema.

A modelagem inicial pode ser feita em alguma ferramenta já existente ou fazer uso do UML Editor Kit, fruto do trabalho de Hartmann et al (2005). Os usuários devem refinar os diagramas UML e utilizar a ferramenta criada pelos autores, chamada TDE (Test Development Environment), para gerar conjuntos de casos de teste em formato XML (eXtensible Markup Language) [OMG, 2010]. A partir dos arquivos XML, são gerados scripts para execução de testes, armazenados no formato XSL (eXtensible Stylesheet Language) [OMG, 2010] e interpretados por uma ferramenta gráfica.

Uma importante contribuição de Hartmann et al. (2005), e que difere dos trabalhos descritos anteriormente, é o uso de arquivos no formato XML e XSL que são interpretados por uma ferramenta para obtenção de scripts e casos de teste.

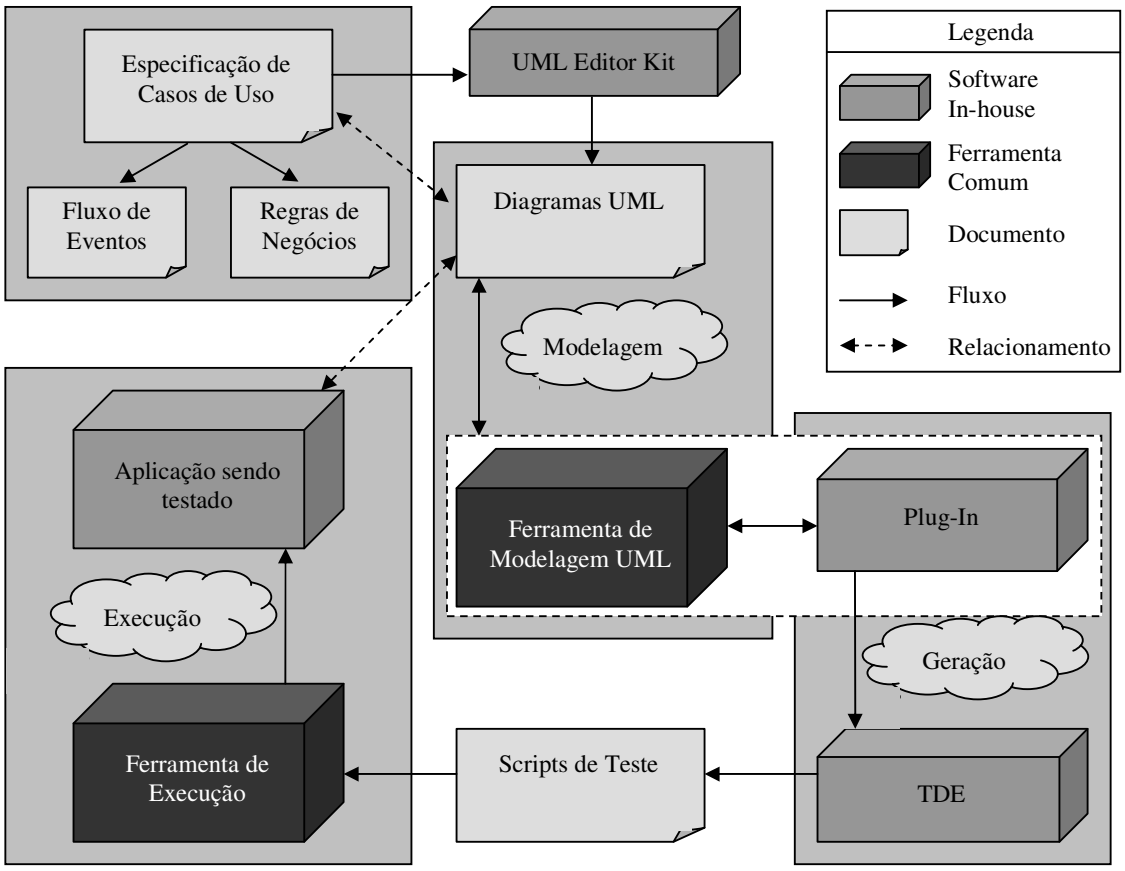


Figura 15: Abordagem para melhoria de teste de sistema (adaptado de [Hartmann et al., 2005])

4.6. Automação da Atividade de Teste com base em Modelos UML

Vieira et al. (2006) também propuseram uma abordagem com base em modelos da UML, cujo objetivo é apoiar a automação da atividade de teste de software. Para tanto, combina técnicas e modelos existentes, como o Modelo de Casos de Uso e o diagrama de atividades, e utiliza estereótipos para adicionar requisitos de teste nos diagramas.

Inicialmente, o método faz uso do Modelo de Casos de Uso para descrever os relacionamentos entre os casos de uso. Após esse mapeamento, diagramas de atividades são elaborados para cada um dos casos de uso, com o intuito de mapear a lógica existente em cada um deles. O conjunto dos diagramas de atividades representa o comportamento geral da aplicação. Com base nos diagramas de atividades elaborados, caminhos de teste são gerados e

uma ferramenta chamada TDE/UML (Test Development Environment using UML) apóia a geração dos casos de teste. Essa ferramenta é um *plug-in* para ambientes existentes, como Rational Rose, ArgoUML, entre outros.

A concepção da utilização de diagramas de atividades para mapear os cursos normal e alternativos de cada caso de uso também é utilizado no contexto deste trabalho, porém ao invés da utilização de diagramas de atividades, optou-se pelo uso de grafos, sem a necessidade de criar estereótipos para o Modelo de Casos de Uso ou diagrama de atividades, como proposto por Vieira et al. (2006).

4.7. Geração Automática de Teste com base em Casos de Uso

Nebut et al. (2006) propuseram uma metodologia para geração automática de cenários de teste funcionais em duas fases. A primeira fase compreende a geração de objetivos de teste a partir de um Modelo de Casos de Uso modificado, com o uso de uma abordagem "requisito por contrato" proposta no trabalho. O intuito dessa abordagem é obter a ordenação das funcionalidades que deveriam ser encontradas no sistema, construindo um modelo de simulação e gerando as seqüências corretas de execuções dos casos de uso. A segunda fase tem por objetivo gerar cenários de teste a partir dos objetivos de teste formulados na fase anterior. Os cenários de testes gerados podem se tornar casos de teste executáveis diretamente ou podem ser modificados para então se tornarem casos de teste executáveis. O processo de transformação de objetivos de teste para cenários de teste envolve trocas de mensagens entre o ambiente e o sistema, as quais devem ser modeladas com o auxílio de outros diagramas da UML, tais como o diagrama de atividades, diagrama de seqüência ou diagrama de estados.

A Figura 16 mostra os passos das 2 fases da metodologia proposta pelos autores. Os passos (a), (b) e (c) correspondem à primeira fase, e os passos (d) e (e) correspondem à segunda fase da metodologia.

A metodologia preocupa-se com a parametrização para geração dos cenários de teste, tal qual proposto por Briand & Labiche (2002), mas a diferença é que os parâmetros são utilizados no Modelo de Casos de Uso estendido, por meio dos contratos, que foram propostos pelos autores, sendo necessária uma adaptação dos modelos existentes na UML para contemplar a parametrização.

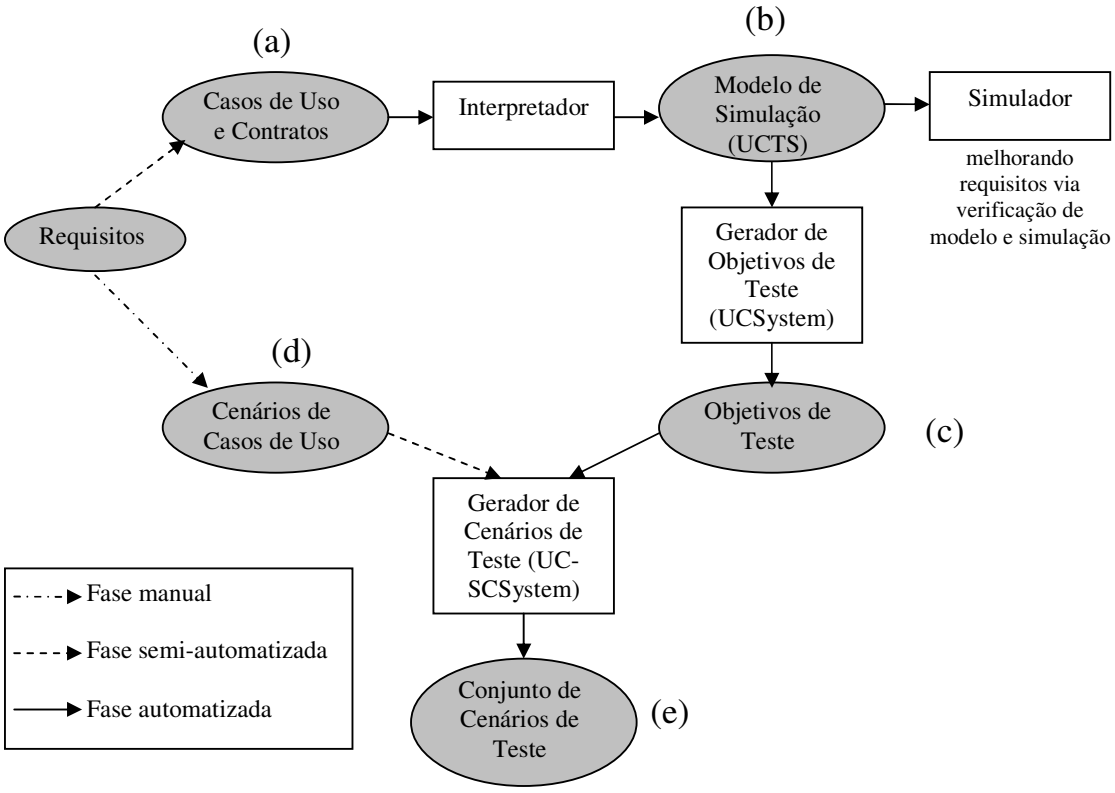


Figura 16: Metodologia de 2 fases para geração de cenários de teste (adaptado de [Nebut et al., 2006])

4.8. Considerações Finais

Apresentaram-se neste capítulo os trabalhos da literatura que serviram de base para a geração de Cenários de Teste com base em Casos de Uso, implementada no ambiente COCAR.

Foi possível visualizar uma integração das propostas dos trabalhos apresentados, tendo como inspiração inicial o uso do Diagrama de Dependência do método SCENT para representar as dependências de execução entre os casos de uso. O trabalho de Briand & Labiche (2002) contribuiu principalmente com a idéia da transformação do Diagrama de Atividades de Casos de Uso em grafos e, a partir dos mesmos, a transformação em árvore, utilizando um algoritmo para percorrer seus nós e gerar casos de teste. O trabalho de Carniello (2003a) apresenta uma forma estruturada de armazenar a especificação dos casos de uso, além

de uma ferramenta de apoio para geração de casos de teste com base em casos de uso, ainda que não se preocupa com a ordem de execução dos mesmos.

Dos demais trabalhos, foram aproveitadas contribuições mais específicos. Algumas dessas contribuições podem ser destacadas no contexto de implementação, como o uso de arquivos XML [Hartmann et al., 2005], e outras na concepção da idéia, como a elaboração de diagramas de atividades individuais para cada caso de uso, que, no contexto deste trabalho, foi utilizada para a elaboração de cenários de teste unitários.

No Capítulo 6 apresenta-se a abordagem elaborada neste trabalho para a geração de cenários de teste com base em casos de uso, mostrando-se com mais detalhes as contribuições de cada trabalho comentado neste capítulo.

Como a geração dos casos de teste com base nos casos de uso foi acoplada ao ambiente COCAR como uma nova funcionalidade, este ambiente será abordado com maior detalhe no capítulo seguinte.

CAPÍTULO 5

O Ambiente COCAR

O objetivo deste capítulo é fazer uma breve apresentação do ambiente COCAR, que foi fruto de outros dois trabalhos de mestrado do grupo. Esse ambiente tem como base a implementação da técnica de leitura TUCCA, tornando-o dedicado para o modelo de casos de uso, com base no qual são realizadas outras funcionalidades. Ele é aqui apresentado porque inicialmente a proposta seria adicionar o módulo de geração de cenários de teste com base em casos de uso nesse mesmo ambiente original, porém com o decorrer do projeto optou-se por alterações de implementação que são descritas no Capítulo 6.

5.1. Considerações Iniciais

O ambiente COCAR foi idealizado com o objetivo de dar suporte a atividades do desenvolvimento de software, considerando como base fundamental os Modelos de Casos de Uso. Visando à construção desse ambiente, alguns trabalhos de mestrado foram conduzidos, com o 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 de desenvolvimento de software. Assim, na Seção 5.2 apresentam-se suas principais funcionalidades bem como algumas de suas principais telas para melhor ilustrá-lo, na Seção 5.3 são feitas considerações sobre a implementação da versão inicial, anterior a este trabalho, e na Seção 5.4 as Considerações Finais.

5.2. O ambiente COCAR

Em sua primeira versão, o COCAR contém as seguintes funcionalidades: i) suporte à elaboração de Modelos de Casos de Uso, ii) geração de Pontos por Casos de Uso e iii) suporte à Rastreabilidade de Requisitos. Um dos objetivos deste trabalho é a adição de uma nova

funcionalidade: iv) geração de Cenários de Teste com base em Casos de Uso. Cada uma dessas quatro funcionalidades corresponde a um trabalho de mestrado, sumarizado a seguir.

i) Suporte à elaboração de Modelos de Casos de Uso:

Esse módulo do COCAR corresponde ao trabalho de mestrado de Belgamo (2004a), intitulado GUCCRA – Guidelines for Use Case Construction and Requirements Document Analysis, mas cuja técnica recebeu, posteriormente, a denominação TUCCA – Technique for Use Case Construction and Construction-Based Requirements Analysis, que tem sido utilizada ao longo deste texto. 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 fique mais sistematizada e independente da experiência do desenvolvedor, como também menos suscetível a mau entendimentos que levem a 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. Ao utilizar essa técnica consegue-se gerar Modelos de Casos de Uso padronizados, atentando para a clareza e completude dos requisitos funcionais e não-funcionais que são retratados nos modelos. 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.

A Figura 17 ilustra um dos primeiros passos da aplicação da técnica TUCCA, quando o usuário da ferramenta faz as marcações para os requisitos, classificando-as em possíveis atores, funções e restrições.

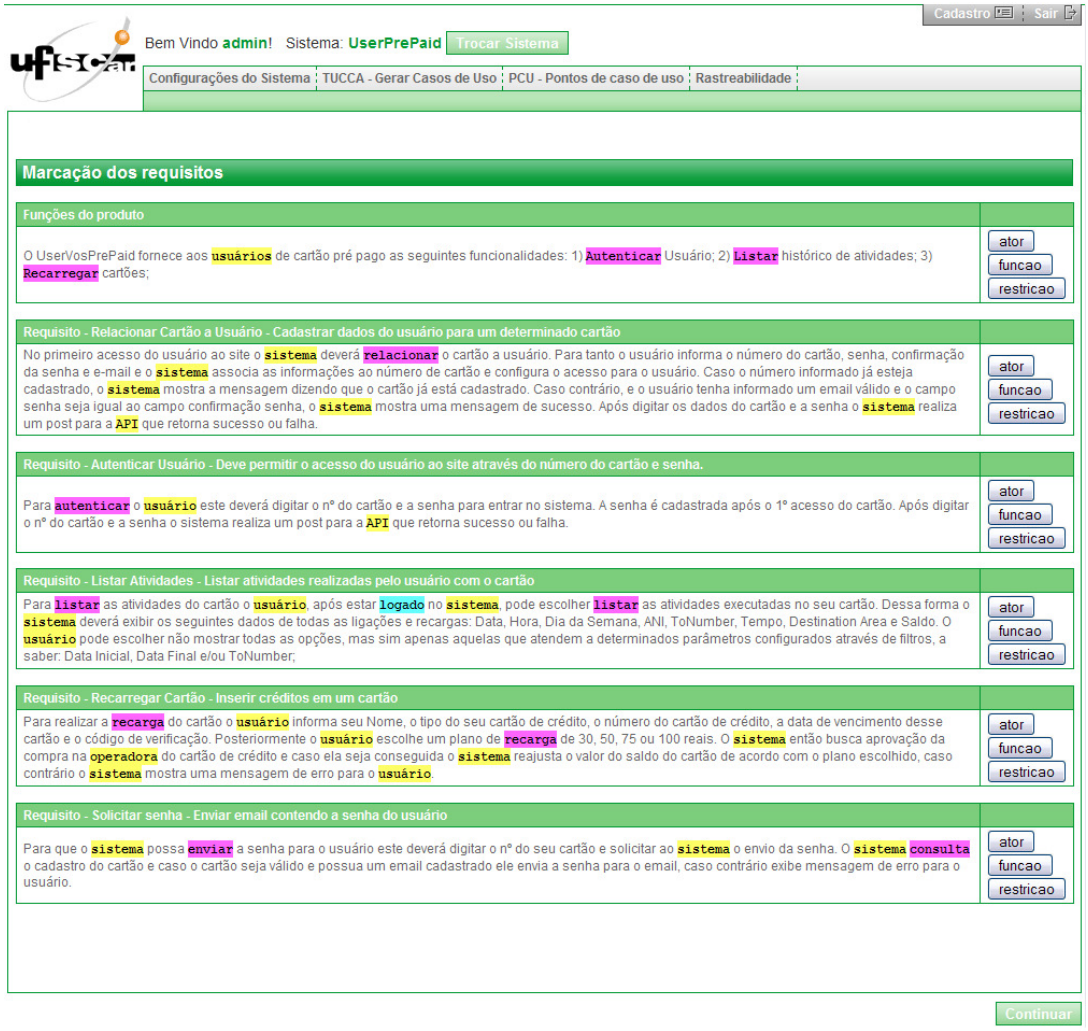


Figura 17: Processo de aplicação da TUCCA, com a marcação de atores, funções e restrições (adaptado de [Thommazo, 2007])

ii) Geração de Pontos de Casos de Uso:

Esse módulo do COCAR corresponde ao trabalho de mestrado de Martins (2007), e tem como objetivo a geração de Pontos por Casos de Uso [Karner, 1993], os quais são gerados na ferramenta 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 compor as especificações, as informações necessárias para a geração dessa métrica ficam acessíveis de uma forma que facilita a sua geração. O valor dos Pontos por Casos de Uso também é convertido em Pontos por Função [Karner, 1993] pela ferramenta. Esses dados auxiliam na fase de planejamento do software uma vez que podem ser usados para estimar tempo, custo e esforço de desenvolvimento.

A Figura 18 mostra a funcionalidade de geração de relatório de Pontos de Casos de Uso, com os dados consolidados.

PCU - Dados consolidados

Ator	Complexidade
sistema	1
Usuário Calling Cards	3
TOTAL	4

Caso de Uso	Complexidade
Enviar Senha	10
Relacionar Cartão a Usuario	15
Consulta Cartao	5
Autenticar Usuário	10
Listar Atividades	10
Recarregar Cartao	10
TOTAL	60

Métrica	Valor
Pontos de caso de uso não ajustado	64
Fator de Complexidade Técnica	1.02
Fator Ambiental	0.74
Pontos de caso de uso ajustado	48.31
Pontos por função	139.64

Figura 18: Relatório de geração de Pontos de Casos de Uso (adaptado de [Martins, 2007])

iii) Suporte à Rastreabilidade de Requisitos;

Esse módulo do COCAR corresponde ao trabalho de mestrado de Thommazo (2007), que tem por objetivo a rastreabilidade dos requisitos entre o Documento de Requisitos e o Modelo de Casos de Uso. Dessa forma, considerando a natureza dinâmica dos requisitos, a ferramenta dá suporte a esse controle, permitindo o acompanhamento das alterações, informação esta que poderá ser usada também para auxiliar a geração de cenários de teste de regressão, o qual deve ser aplicado sempre que o sistema sofrer alterações. Além disso, o trabalho também trata da previsão do impacto gerado por modificações em um requisito sobre os outros requisitos, criando de maneira automática a matriz de impacto [Sommerville, 2003] para esse controle. O módulo de suporte à rastreabilidade de requisitos não avalia diretamente o impacto sobre outros artefatos produzidos durante o processo de desenvolvimento de software, como Modelos de Classes ou Modelos de Componentes. Porém, uma vez que são identificados os requisitos que são impactados pela mudança de um requisito específico, é

possível identificar artefatos candidatos a sofrerem modificações uma vez que tenham sido gerados a partir desses requisitos que tenham sido impactados.

A Figura 19 é um exemplo de como a rastreabilidade entre requisito e caso de uso é visualizada na ferramenta pelo usuário.

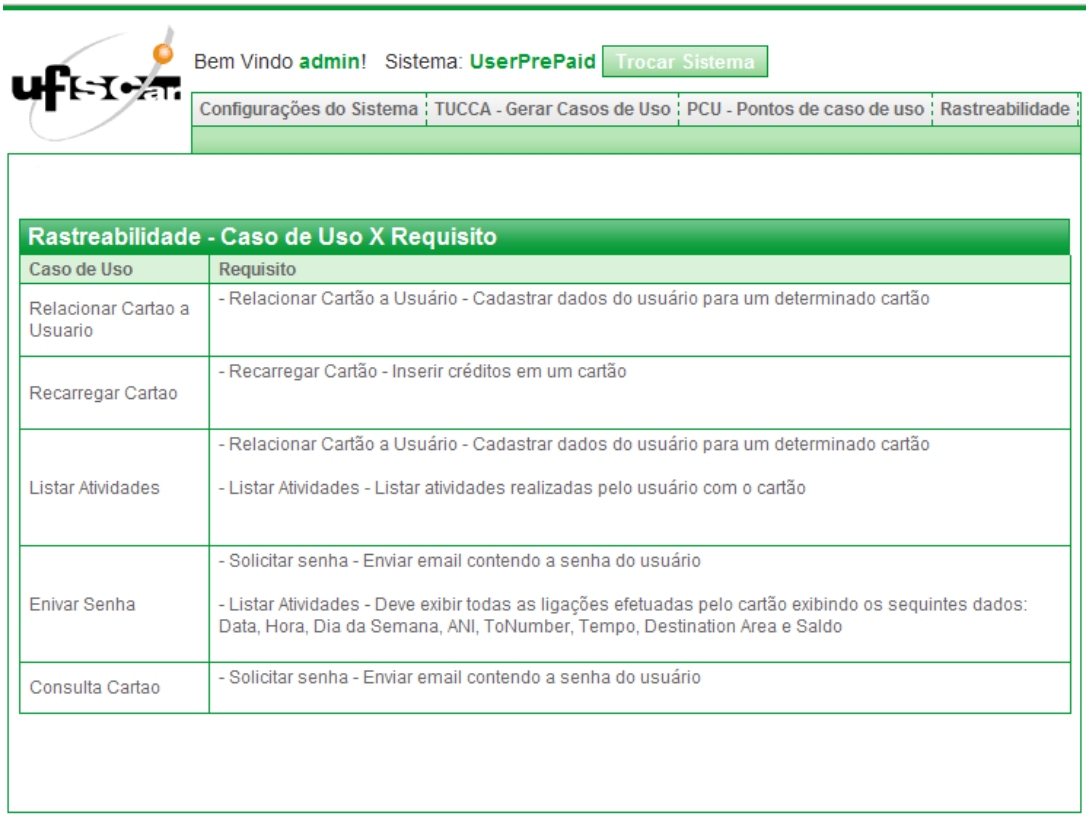


Figura 19: Exemplo de visualização da rastreabilidade entre requisito e Caso de Uso (adaptado de [Thommazo, 2007])

iv) Geração de Cenários de Teste com base em Casos de Uso:

Esse módulo do COCAR corresponde ao trabalho aqui proposto, que consiste em gerar cenários de teste a partir dos casos de uso, cujo desenvolvimento e implementação são detalhados mais adiante, no Capítulo 6.

Além desses módulos, existe a intenção de se adicionarem outros gradativamente. Um deles é um módulo que auxilie na elaboração de Documentos de Requisitos. Nesse sentido, um trabalho de mestrado já foi realizado e defendido por Kawai (2005). Esse trabalho propôs diretrizes para a construção do Documento de Requisitos, facilitando a identificação das entidades envolvidas e seus papéis, informações de entrada e saída para cada processamento, a execução da funcionalidade em si, além de restrições e condições de execução de cada

funcionalidade. No entanto, o foco desse trabalho ficou restrito aos requisitos funcionais e, para disponibilizar esse recurso no COCAR, pretende-se fazer algumas extensões para abordar também os requisitos não-funcionais, por exemplo. Além das diretrizes, esse trabalho também propôs um *checklist* para auxiliar o engenheiro de software a garantir que seu Documento de Requisitos contenha todas as informações que foram julgadas necessárias de acordo com as diretrizes.

Para viabilizar a implementação do ambiente COCAR e agregar as funcionalidades propostas por esses diversos trabalhos, é necessária uma base de informações centralizada e compartilhada entre todos os módulos do ambiente.

O grande objetivo da ferramenta é poder contribuir com a comunidade de Engenharia de Software provendo apoio ao desenvolvimento de softwares com a fundamentação de vários trabalhos de pesquisa científica.

5.3. Considerações Sobre a Implementação da Versão Inicial

O ambiente escolhido para a implementação inicial do COCAR foi o ambiente Web Cliente Servidor, sendo o objetivo principal ter maior facilidade de acesso, abstraindo-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. Com essa escolha, buscou-se também uma maior integração e fácil acesso por parte de todos os envolvidos no projeto, que poderiam acessá-la remotamente pela Internet com o uso de um web browser.

A linguagem escolhida para implementar o ambiente em um Web Cliente Servidor foi a linguagem Java por vários motivos, a saber [Thommazo, 2007]:

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, 2010];
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, 2009a];
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, 2009b];
6. Como o ambiente COCAR é uma ferramenta cujo crescimento já está previsto com a adição de outros recursos e funcionalidades, buscou-se utilizar padrões de projetos existentes na literatura que permitam a fácil manutenção do software buscando conferir ao produto final alta manutenibilidade.

Os padrões MVC (Model-View-Controller) foram escolhidos para separar as camadas de regras de negócio, de apresentação e de controle, visando a um menor acoplamento do software, facilitando o processo de manutenção, com a independência de camadas. Para tanto se fez uso do *framework* Struts [Apache, 2010] e DAO (Data Active Objects).

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 vasta documentação.

5.4. Considerações Finais

Apresentou-se neste capítulo o ambiente COCAR, que envolve este e outros trabalhos, cada qual atuando sobre um aspecto do processo de desenvolvimento de software, mas todos eles tendo como base o Modelo de Casos de Uso.

Um dos trabalhos guia a atividade de construção dos Modelos de Casos de Uso ao mesmo tempo em que avalia o Documento de Requisitos, por meio da aplicação da técnica de leitura TUCCA. Os outros trabalhos utilizam a representação do modelo de casos de uso construído no ambiente para aplicar as funcionalidades propostas, ou seja, gerenciar os requisitos por meio de algumas opções de rastreabilidade dos requisitos, e também gerar os Pontos por Casos de Uso, para dar suporte a atividades de estimativas de projeto.

Foram comentados ainda aspectos de implementação da versão da ferramenta anterior à implementação do novo módulo de geração de cenários de teste a partir de casos de uso, uma vez que o ambiente sofreu alterações para uma melhor adaptação do novo módulo. O

novo módulo, bem como detalhes de sua implementação, e as modificações realizadas são descritas no capítulo a seguir.

CAPÍTULO 6

Cenários de Teste com base em Casos de Uso no Ambiente COCAR

O objetivo deste capítulo é apresentar as novas funcionalidades implementadas no ambiente COCAR, todas frutos deste trabalho de mestrado, sobretudo o módulo de Geração de Cenários de Teste com base em Casos de Uso. Apresenta ainda os demais módulos implementados, sendo que alguns foram previstos inicialmente, como a importação de Casos de Uso a partir de arquivos XMI, e outros surgiram durante a execução das atividades inicialmente programadas, tais como o cadastro manual de Casos de Uso e a necessidade de conversão da ferramenta para Desktop. Todas as atividades são apresentadas detalhadamente neste capítulo.

6.1. Considerações Iniciais

Como visto no capítulo anterior, o ambiente COCAR foi desenvolvido inicialmente para a tecnologia Web Cliente Servidor, buscando-se com essa escolha uma maior integração e fácil acesso por parte de todos os envolvidos no projeto, que poderiam acessá-lo remotamente pela Internet com o uso de um web browser. Além disso, foi utilizada a linguagem Java e o padrão MVC (Model-View-Controller), além do gerenciador de banco de dados MySQL, conforme detalhamento no Capítulo 5.

No início da implementação de um novo módulo de geração de cenários de teste com base em casos de uso para o ambiente COCAR, surgiram elementos que mostraram a necessidade de uma mudança no projeto inicial. O principal elemento foi a possibilidade de integração da ferramenta Crista, que é fruto do trabalho de mestrado de Porto (2009) [Porto et al., 2008], com o ambiente COCAR. A Crista é uma ferramenta desenvolvida para o ambiente Desktop, pois utiliza recursos gráficos que não estão disponíveis em nenhum pacote para o ambiente Web e, dessa maneira, foi implementada em sua totalidade para Desktop.

Com essa provável integração entre o ambiente COCAR e a ferramenta Crista, que seria parte de um projeto de doutorado a ser iniciado, optou-se por já implementar os novos módulos - frutos deste trabalho de mestrado - na versão Desktop, para não haver a

necessidade de uma migração futura, à medida que a Crista fosse incorporada ao COCAR. Assim sendo, optou-se por gastar um tempo maior na etapa de implementação e converter os principais módulos do ambiente COCAR, necessários para a conclusão deste projeto, da versão Web Cliente Servidor para a versão Desktop. Para tanto, manteve-se a mesma linguagem Java, na qual a Crista foi implementada também, o mesmo modelo MVC e o mesmo gerenciador de banco de dados MySQL.

Para implementar as novas funcionalidades no ambiente COCAR foi necessário converter alguns módulos essenciais da versão Web para a versão Desktop, tais como Cadastro de Sistemas, Cadastro de Atores, Cadastro de Requisitos. Essa conversão foi fundamental para que o módulo de Geração de Cenários de Teste com base em Casos de Uso, um dos objetivos deste trabalho, pudesse ser implementado, além dos módulos de Inserção Manual de Casos de Uso e de Integração do COCAR com outras ferramentas. Esses novos módulos são apresentados nas seções seguintes.

6.2. Geração de Cenários de Teste com base em Casos de Uso

A geração de cenários de teste com base em casos de uso é a principal contribuição deste trabalho e foi desenvolvida sobretudo com base nos trabalhos de Ryser & Glinz (1995; 1999a; 1999b; 1999c; 2000), pelo qual foi criado o método SCENT, no trabalho de pesquisa de Briand & Labiche (2002) e no trabalho de pesquisa de Carniello (2003a; 2003b). Outros trabalhos, como [Hartmann et al., 2005] [Vieira et al., 2006] e [Nebut et al., 2006] também contribuíram para algumas tomadas de decisão, embora com menor interferência na abordagem aqui apresentada. Essa geração de cenários de teste tem por objetivo gerar os diversos cenários possíveis para um sistema, seja ele em nível unitário (para um caso de uso) ou de integração (compreendendo dois ou mais casos de uso), dando subsídios para que a equipe de teste elabore o plano de teste e seus respectivos casos de teste de maneira mais eficaz.

O método SCENT [Ryser, 1995; Ryser & Glinz, 1999a; Ryser & Glinz, 1999b; Ryser & Glinz, 1999c; Ryser & Glinz, 2000] foi importante para a concepção do módulo de geração de cenários de teste, pois ele mostra a relevância de se estabelecer a dependência lógica entre os casos de uso. Ainda que o método não mostre como construir um diagrama de dependência entre os casos de uso de maneira eficiente, as poucas diretrizes e os elementos para construção

do diagrama apresentados, além do exemplo do sistema de Biblioteca (vide Figura 10), serviram como inspiração para aprofundamento do estudo de como relacionar os casos de uso com o intuito de gerar cenários de teste, sobretudo cenários de teste de integração. Os cenários de teste de integração também foram alvo do trabalho de Nebut et al. (2006), no qual tais cenários foram modelados por meio de uma adaptação do diagrama de atividades tradicional da UML.

O trabalho de Briand & Labiche (2002) foi importante por mostrar que os casos de uso podem ser organizados por meio de um diagrama de atividades da UML e, com base nesse diagrama, podem ser elaborados grafos com base na ordem de execução dos casos de uso. Essa idéia, juntamente com a abordagem do método SCENT para a concepção do diagrama de dependência, foram fundamentais no processo de geração de cenários de teste unitários e de integração, uma vez que ambos são baseados em grafos de execução, sejam eles dos próprios casos de uso (cenários de integração), ou dos passos de seus cursos normais e alternativos (cenários unitários). Essa mesma idéia de tratar os casos de uso isoladamente com o intuito de gerar cenários de teste unitários também pode ser observada no trabalho de Vieira et al. (2006).

O trabalho de Carniello (2003a; 2003b) inspirou o armazenamento dos casos de uso por meio de tabelas relacionais que permitiram acessar a informação necessária para a geração dos cenários de teste, de maneira análoga à que Carniello fez utilizando listas encadeadas.

O trabalho de Hartmann et al. (2005) inspirou a utilização de arquivos XMI para a integração do ambiente COCAR com outras ferramentas CASE, uma vez que os autores utilizaram arquivos XML como entrada da ferramenta desenvolvida em seu trabalho.

Enfim, todos os trabalhos citados foram importantes e contribuíram de alguma maneira para a realização deste projeto. Como resultado foi implementado um módulo que gera diretrizes para a aplicação de cenários de teste unitários e de integração, conforme é apresentado nas Seções 6.4.1 e 6.4.2. A tela para geração dos cenários de teste é exibida na Figura 20.

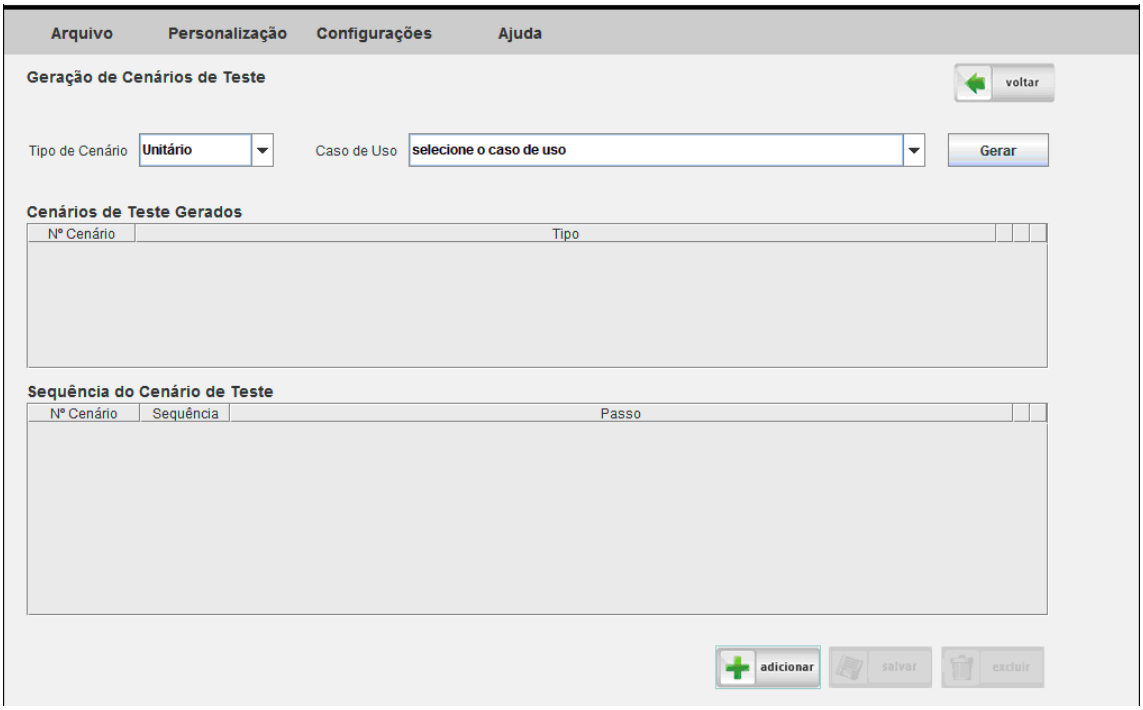


Figura 20: Tela de Geração de Casos de Teste Unitários e de Integração

Nessa tela é possível selecionar o tipo de cenário de teste a ser gerado, se é unitário ou de integração. Há uma caixa de seleção denominada "Tipo de Cenário" com as duas opções disponíveis. Se o tipo de cenário de teste escolhido for o unitário, então a caixa de seleção denominada "Caso de Uso" é habilitada e os casos de uso do sistema ativo são exibidos para escolha.

Ao clicar no botão "Gerar", os cenários de teste são gerados de acordo com o tipo selecionado e carregados para a lista denominada "Cenários de Teste Gerados". Nessa lista cada cenário contém um número, que é gerado seqüencialmente, e o tipo do mesmo, se é unitário ou de integração.

Ao selecionar um dos cenários de teste gerados, automaticamente a ferramenta exibe os passos desse cenário de teste na lista denominada "Seqüência do Cenário de Teste". Uma vez gerados, há a possibilidade de imprimi-los ou salvá-los em arquivo de formato texto.

6.2.1. Cenários de Teste Unitário

No contexto do ambiente COCAR, cenário de teste unitário compreende a geração de cenários de teste para um único caso de uso, tendo por base os passos de seu curso normal,

além de incluir os passos de todos os seus cursos alternativos, quando houver. A implementação dessa funcionalidade foi inspirada no trabalho de Briand & Labiche (2002), de construir grafos de execução que podem ser percorridos por meio de alguma técnica de execução, possibilitando variadas combinações. A diferença é que, no contexto deste trabalho, os vértices do grafo correspondem aos passos dos cursos do caso de uso e não aos casos de uso em si para relacionar a dependência entre eles.

Dessa maneira, a concepção utilizada é de que os passos do curso normal sejam interpretados como os vértices principais de um grafo direcionado e seqüencial, respeitando a ordem de execução do curso normal do caso de uso. Os passos de um curso alternativo são tratados como uma ramificação dos vértices principais do grafo - que representa o curso normal - cujo ponto de incidência é o passo para o qual o curso alternativo existe. Com isso, o curso normal torna-se o primeiro cenário de teste unitário e, para cada curso alternativo existente, obtém-se um novo cenário de teste, que é composto pelos passos do curso normal até o passo em que ocorre o curso alternativo, bem como pelos passos desse curso alternativo.

No caso de o último passo do curso alternativo voltar para um passo do curso normal, os passos restantes do curso normal são adicionados ao caso de teste também. Para que o algoritmo de geração de casos de teste funcione de acordo com o que foi escrito, o passo que retorna ao curso normal deve ser descrito da seguinte maneira: colocar a expressão “Volta ao passo” seguido do número do passo do curso normal para o qual deve retornar e dar continuidade ao caso de teste. Para exemplificar essa situação, considere que o último passo do curso alternativo volte para o passo 5 do curso normal de um caso de uso. A descrição deste passo deve conter “Volta ao passo 5”.

É importante ressaltar que, no caso de um curso normal possuir mais de um curso alternativo, apenas um curso alternativo é relacionado a um cenário de teste, ou seja, segundo o algoritmo construído, não haverá num mesmo caso de teste unitário, dois cursos alternativos sendo percorridos. Tal restrição foi aplicada por causa da enorme complexidade de se implementar todas as possíveis combinações de cursos alternativos. Assim, optou-se por fazer uma analogia ao critério de teste estrutural (caixa branca) dos Caminhos Básicos [Presmann, 2006]. Esse critério é aplicado de forma alternativa ao critério Todos os Caminhos, pois este último é, na maioria das vezes, impraticável. Como todos os caminhos do grupo de controle são uma combinação dos caminhos básicos, percorrendo-se estes, supostamente os outros também terão sido percorridos, ou seja, supostamente a combinação de todos os cursos também terá sido exercitada.

O número de cenários gerados para o teste unitário corresponde ao número total de cursos que esse caso de uso possui, isto é, dado um caso de uso que possua um curso normal e quatro cursos alternativos, o número total de cenários de teste unitários gerados será cinco, sendo o primeiro referente aos passos do curso normal do caso de uso, e os demais combinações entre o curso normal e o curso alternativo, sendo que somente um curso alternativo é percorrido para cada cenário de teste, conforme explicado anteriormente.

A Figura 21 mostra os cenários de teste gerados para o caso de uso "Autenticar Usuário" no ambiente COCAR. Como esse caso de uso possui um curso normal e cinco cursos alternativos, sendo um para o passo 3, um para o passo 4, um para o passo 5 e dois para o passo 6, o número total de cenários de teste gerados foi seis e eles estão carregados na lista "Cenários de Teste Gerados". Clicando-se no cenário de número 3, seus passos são exibidos na lista "Seqüência do Cenário de Teste".

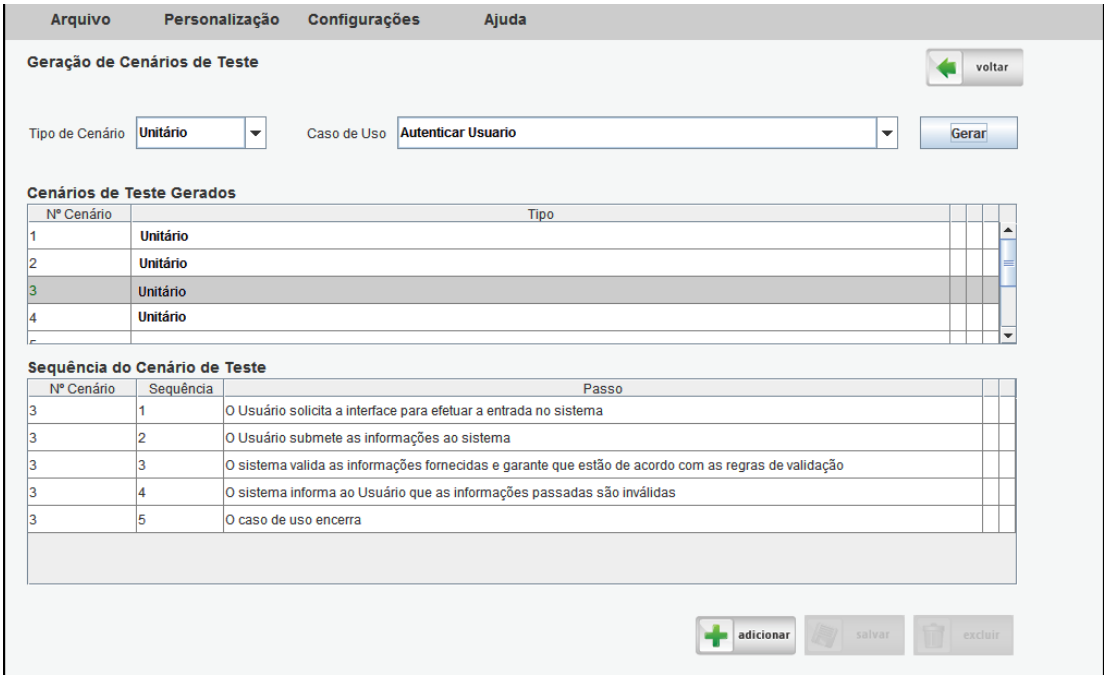


Figura 21: Exemplo de Geração de Cenário de Teste Unitário

6.2.2. Cenários de Teste de Integração

No contexto do ambiente COCAR, cenários de teste de integração referem-se ao exercício das dependências entre os casos de uso. Tais dependências devem ser definidas pelo usuário que estiver registrando os casos de uso no ambiente, indicando para um determinado caso de uso qual ou quais outros casos de uso devem ser executados anteriormente como pré-

condição para sua realização. Com essa informação, constrói-se uma estrutura lógica similar ao Diagrama de Dependência do método SCENT criado por Ryser & Glinz (1995).

Uma vez armazenadas todas as dependências entre os casos de uso na base de dados, inspirou-se tanto no trabalho de Carniello (2003a; 2003b) como no de Briand & Labiche (2002) para gerar grafos a partir de um diagrama de atividades que mostra a seqüência entre os casos de uso. No contexto deste trabalho, tal grafo é proveniente de tabelas e relacionamentos no banco de dados ao invés de um diagrama de atividades proposto por Briand & Labiche (2002) em seu trabalho.

Vale ressaltar que para um determinado caso de uso é preciso indicar apenas os casos de uso dos quais ele depende no nível imediatamente anterior. Para exemplificar, suponha que haja quatro casos de uso, denominados "1", "2", "3" e "4" respectivamente, cuja dependência de execução é mostrada no grafo da Figura 22.

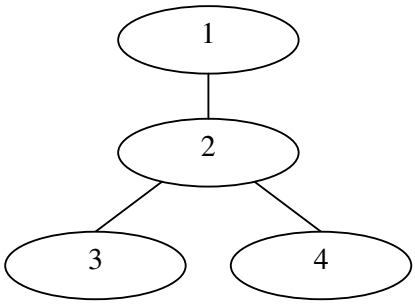


Figura 22: Grafo de Dependência de Casos de Uso

Nesse exemplo, a dependência do caso de uso "3" é somente o caso de uso "2"; a dependência do caso de uso"2" é o caso de uso "1"; e a dependência do caso de uso "4" é o caso de uso "2".

A informação sobre as dependências entre os casos de uso deve ser feita manualmente para todas as opções de cadastro dos casos de uso no COCAR, ou seja, usando a TUCCA, importando um arquivo XMI [OMG, 2010] ou cadastrando o caso de uso de forma manual. A Figura 25 mostra onde essa informação deve ser fornecida pelo usuário, que é no campo "Dependência".

O algoritmo inicia buscando todos os casos de uso que não possuem nenhuma dependência para sua execução, isto é, os casos de uso que são os vértices iniciais do grafo. Uma vez determinados esses vértices, para cada um deles é executada uma busca em

profundidade por meio de métodos recursivos. A opção <<include>> existente entre casos de uso é tratada como uma dependência e, quando utilizada, implica que ambos os casos de uso devem ser exercitados em conjunto, pois a execução de um implica na execução obrigatória do outro. Já a opção <<extend>> é tratada como opcional e, toda vez que um caso de uso possuir outro caso que o estenda, implica que mais um cenário de teste será gerado, um que exercita o caso de uso extensor, e outro que não o faz.

A Figura 23 mostra um exemplo de cenário de teste de integração gerado por meio do ambiente COCAR. Nesse exemplo foram gerados dois cenários de teste, os quais podem ser vistos na lista "Cenários de Teste Gerados". Ao clicar em um dos cenários gerados, os detalhes de sua execução são exibidos na lista "Seqüência do Cenário de Teste".

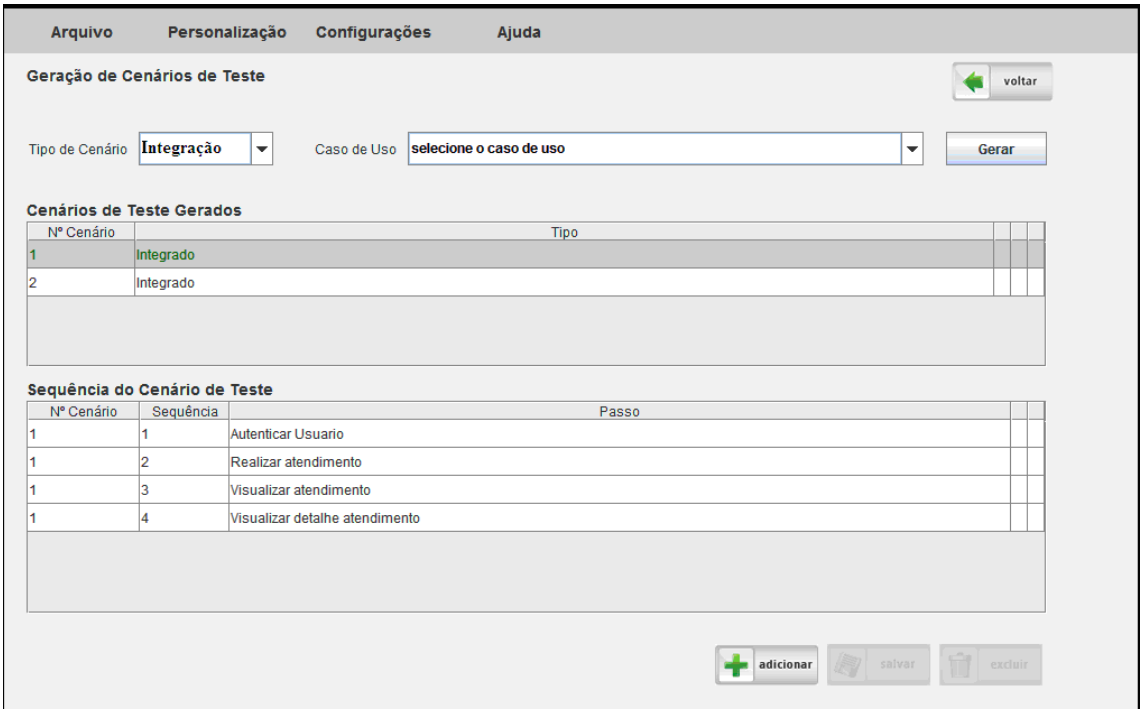


Figura 23: Exemplo de Cenário de Integração Gerado

6.3 Módulo de Inserção Manual de Casos de Uso

Esse módulo tem por objetivo aumentar a flexibilidade do ambiente e permitir que os usuários cadastrem casos de uso sem a necessidade de aplicar a técnica TUCCA, por meio de uma tela de cadastro. Salienta-se que, ao optar por essa forma, o usuário não pode utilizar

recursos como os módulos de Rastreabilidade de Requisitos ou Pontos por Caso de Uso (PCU), disponíveis no ambiente quando a TUCCA é aplicada para definição dos casos de uso. Ambos os módulos, de Rastreabilidade e de Pontos por Caso de Uso, ainda não foram convertidos para versão Desktop.

A inserção dos casos de uso é feita por meio do menu Personalização, com as opções "Cadastrar Caso de Uso", "Cadastrar Curso Normal" e "Cadastrar Curso Alternativo", como mostra a Figura 24, as quais são comentadas em seguida.

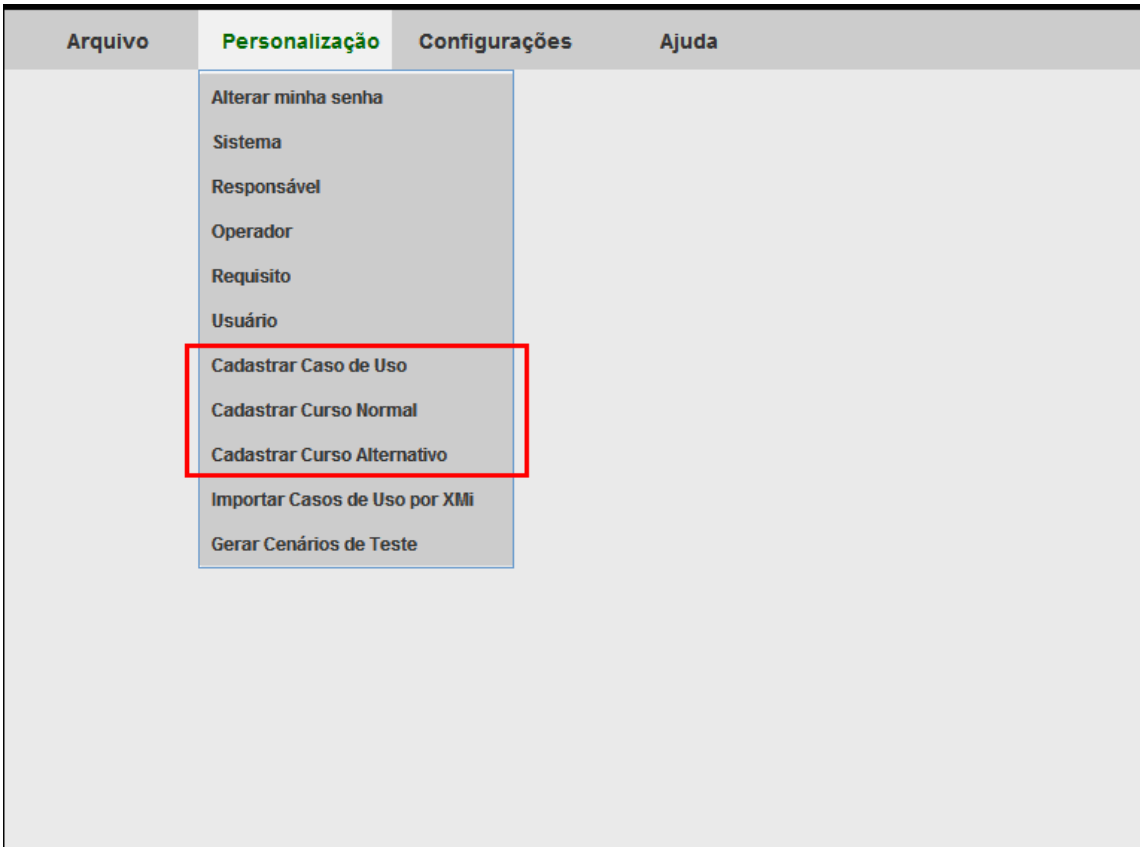


Figura 24: Opções do Menu para acesso ao cadastro manual de Caso de Uso

6.3.1. Cadastrar Caso de Uso

A opção "Cadastrar Caso de Uso" abre a tela para inserção do cabeçalho do caso de uso, com as informações necessárias para armazenamento no banco de dados. Os campos são os mesmos que o da versão anterior Web do ambiente COCAR após a aplicação da TUCCA, com o acréscimo do campo "Dependência". São eles:

- Nome: nome do caso de uso que será inserido.
- Descrição: breve descrição do caso de uso para entendimento de seu objetivo.
- Pré-Condição: condições necessárias para que o caso de uso possa ser utilizado.
- Disparador: indica qual o evento disparador do caso de uso, se houver.
- Ator: indica o ator relacionado ao caso de uso.
- Include: indica os casos de uso incluídos no caso de uso principal.
- Extend: indica os casos de uso estendidos pelo caso de uso principal.
- Dependência: indica os casos de uso que devem ser executados anteriormente ao caso de uso atual.
- Autor: indica o autor do caso de uso.
- Versão: aponta a versão atual do caso de uso.

A tela de cadastro de Caso de Uso é mostrada na Figura 25.

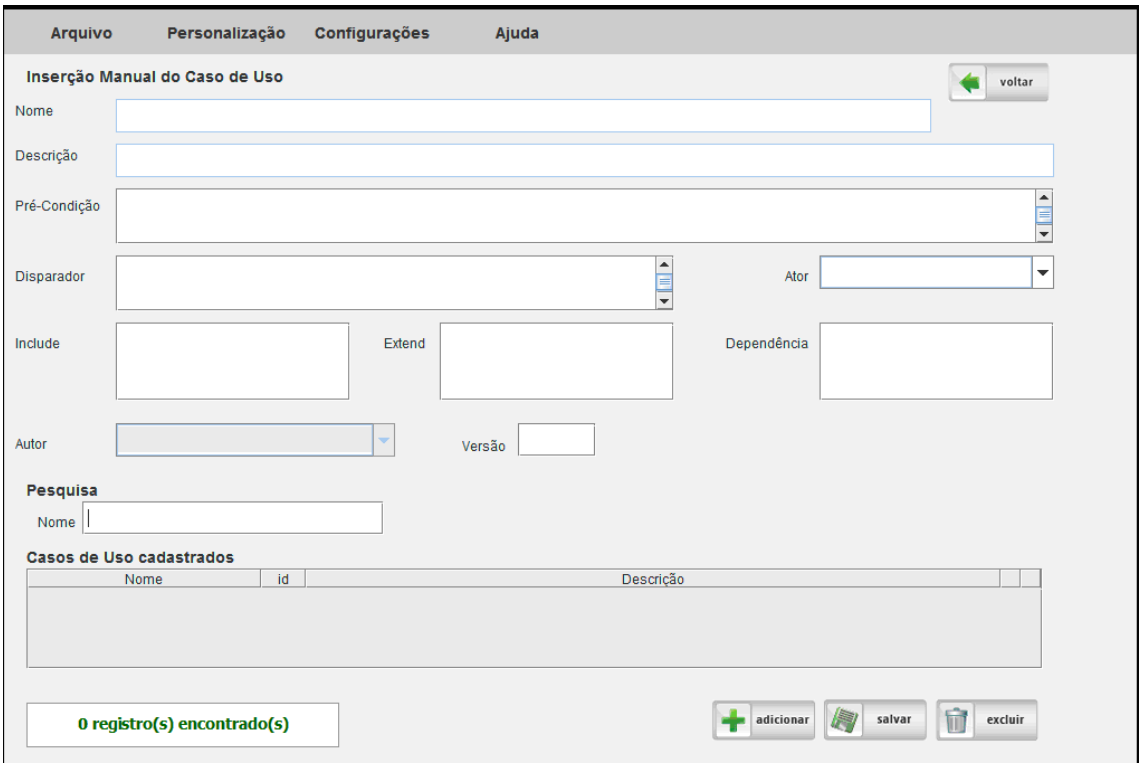


Figura 25: Tela para cadastro manual de Caso de Uso

6.3.2. Cadastrar Curso Normal

Uma vez registradas as informações de cabeçalho do caso de uso, o próximo passo é o cadastro de seu curso normal.

A tela de cadastro do curso normal do caso de uso inicia com os campos desabilitados, permitindo apenas consulta. Quando o usuário clica no botão "adicionar", todos os campos da tela tornam-se ativos, e o primeiro passo é selecionar um dos casos de uso cadastrados anteriormente no sistema. Se o caso de uso estiver somente com o cabeçalho cadastrado, nenhum passo do curso normal é exibido. Porém, se já há passos cadastrados, são mostrados em forma de lista na tela, permitindo alteração.

Além do nome do caso de uso que o usuário deve selecionar na lista de casos de uso disponíveis para o sistema, é preciso informar a seqüência do passo a ser cadastrado, sua descrição, e indicar se o passo possui um curso alternativo ou não, por meio da caixa de seleção "Tem alternativo?", que permite as opções Sim e Não.

Ao clicar no botão "Salvar", a lógica do programa entende se é uma inserção de um novo passo, ou alteração de um passo existente. Na Figura 26 é mostrada a tela de cadastro de curso normal.

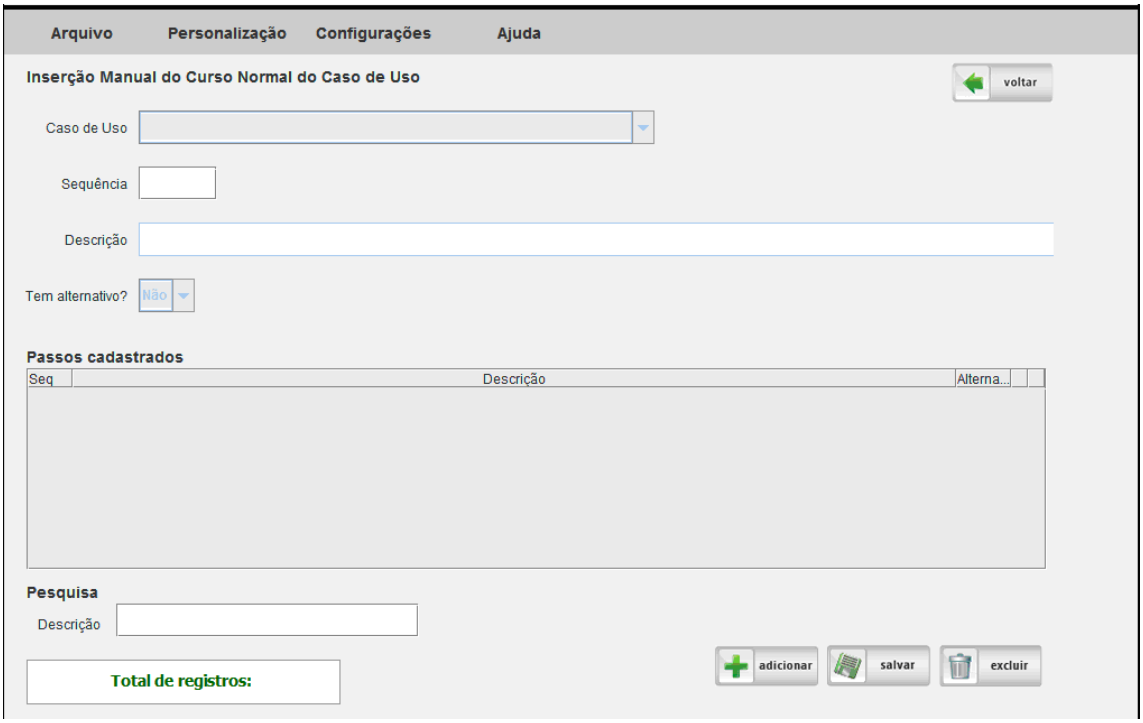


Figura 26: Tela de Cadastro de Curso Normal do Caso de Uso

6.3.3. Cenários de Teste de Integração

A tela de cadastro do curso alternativo também inicia com os campos desabilitados, em modo consulta. Assim como na tela anterior, é preciso que o usuário clique no botão "adicionar" para que todos os campos da tela tornem-se ativos, e o primeiro passo é selecionar um dos casos de uso cadastrados anteriormente no sistema. Ao selecionar um caso de uso, os passos de seu curso normal são automaticamente carregados para a lista nomeada "Curso Normal", para que o usuário clique sobre o passo que deseja incluir um curso alternativo. Se o passo do curso normal selecionado já possuir um ou mais cursos alternativos, os mesmos são carregados para a lista "Curso Alternativo da Seqüência do Caso de Uso", permitindo alteração. Caso contrário, os passos do novo curso alternativo são registrados, sendo necessário informar a ordem do curso alternativo para o passo do curso normal escolhido (pois um passo do curso normal pode ter mais de um curso alternativo), sua seqüência dentro do curso alternativo e sua descrição.

Ao clicar no botão “Salvar”, a lógica do programa entende se é uma inserção de um novo passo, ou alteração de um passo existente. Na Figura 27 é mostrada a tela de cadastro de curso alternativo.

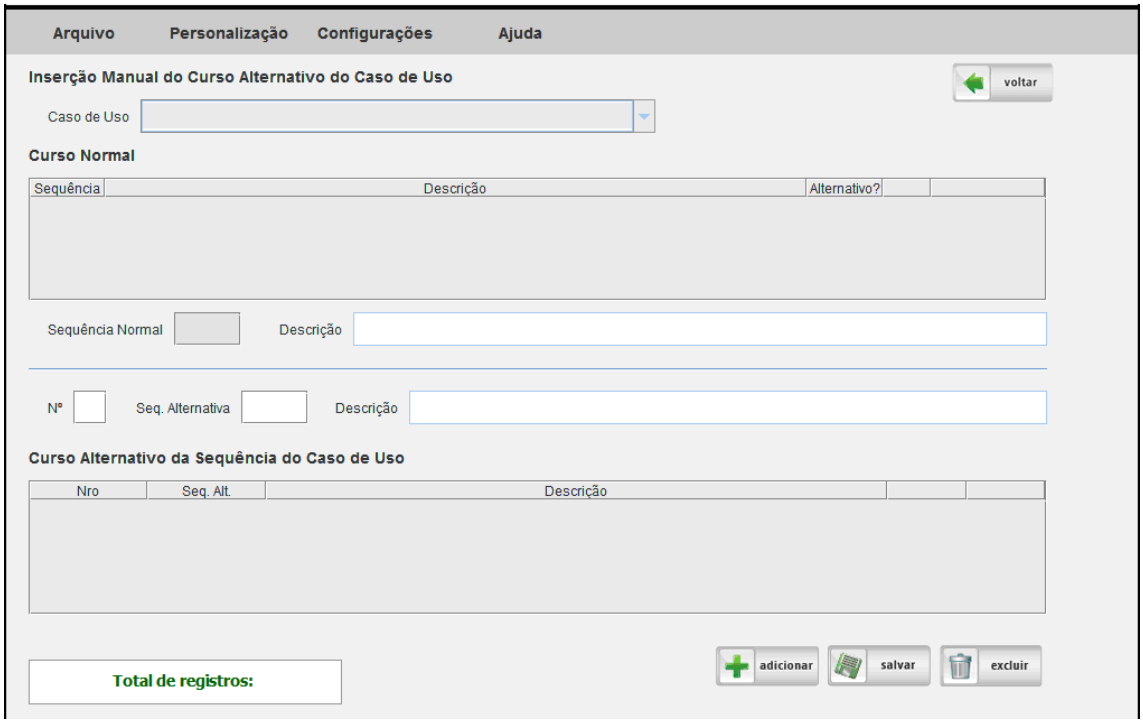


Figura 27: Tela de Cadastro de Curso Alternativo do Caso de Uso

Cadastrados os casos de uso, com os seus respectivos cursos normal e alternativos no ambiente COCAR, é possível utilizar o módulo de Geração de Cenários de Teste com base em Casos de Uso, que é o maior objetivo deste trabalho.

Na próxima seção apresenta-se como importar no ambiente COCAR um caso de uso criado em outra ferramenta.

6.4. Integração do COCAR com Outras Ferramentas

O objetivo dessa funcionalidade é fazer com que usuários que utilizem outras ferramentas de modelagem de casos de uso e queiram apenas gerar cenários de teste por meio do ambiente COCAR, tenham essa possibilidade.

As principais ferramentas pelas quais é possível a modelagem de casos de uso possuem a opção de exportação dos casos de uso gerados para arquivo no padrão XMI (XML Metadata Interchange) [OMG, 2010]. Entretanto, não há uma padronização entre os arquivos XMI gerados pelas diferentes ferramentas, o que torna esta integração mais complexa e exige o tratamento das particularidades para cada uma delas.

A utilização do padrão XMI permitiu que se criasse uma lógica pela qual fosse possível capturar o nome do caso de uso, bem como os passos de seus cursos normal e alternativos e inseri-los na base de dados, possibilitando assim a utilização do módulo de Geração de Cenários de Teste, detalhado na Seção 6.2.

Para mostrar a viabilidade da integração dessas ferramentas disponíveis com o ambiente COCAR, fez-se isso para uma delas, sendo que para as outras um procedimento similar deve ser realizado. A ferramenta selecionada foi a ArgoUML [ArgoUML, 2010], que está entre as ferramentas mais utilizadas pela comunidade, além de possuir o recurso de exportação do modelo de caso de uso para arquivo XMI gratuitamente, o que não ocorre com a ferramenta Astah [Astah, 2010], antigamente denominada Jude, na qual a opção de exportar modelos para arquivo XMI é permitida somente na versão paga da ferramenta.

Após a opção pela ferramenta ArgoUML, realizaram-se estudos para analisar a estrutura do arquivo XMI gerado com o objetivo de construir um algoritmo que permitisse a importação das informações contidas no arquivo XMI para o ambiente COCAR. Para tanto, foram analisadas as "tags" do arquivo XMI, bem como a maneira como as informações necessárias para compor os dados do caso de uso no COCAR eram armazenadas neste formato de arquivo gerado pela ArgoUML. Verificou-se que o nome do caso de uso é armazenado na "tag" <UML:UseCase>.

No entanto, quanto aos cursos normal e alternativos, não é possível identificá-los no arquivo XMI, uma vez que na ArgoUML eles podem ser escritos como um campo texto sem qualquer formatação. Dessa forma, para eles, é necessário estabelecer um padrão de descrição para ser usado na ArgoUML, de forma que eles possam ser recuperados e armazenados no COCAR.

Assim, foram adotadas as seguintes regras que tornam a integração com o ambiente possível:

- Todos os passos, sejam eles do curso normal ou dos alternativos, devem ser inseridos na aba "Documentação", campo "Documentação" da ferramenta ArgoUML.
- O início do curso normal deve ser determinado com o uso das palavras "Curso Normal".
- Para cada curso alternativo existente, demarcar com as palavras "Curso Alternativo", podendo ou não ser sucedidos pelo número que representam, como por exemplo, "Curso Alternativo 1", "Curso Alternativo 2". Cada vez que o algoritmo encontrar as palavras "Curso Alternativo" entenderá que se trata de um novo curso alternativo.
- Todos os passos tanto do curso normal quanto dos alternativos devem terminar com o uso da tecla <ENTER>, pois o algoritmo entenderá o código ASCII que representa a tecla <ENTER> como o final do passo. Desta maneira, cada passo deve ser sempre descrito numa única linha.
- O formato de numeração para o curso normal deve ser o número correspondente ao passo, seguido de um ponto, como por exemplos: 1. ou 2.
- O formato de numeração para os cursos alternativos deve ser o número correspondente ao passo do curso normal a que ele se refere, seguido de um ponto, seguido pela seqüência numérica dentro do curso alternativo, seguido de outro ponto, como por exemplos: 2.1. e 2.2., que indicam que o curso alternativo importado refere-se ao passo 2 do curso normal, e tem as seqüências 1 e 2.

Na Figura 28 é mostrado como devem ser descritos na ArgoUML os cursos normal e alternativos, conforme as regras estabelecidas anteriormente. Ela ilustra o caso de uso "Autenticar Usuário", que possui o curso normal e um curso alternativo para o passo 3 do curso normal com 2 seqüências de execução.

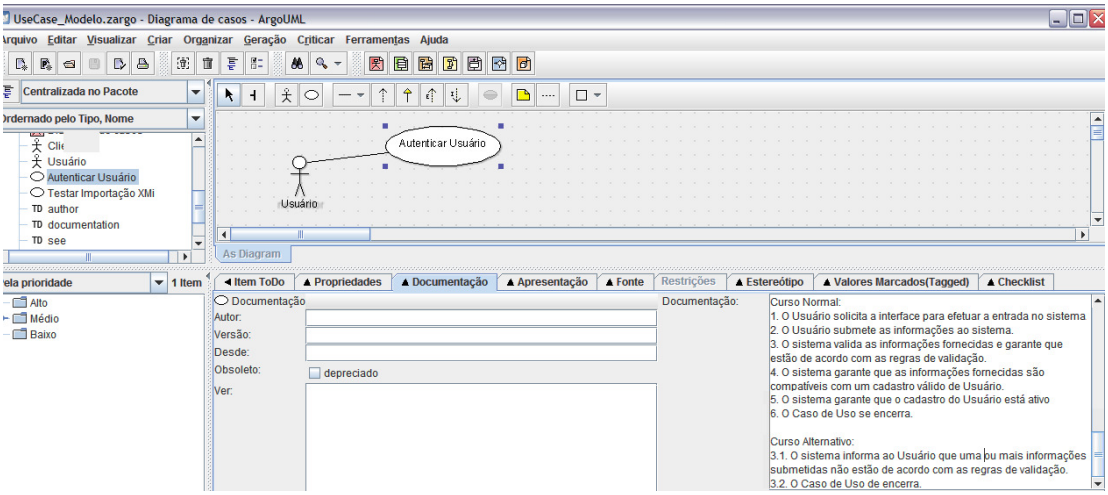


Figura 28: Tela de Modelagem do Caso de Uso na ferramenta ArgoUML

Uma vez modelado o caso de uso na ferramenta *case* seguindo as regras estabelecidas e exportado para arquivo no formato XMI, a próxima etapa é importar este arquivo utilizando uma tela do ambiente COCAR. Para que isso fosse possível, um algoritmo foi criado para entender as informações necessárias dentro do arquivo e carregá-las numa tela do ambiente e importá-las para o banco de dados. O algoritmo possui uma lógica estabelecida, cujos principais pontos são:

- As linhas do arquivo XMI são lidas uma a uma seqüencialmente.
- Ao encontrar a tag “<UML:UseCase>”, o algoritmo entende que esta contém o nome do caso de uso. Dessa forma, busca pela propriedade “Name” da tag e então obtém o nome do caso de uso.
- Em seguida é feita a busca dos passos dos cursos normal e alternativos, que está na tag “<UML:TaggedValue.dataValue>” seguinte à tag “<UML:UseCase>. O campo “dataValue” contém todos os passos para o caso de uso e, caso as regras anteriores de escrita dos cursos do caso de uso tenham sido corretamente seguidas, é possível separar os passos entre o curso normal e os cursos alternativos, importando para a tela e também para as tabelas corretas do banco de dados pelo COCAR.

Um exemplo de arquivo XMI gerado para o caso de uso “Autenticar Usuário” exibido na Figura 28 é apresentado na Figura 29.

```
- <UML:UseCase xmi.id="9-8-3-116--3c1f6040:12da42b81ec:-8000:0000000000000A73" name="Autenticar Usuário"
  isSpecification="false" isRoot="false" isLeaf="false' isAbstract="false">
  - <UML:ModelElement.taggedValue>
    - <UML:TaggedValue xmi.id="9-8-3-116--3c1f6040:12da42b81ec:-8000:0000000000000A7C" isSpecification="false">
      - <UML:TaggedValue.dataValue>
          Curso Normal: 1. O Usuário solicita a interface para efetuar a entrada no sistema. 2. O Usuário submete as informações
          ao sistema. 3. O sistema valida as informações fornecidas e garante que estão de acordo com as regras de validação.
          4. O sistema garante que as informações fornecidas são compatíveis com um cadastro válido de Usuário. 5. O sistema garante
          que o cadastro do Usuário está ativo 6. O Caso de Uso se encerra. Curso Alternativo: 3.1. O sistema informa ao Usuário
          que uma ou mais informações submetidas não estão de acordo com as regras de validação. 3.2. O Caso de Uso de encerra.
        </UML:TaggedValue.dataValue>
      - <UML:TaggedValue.type>
          <UML:TagDefinition xmi.idref="9-8-2--43-707c9f14:12ad912f34f:-8000:000000000000086F"/>
        </UML:TaggedValue.type>
      </UML:TaggedValue>
    </UML:ModelElement.taggedValue>
  </UML:UseCase>
```

Figura 29: Exemplo de arquivo XMI em que foram seguidas as regras estabelecidas para integração da ArgoUML com o COCAR

No ambiente COCAR foi implementada uma tela que permite a importação do arquivo XMI gerado pela ferramenta de modelagem. Nesta tela há um botão que, quando pressionado, abre uma caixa de diálogo permitindo ao usuário selecionar o arquivo XMI que deseja utilizar para o caso de uso, além do botão “Importar”, que invoca o algoritmo responsável pela leitura do arquivo e aplicação de toda a lógica de separação do nome do caso de uso e de seu curso normal e cursos alternativos. A tela de importação de arquivos XMI para o ambiente COCAR é mostrada na Figura 30.

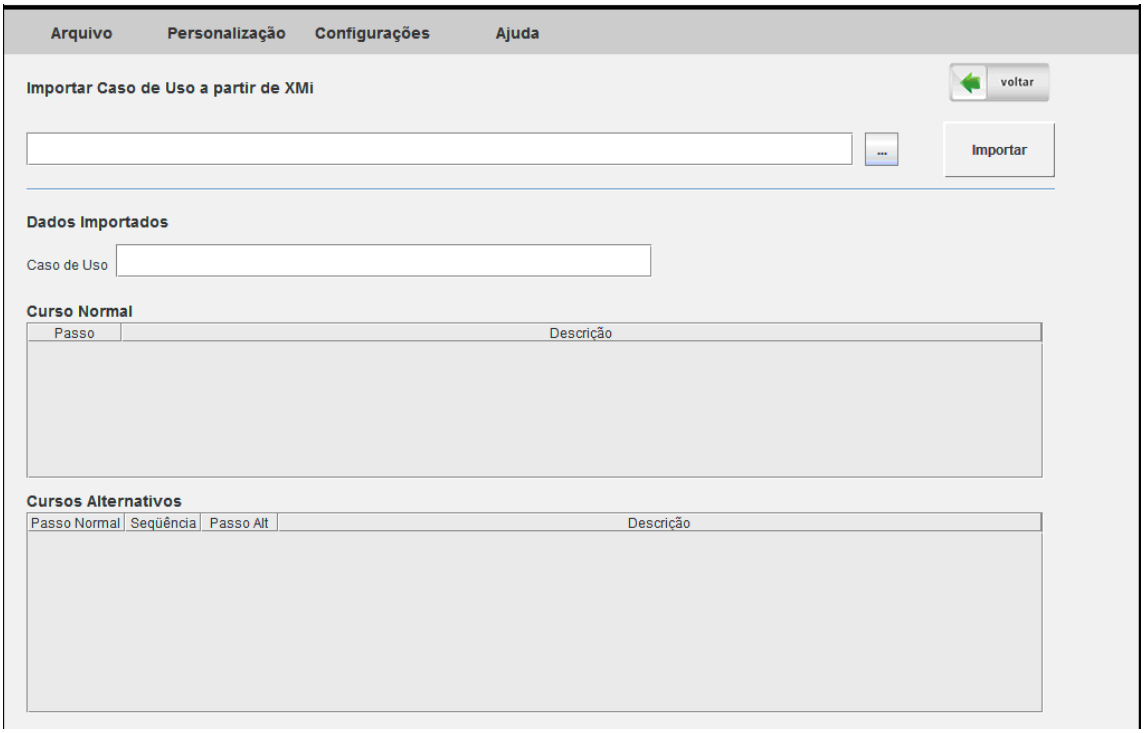


Figura 30: Tela de Importação de Caso de Uso por Arquivo XMI

A ferramenta ArgoUML permite a modelagem de casos se uso que se relacionam por meio das opções <<include>> e <<extend>>, porém não há a possibilidade de descrever a ordem de execução ou dependência, que é importante no contexto deste trabalho para geração de cenários. Essa dependência da execução de um caso de uso em relação a outro pode ser definida na tela de cadastro de casos de uso detalhada na Seção 6.2.1 deste capítulo.

Uma vez importadas as informações do caso de uso para o COCAR, o usuário deve acessar a tela de cadastro manual do caso de uso, descrita na seção anterior, para complementar as informações referentes ao caso de uso importado, tais como suas pré-condições, autor, versão e dependências em relação a outros casos de uso. Decidiu-se, então, que o usuário deve informar os casos de uso incluídos ou estendidos ao caso de uso importado, se houver, nessa mesma tela de cadastro do caso de uso, já que o usuário terá de acessá-la para registrar as informações complementares do caso de uso.

6.5. Considerações Finais

Neste capítulo apresentou-se o módulo de Geração de Cenários de Teste com base em Casos de Uso no ambiente COCAR, o qual consiste na maior contribuição deste trablho. Esse módulo foi desenvolvido principalmente com base nos trabalhos de Ryser & Glinz (1995; 1999a; 1999b; 1999c; 2000), de Briand & Labiche (2002) e de Carniello (2003a; 2003b), cujas idéias também puderam ser observadas nos trabalhos de Hartmann et al. (2005), Vieira et al. (2006) e Nebut et al. (2006). O objetivo desse módulo é gerar cenários de teste unitários e de integração, servindo de apoio à elaboração do plano de teste e de seus respectivos casos de teste, além de auxiliar na inspeção do próprio plano de teste, quando já existente, e de outros artefatos, como a especificação de casos de uso.

Outros dois módulos foram implementados no ambiente: o de Inserção Manual de Casos de Uso e o de Integração do COCAR com outras ferramentas. O primeiro permite que os usuários cadastrem casos de uso sem a necessidade de aplicar a técnica TUCCA, por meio de uma tela de cadastro. O outro permite que usuários que utilizem outras ferramentas de modelagem de casos de uso importem casos de uso para o ambiente por meio de arquivo XMI, gerado na ferramenta ArgoUML. Esses módulos têm por objetivo possibilitar que usuários que já possuem sistemas documentados, com base em casos de uso, gerem cenários de teste por meio do ambiente COCAR.

O próximo capítulo é dedicado ao estudo de caso para análise dos resultados obtidos com essas novas funcionalidades implementadas no ambiente COCAR.

CAPÍTULO 7

Estudo de Caso

O objetivo deste capítulo é apresentar um estudo de caso explorando as funcionalidades implementadas no ambiente COCAR, resultantes deste trabalho de mestrado, sendo que a principal delas é o módulo de Geração de Cenários de Teste com base em Casos de Uso. Esse estudo de caso usou para comparação com o trabalho desenvolvido dados de teste de uma empresa real e avaliou as contribuições de se conduzir o teste com o suporte do ambiente COCAR.

7.1. Considerações Iniciais

Como mencionado por Basili et al (1996b), toda proposta apresentada, seja ela referente a técnicas, métodos, ferramentas, etc., deve ser avaliada para dar subsídios aos desenvolvedores de software quando estes precisam escolher algum desses recursos para usar na prática. Assim, procurou-se, neste trabalho, conduzir uma avaliação das funcionalidades que foram adicionadas ao ambiente COCAR, de forma a poder caracterizar possíveis contribuições e limitações.

Conduziu-se então um estudo de caso e buscou-se para tanto, um exemplo real de um sistema que possui toda a documentação de casos de uso, bem como dos casos de teste instanciados e executados pela equipe de desenvolvimento por ocasião de sua implementação. O sistema é um software público e pode ser encontrado em [Software Público, 2010].

A documentação desse sistema, fornecida pela empresa, que por solicitação será aqui referenciada por Empresa X, foi acessada por meio do endereço http://www.softwarepublico.gov.br, solicitando-se acesso ao nome do sistema informado pela empresa, e tendo o acesso aprovado por seus administradores.

A documentação à qual se teve acesso é composta pelo diagrama de casos de uso, suas respectivas especificações, uma tabela com os atores e o plano de teste. O número total de casos de uso existentes na documentação é quarenta e oito, sendo que foram selecionados doze casos de uso para este estudo de caso, o que corresponde a 25% do total.

Em relação à documentação dos casos de teste, esta é sigilosa e somente foi obtido acesso depois de estabelecer diálogo com gerente da área de Tecnologia da Informação da Empresa X, explicando que seria para uso de pesquisa. O documento com o plano de teste e todos os casos de teste instanciados e executados foi liberado mediante ao compromisso de manter sigilo em relação ao nome da empresa e dos dados de casos de teste que pudessem se referir diretamente ao seu nome. Como toda essa documentação é extensa, parte dela está exemplificada no Anexo I.

A especificação dos casos de uso obedece a um *template* que contém as informações essenciais para caracterização de casos de uso, incluindo as informações necessárias para serem usados no ambiente COCAR. O plano de teste contém os casos de teste instanciados para cada caso de uso existente, exibindo o nome e descrição do caso de teste, as pré-condições necessárias para sua execução, seus respectivos passos de execução e os resultados esperados.

Ressalta-se que como os casos de uso já existem e são de um sistema em ambiente de produção, não seria possível o uso da TUCCA para construção dos casos de uso. Assim sendo, e com a documentação das especificações dos casos de uso do sistema, a inserção dos casos de uso no ambiente COCAR deu-se por meio do módulo de cadastro manual de casos de uso, fruto deste trabalho. Como não havia acesso aos arquivos XMI dos casos de uso do sistema, apenas um dentre os casos de uso existentes na documentação foi modelado na ferramenta ArgoUML e gerado arquivo no formato XMI para ser importado no ambiente COCAR e mostrar o funcionamento dessa nova funcionalidade de importação de casos de uso a partir de arquivo XMI.

Assim, o objetivo deste capítulo é fazer uma primeira avaliação do que foi desenvolvido neste mestrado, dando-se ênfase aos módulos que foram implementados e descritos no Capítulo 6. O capítulo está organizado da seguinte forma: a Seção 7.2 mostra um exemplo de como casos de uso podem e devem ser inseridos no ambiente COCAR, primeiramente por uso do módulo de inserção manual de casos de uso, e depois por meio de importação de arquivo XMI modelado na ferramenta case ArgoUML. A Seção 7.3 mostra um exemplo de como cenários de teste unitários e de integração podem ser gerados para o sistema utilizando o novo módulo implementado de Geração de Cenários de Teste com base em Casos de Uso. A Seção 7.4 traz os resultados obtidos com o estudo de caso e as análises feitas em relação à modelagem real do sistema e à gerada pelo ambiente COCAR. Por fim a Seção 7.5 mostra as considerações finais.

7.2. Como Gerar Cenários de Teste no Ambiente COCAR

A geração de cenários de teste com base em casos de uso armazenados no ambiente COCAR foi o principal foco deste trabalho de mestrado. O pré-requisito para que essa funcionalidade seja utilizada é que os casos de uso estejam devidamente armazenados no ambiente e que eles estejam interligados entre si com base na dependência lógica de execução. Vale lembrar que os casos de uso podem ser gerados no ambiente por meio da aplicação da técnica TUCCA ou podem ser inseridos manualmente ou por importação de arquivo XMI.

Há dois tipos de cenários de teste que podem ser gerados no ambiente COCAR: unitários ou de integração. Os cenários de teste unitários referem-se a um único caso de uso, percorrendo-se os passos de seu curso normal e de todos os seus cursos alternativos. Os cenários de teste de integração exercitam a dependência entre os casos de uso, seja ela por *include*, *extend*, ou devido à dependência lógica, isto é, à precedência entre as funcionalidades modeladas em cada caso de uso.

O módulo de geração de cenários de teste no ambiente pode ser acessado por meio do menu “Personalização”, opção “Gerar Cenários de Teste”.

7.2.1. Geração de Cenários de Teste Unitários

Ao acessar a tela de geração de cenários de teste, o primeiro passo é clicar no botão “adicionar” para habilitar a lista de seleção de tipo de cenário. Nessa lista escolhe-se a opção “Unitário” e é preciso, então, selecionar o caso de uso para o qual se deseja gerar cenários de teste. Todos os casos de uso referentes ao sistema são exibidos na lista “Caso de Uso”. Selecionados o tipo “Unitário” e o nome do caso de uso desejado, basta clicar no botão “Gerar”.

O exemplo escolhido para geração de cenários de teste é o caso de uso documentado pela Empresa X chamado “Realizar Atendimento”. Esse caso de uso possui um curso normal e dois cursos alternativos, sendo um para o passo 2 e outro para o passo 3 do curso normal.

Ao clicar no botão “Gerar” foram gerados três cenários de teste para esse caso de uso. O primeiro corresponde ao seu curso normal, e os demais exercitam os cursos alternativos,

correspondendo a cenários de teste para cada curso alternativo. Os cenários resultantes são exibidos na lista "Cenários de Teste Gerados" e, ao clicar em cada um dos cenários gerados, seus passos de execução são exibidos na lista "Seqüência do Cenário de Teste".

A Figura 31 mostra os três cenários de teste unitários gerados para o caso de uso "Realizar Atendimento", além dos passos de execução do cenário número 3 que foi selecionado na lista de cenários de teste gerados.

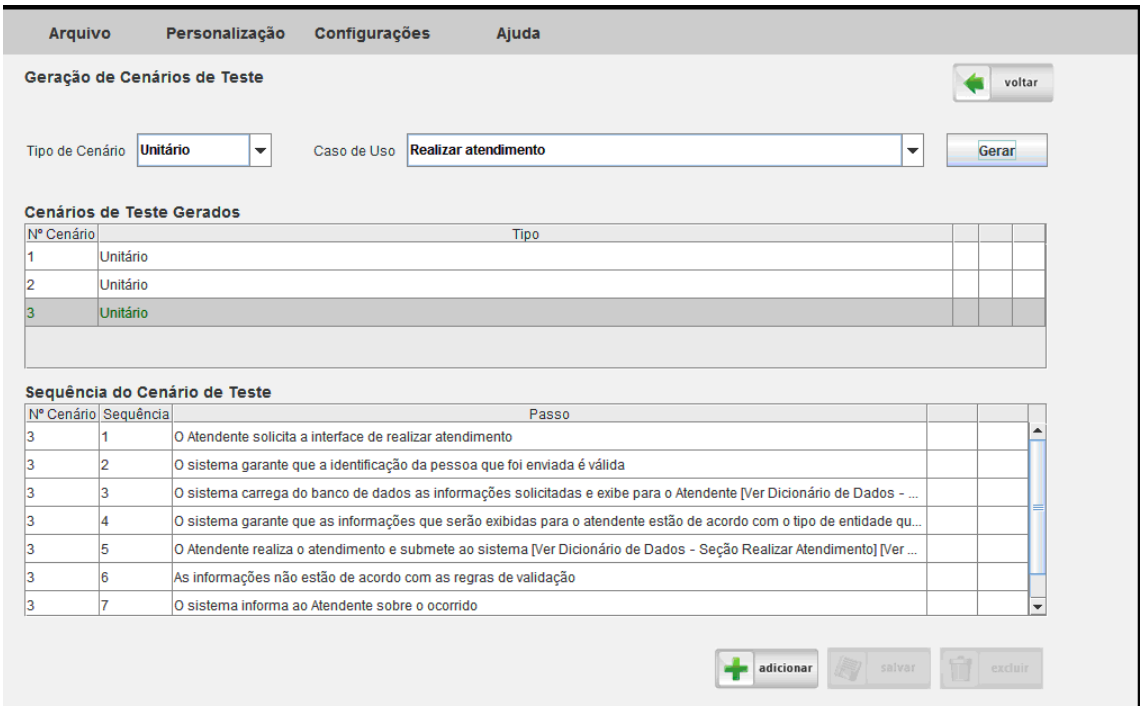


Figura 31: Cenários de Teste Unitários gerados pelo COCAR para o Caso de Uso "Realizar Atendimento"

7.2.2. Geração de Cenários de Teste de Integração

Para gerar cenários de teste de integração, primeiro é preciso clicar no botão "adicionar" da tela de geração de cenários de teste e depois escolher a opção "Integração" na lista de seleção "Tipo de Cenário". Como o cenário de teste é de integração não é preciso selecionar nenhum caso de uso na lista de seleção "Caso de Uso", bastando apenas clicar no botão "Gerar".

Ao clicar no botão “Gerar”, as dependências de precedência existentes entre os casos de uso são identificadas e os possíveis cenários de integração são exibidos na lista “Cenários de Teste Gerados”. Para visualizar os detalhes de um cenário de teste de integração, basta clicar em qualquer um dos cenários gerados e seus passos de execução, isto é, a seqüência em que as funcionalidades dos casos de uso devem ser exploradas é exibida na lista “Seqüência do Cenário de Teste”.

Para o exemplo a seguir, foram inseridos alguns casos de uso existentes na documentação fornecida pela Empresa X. Feita a devida amarração de dependência entre eles, geraram-se os possíveis cenários de teste de integração, num total de 7 cenários.

A Figura 32 mostra os cenários de teste de integração gerados para os casos de uso inseridos no ambiente COCAR, bem como os detalhes de execução do cenário número 1, o qual indica que a seqüência em que os casos de uso devem ser exercitados é “Autenticar Usuário”, “Realizar Atendimento”, “Visualizar Atendimento”, “Visualizar Detalhe do Atendimento”.

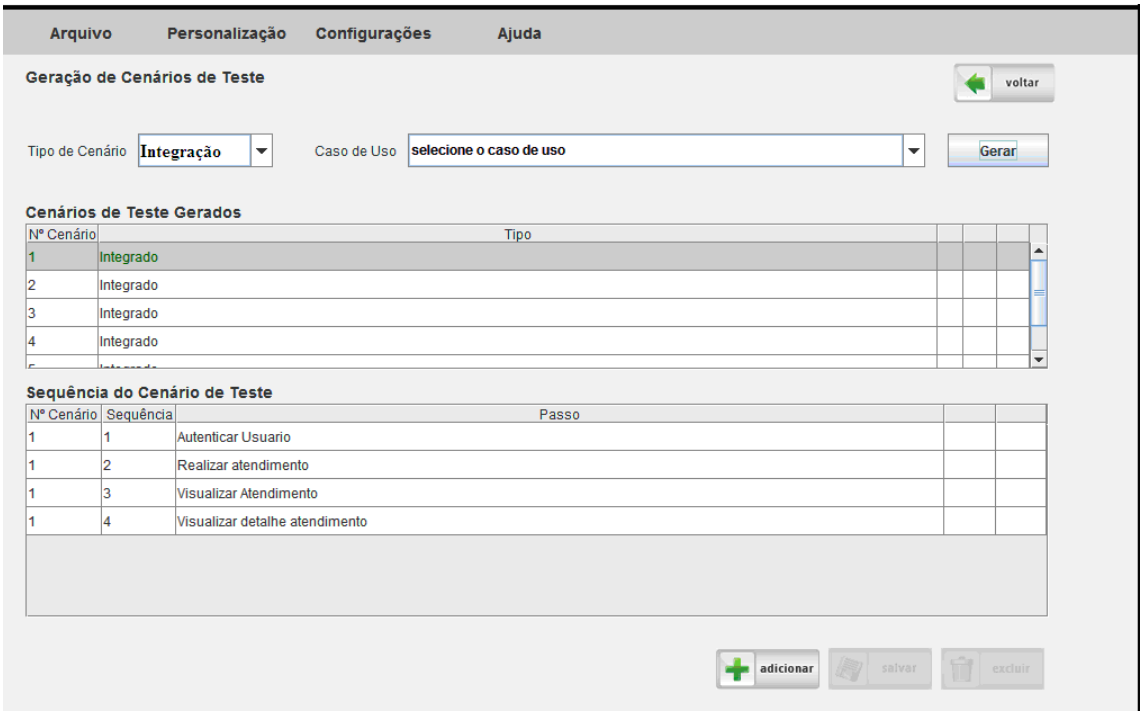


Figura 32: Cenários de Teste de Integração gerados pelo COCAR para os Casos de Uso registrados no ambiente

7.3. Inserindo no Ambiente COCAR Casos de Uso desenvolvidos externamente a ele

O objetivo desta seção é mostrar como um caso de uso já existente e documentado externamente ao ambiente COCAR pode ser inserido nele. Essa inserção pode ser feita por meio do módulo de cadastro manual de caso de uso, se o desenvolvedor não tiver utilizado ferramentas de apoio, ou por meio de arquivo XMI, caso tenha sido usada alguma ferramenta que gere esse tipo de arquivo. Lembra-se que este último caso foi implementado no COCAR tomando-se como referência a ferramenta ArgoUML. Nas seções seguintes exemplificam-se essas duas possibilidades com base em um caso de uso da documentação fornecida pela Empresa X.

7.3.1. Inserção Manual do Caso de Uso

Essa opção deve ser usada quando já existe um modelo de caso de uso, mas ele não foi desenvolvido com o suporte de nenhuma ferramenta específica para isso, como por exemplo a ArgoUML, que já possui a funcionalidade de importação de caso de uso por meio de arquivo XMI implementada no ambiente.

O primeiro passo é acessar a opção “Cadastrar Caso de Uso” no menu “Personalização”, quando será aberta uma tela para cadastrar o caso de uso, o que é feito pelo botão “adicionar”, que entra em modo de inserção.

A Figura 33 mostra um exemplo de como os casos de uso foram documentados pela Empresa X. O exemplo refere-se ao caso de uso “Visualizar Atendimento”. Com base nessa documentação é que este e outros casos de uso foram inseridos no ambiente COCAR.

Visualizar Atendimento

Histórico de Revisão

Data	Versão	Descrição	Responsáveis pela Versão
29/01/2008	1	Criação do documento	Jefferson Lima – W3S

1 Descrição

Esse caso de uso descreve uma ação concreta (funcionalidade do sistema) que permite ao Atendente visualizar as informações de atendimento.

2 Atores

Ator	Descrição
Atendente	Usuário com atribuições de grupo: Técnico ou Coordenador da Entidade.

3 Pré-condições

Pré-condições
O Atendente deve estar autenticado e ter permissão para visualizar atendimento. O Atendente deve ter feito o procedimento de busca de usuários e selecionado um para a visualização das informações.

4 Fluxo Principal

1. O Atendente solicita a interface de visualização de atendimento.
2. O sistema garante que a identificação da pessoa que foi enviada é válida.
3. O sistema carrega do banco de dados as informações solicitadas e as exibe para o Atendente **[Ver Dicionário de Dados – Seção Visualizar Atendimento].**
4. O caso de uso encerra.

Figura 33: Parte da especificação do Caso de Uso “Visualizar Atendimento” que foi inserido manualmente no COCAR

As informações contidas na documentação são suficientes para o ambiente, pois possui o nome do caso de uso, sua descrição, pré-condições e atores relacionados. Essas informações foram inseridas na tela de cadastro de caso de uso, como é mostrado na Figura 34. Como não há casos de uso incluídos ou estendidos para o caso de uso “Visualizar Atendimento”, conforme a documentação da Figura 33, os campos “Include” e “Extend” da tela, devem permanecer nulos. O campo “Dependência”, que serve para fazer a amarração com os casos de uso que devem ser executados anteriormente ao caso de uso que está sendo inserido, é de fundamental importância para os cenários de teste de integração. Assim, caso haja dependência do caso de uso em questão com algum outro, essa informação deve ser registrada nesse campo. No exemplo do caso de uso “Visualizar Atendimento”, o caso de uso “Realizar Atendimento” deve ser selecionado, uma vez que somente é possível visualizar um atendimento se o mesmo já foi realizado. O ator principal do caso de uso, segundo os diagramas da Empresa X, apresentados no Anexo I, é o ator “Usuário Técnico da Entidade”.

Além disso, as informações sobre o autor e a versão do caso de uso devem ser registradas nessa tela. Para terminar a inserção, deve-se clicar no botão “salvar”.

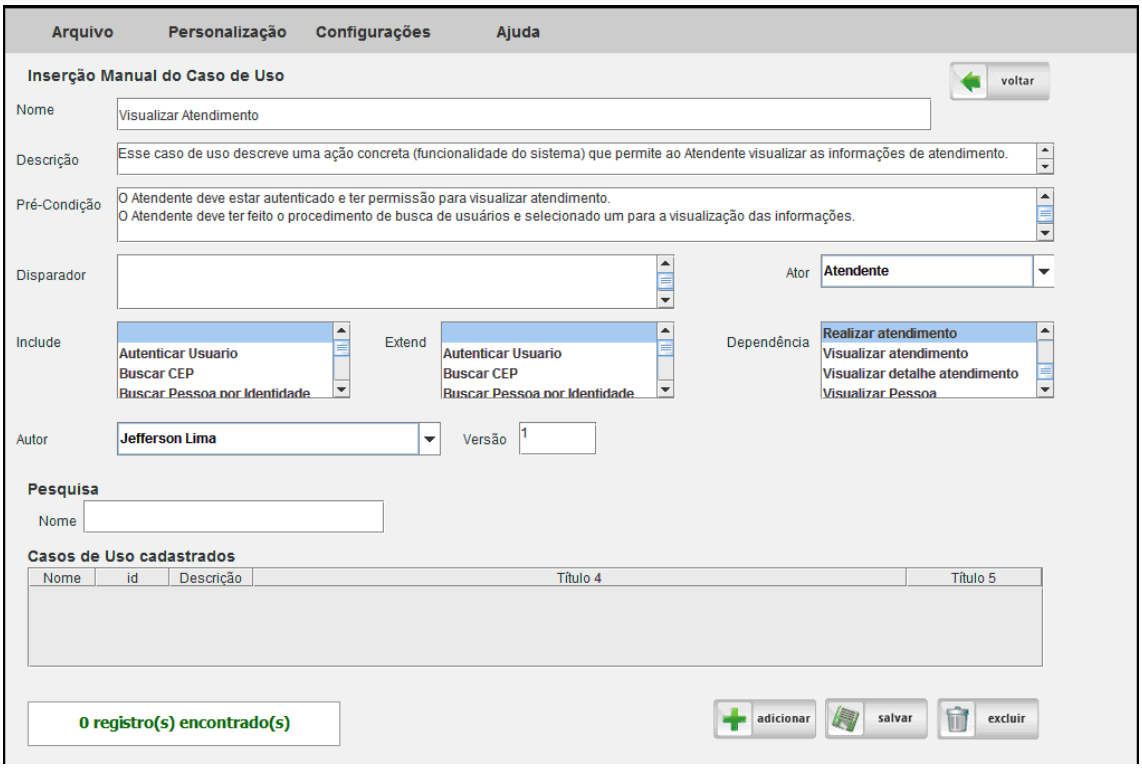


Figura 34: Tela do COCAR usada para inserção manual de Caso de Uso

Uma vez cadastradas todas as informações sobre o cabeçalho do caso de uso, a etapa seguinte é registrar os passos de seu curso normal. Ao escolher a opção “Cadastrar Curso Normal” no menu “Personalização”, a tela de cadastro do curso normal do caso de uso é aberta. A Figura 33 mostra os quatro passos do curso normal do caso de uso “Visualizar Atendimento”.

O primeiro passo a ser executado na tela de cadastro do curso normal é clicar no botão “adicionar” para entrar em modo de inserção. Como isso é feito em outra tela, o usuário deve selecionar o caso de uso no qual ele está trabalhando – “Visualizar Atendimento”. No campo “Seqüência” deve-se informar o número do passo do curso normal, e no campo “Descrição”, a descrição correspondente ao passo que está sendo registrado, conforme exemplificado na Figura 35. O campo “Tem Alternativo?” possui os valores “Sim” e “Não” pré-definidos, para que se possa informar a existência (ou não) de cursos alternativos. Para o caso de uso “Visualizar Atendimento”, somente o passo 2 do curso normal tem um curso alternativo. Para terminar a inserção de um passo, deve-se sempre clicar no botão “salvar”. A Figura 35 mostra

a tela de cadastro do curso normal com os passos do caso de uso “Visualizar Atendimento” devidamente registrados.

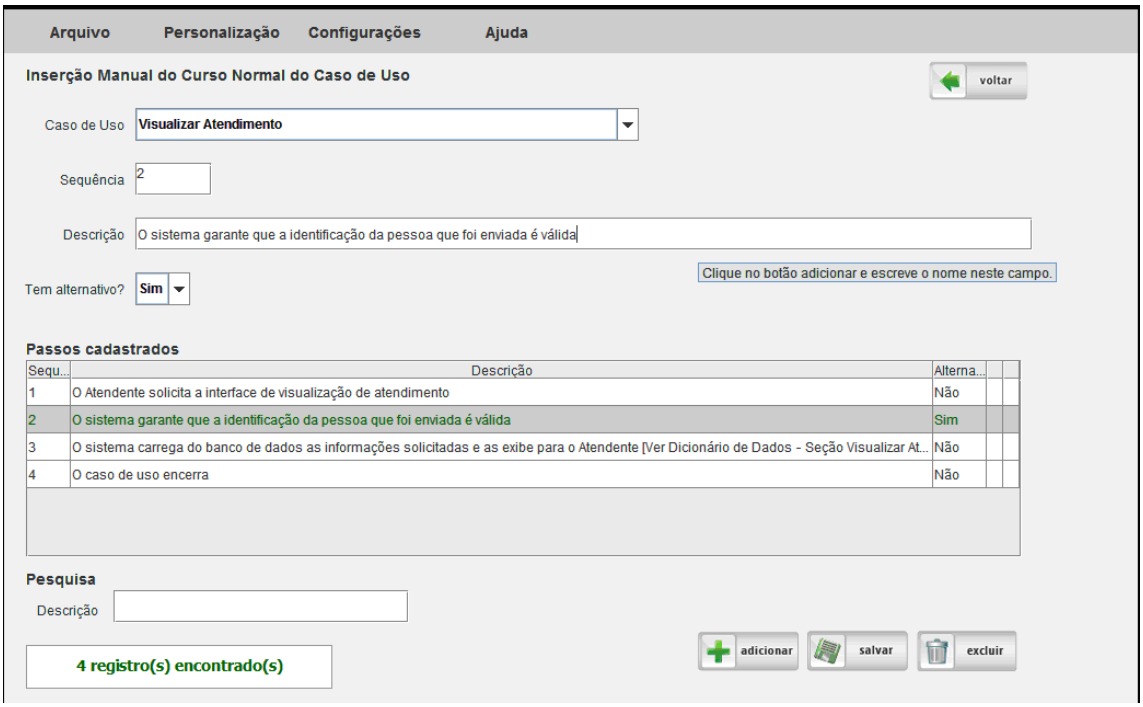


Figura 35: Tela do COCAR para inserção do curso normal do Caso de Uso

Uma vez cadastrado o curso normal do caso de uso, a próxima tarefa é cadastrar seus cursos alternativos, se houver. A tela de cadastro de curso alternativo deve ser acessada por meio do menu “Personalização”, item “Cadastrar Curso Alternativo”.

Para o exemplo em questão, o caso de uso “Visualizar Atendimento” possui apenas um curso alternativo para o passo 2 do curso normal. Para registrá-lo deve-se selecionar o botão “adicionar” para entrar no modo de inserção. Deve-se então selecionar o caso de uso “Visualizar Atendimento” na lista de casos de uso. Ao se fazer isso, os passos do curso normal são carregados no campo “Curso Normal”. Em seguida, deve-se selecionar o passo 2 do “Curso Normal” e então se deve informar um número seqüencial no campo “Nº”, que identifique o curso alternativo, já que o mesmo curso normal pode possuir vários cursos alternativos. Além disso, no campo “Seq. Alternativa”, deve-se informar a seqüência do passo dentro do curso alternativo, e sua descrição no campo “Descrição”. É preciso então clicar no botão “salvar” para registrar o passo do curso alternativo. A Figura 36 exibe o curso alternativo referente ao passo 2 do caso de uso “Visualizar Atendimento” cadastrado no ambiente COCAR.

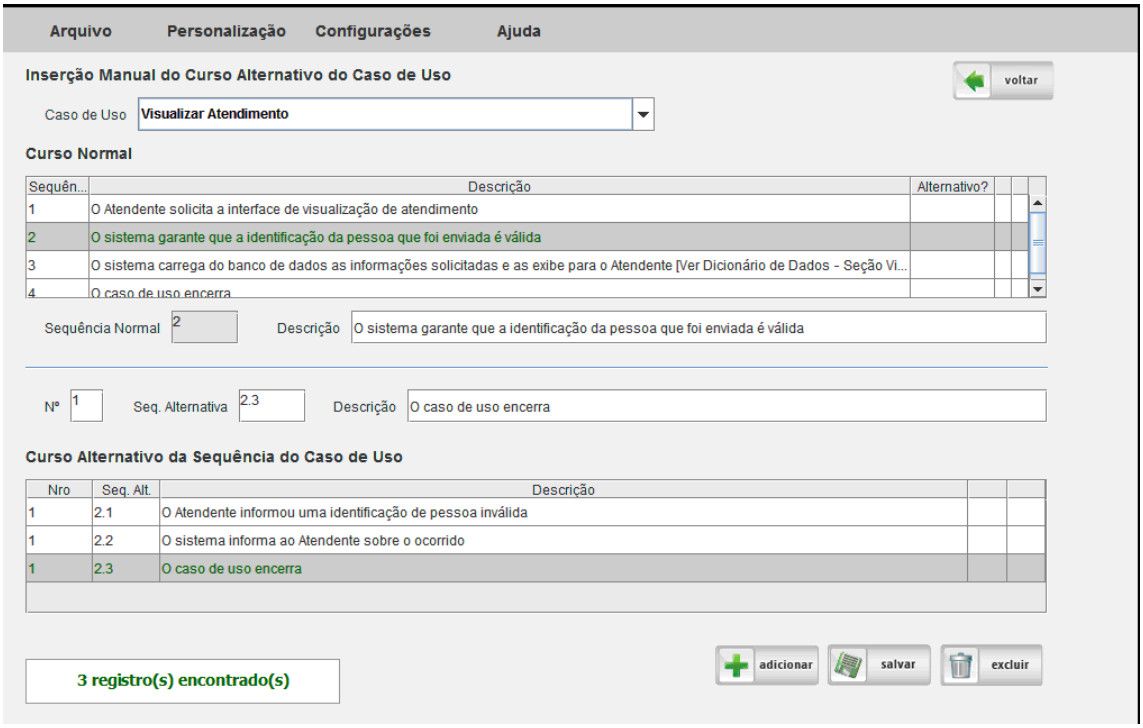


Figura 36: Tela do COCAR para inserção do curso alternativo do Caso de Uso

Vale ressaltar que o uso de uma seqüência lógica para o campo "Seq. Alternativa" auxilia no entendimento do curso alternativo. Essa seqüência é denotada no formato "N.M", no qual N se refere ao passo do curso normal que tem esse curso alternativo e M se refere à seqüência numérica dentro do curso alternativo, como pode ser observado na Figura 36.

Concluídas as etapas de cadastrar o cabeçalho do caso de uso, seu curso normal e seus possíveis cursos alternativos, o caso de uso está devidamente registrado no ambiente COCAR e pode seu utilizado para a geração de cenários de teste.

7.3.2. Importação de Caso de Uso a partir de XMI

Como não havia arquivos XMI fornecidos pela Empresa X referentes aos casos de uso da documentação apresentada no Anexo I, com base no diagrama e especificação, o autor modelou o caso de uso "Gerar Relatório" na ferramenta ArgoUML e exportou seu conteúdo para arquivo XMI. O caso de uso "Gerar Relatório" possui o curso normal com cinco passos e um curso alternativo com três passos referente ao passo 4 do curso normal. A modelagem foi feita segundo as regras de construção detalhadas na Seção 6.3 do Capítulo 6. Parte do arquivo

XMI gerado é exibido na Figura 37 e foi utilizado como exemplo para mostrar o funcionamento do módulo de Importação de Caso de Uso a partir de Arquivo XMI.

```
- <XMI.content>
  - <UML:Model xmi.id="9-8-1-22-79e501f8:12dc880be60:-8000:0000000000000865" name="modeloSemNome"
    isSpecification="false" isRoot="false" isLeaf="false" isAbstract="false">
    - <UML:Namespace.ownedElement>
      - <UML:UseCase xmi.id="9-8-1-22-79e501f8:12dc880be60:-8000:0000000000000866" name="Gerar Relatório"
        isSpecification="false" isRoot="false" isLeaf="false" isAbstract="false">
        - <UML:ModelElement.taggedValue>
          - <UML:TaggedValue xmi.id="9-8-1-22-79e501f8:12dc880be60:-8000:0000000000000872" isSpecification="false">
            - <UML:TaggedValue.dataValue>
                Curso Normal: 1. O Atendente solicita a interface para gerar relatório 2. O Atendente seleciona as informações desejadas
                3. O Atendente seleciona um formato para exportar as informações selecionadas anteriormente 4. O sistema garante que
                as informações estão de acordo com as regras de validação 5. O caso de uso encerra Curso Alternativo 1: 4.1. Informações
                não estão de acordo com as regras de validação 4.2. O sistema informa ao atendente sobre o ocorrido 4.3. O caso de uso encerra
              </UML:TaggedValue.dataValue>
            - <UML:TaggedValue.type>
                <UML:TagDefinition xmi.idref="9-8-1-22-79e501f8:12dc880be60:-8000:000000000000086F"/>
              </UML:TaggedValue.type>
            </UML:TaggedValue>
          </UML:ModelElement.taggedValue>
        </UML:UseCase>
```

Figura 37: Arquivo XMI do Caso de Uso “Gerar Relatório”

Tendo gerado o arquivo XMI, o processo de importação para o ambiente COCAR inicia-se com a seleção da opção “Importar Caso de Uso por XMI” no menu “Personalização”. Na tela carregada, há o botão “...”, destacado na Figura 38, que abre uma caixa de diálogo com o usuário para que o arquivo XMI seja localizado e importado para o ambiente. Seleciona-se então o arquivo XMI desejado (no exemplo é o “Gerar_Relatorio.xmi”) e então se clica no botão “Importar”. A leitura e importação do arquivo XMI são acionadas e, não havendo erro, o nome do caso de uso é carregado no campo “Caso de Uso”, os passos do curso normal são carregados no campo “Curso Normal” e os passos de todos os cursos alternativos (se houver) são carregados no campo “Cursos Alternativos”. Além disso, essas informações são devidamente inseridas nas tabelas do banco de dados. O resultado da importação do arquivo “Gerar_Relatorio.xmi” é exibido na Figura 38.

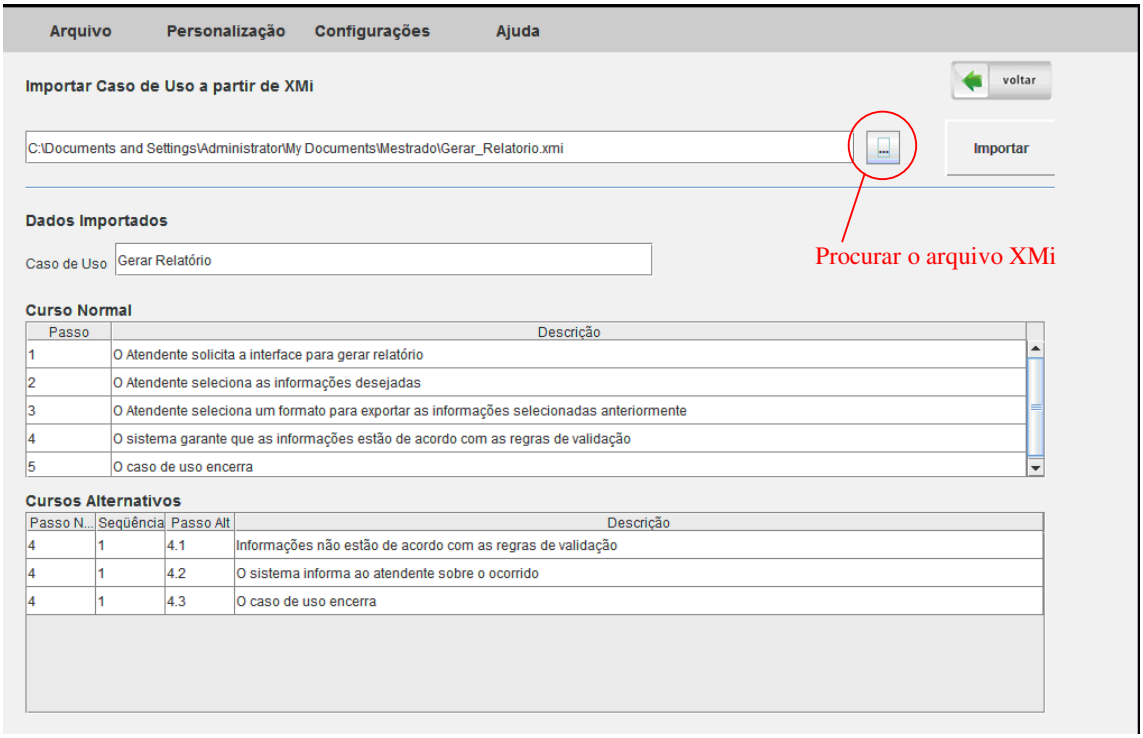


Figura 38: Tela do COCAR usada para importação de casos de uso por meio de arquivo XMI

7.4. Análise de Resultados

Para analisar os resultados obtidos com a implementação das novas funcionalidades no ambiente COCAR, sobretudo o módulo de geração de cenários de teste, foram inseridos no ambiente 12 casos de uso selecionados dentre os 48 existentes na documentação fornecida pela Empresa X, o que compreende 25% do total. Desses 12 casos de uso, 11 foram inseridos de forma manual e 1 por meio de arquivo XMI.

O procedimento realizado foi o seguinte:

1 – Inserção de 11 casos de uso por meio do módulo de cadastro manual de caso de uso.

2 – Inserção de 1 caso de uso modelado na ferramenta ArgoUML e importado para o COCAR por meio de arquivo XMI.

3 – Geração dos cenários de teste unitários para cada caso de uso inserido no ambiente.

4 – Análise dos resultados obtidos ao comparar os cenários de teste unitários gerados no ambiente com os casos de teste existentes no plano de teste da Empresa X.

5 – Geração dos cenários de teste de integração no COCAR.

6 – Análise dos resultados obtidos ao comparar os cenários de teste de integração gerados no ambiente com os cenários deduzidos do plano de teste da Empresa X, com base nas especificações dos casos de teste.

Os seis passos realizados são detalhados na seqüência para uma melhor compreensão.

Passo 1:

Foram selecionados e inseridos 11 casos de uso no ambiente COCAR por meio do módulo de cadastro manual de casos de uso. São eles: "Autenticar Usuário", "Realizar Atendimento", "Visualizar Atendimento", "Visualizar Detalhe do Atendimento", "Cadastrar Pessoa", "Gerenciar Pessoa", "Buscar Pessoa por Identificação", "Buscar Pessoa por Nome", "Visualizar Pessoa", "Gerenciar Entidade" e "Buscar CEP".

Para exemplificar como a inserção das informações referentes aos casos de uso foi realizada, considere o caso de uso "Realizar Atendimento", cuja documentação é exibida na Figura 39. Com base neste documento é que o caso de uso foi registrado no ambiente.

Caso de Uso

Realizar Atendimento

4 Fluxo Principal

1. O Atendente solicita a interface de realizar atendimento.
2. O sistema garante que a identificação da pessoa que foi enviada é válida.
3. O sistema carrega do banco de dados as informações solicitadas e exibe para o Atendente **[Ver Dicionário de Dados – Seção Realizar Atendimento].**
4. O sistema garante que as informações que serão exibidas para o atendente estão de acordo com o tipo de entidade que está sendo trabalhada.
5. O Atendente realiza o atendimento e submete ao sistema **[Ver Dicionário de Dados – Seção Realizar Atendimento] [Ver Especificação Suplementar – Seção Realizar Atendimento].**
6. O sistema valida as informações fornecidas e garante que estão de acordo com as regras de validação **[Ver Dicionário de Dados – Seção Realizar Atendimento].**
7. O sistema persiste as informações fornecidas no banco de dados.
8. O caso de uso encerra.

5 Fluxos Alternativos

Passo	Condição	Fluxo
2	O Atendente informou uma identificação de pessoa inválida.	1. O sistema informa ao Atendente sobre o ocorrido [FTN0004]. 2. O caso de uso encerra.
6	As informações não estão de acordo com as regras de validação.	1. O sistema informa ao Atendente sobre o ocorrido [FTN0003]. 2. O caso de uso encerra.

Figura 39: Exemplo de Documentação dos Cursos do Caso de Uso "Realizar Atendimento"

Desse documento do caso de uso “Realizar Atendimento” foram extraídos:

- Passos do Curso Normal:

  1. O Atendente solicita a interface de realizar atendimento.
  2. O sistema garante que a identificação da pessoa que foi enviada é válida.
  3. O sistema carrega do banco de dados as informações solicitadas e exibe para o Atendente.
  4. O sistema garante que as informações que serão exibidas para o atendente estão de acordo com o tipo de entidade que está sendo trabalhada.
  5. O Atendente realiza o atendimento e submete ao sistema.
  6. O sistema valida as informações fornecidas e garante que estão de acordo com as regras de validação.
  7. O sistema persiste as informações fornecidas no banco de dados.
  8. O caso de uso encerra.

- Passos do Curso Alternativo 1 (para o passo 2 do curso normal):

  2.1. O Atendente informou uma identificação de pessoa inválida.
  2.2. O sistema informa ao Atendente sobre o ocorrido.
  2.3. O caso de uso encerra.

- Passos do Curso Alternativo 2 (para o passo 6 do curso normal):

  6.1. As informações não estão de acordo com as regras de validação.
  6.2. O sistema informa ao Atendente sobre o ocorrido.
  6.3. O caso de uso encerra.

Passo 2:

O caso de uso “Gerar Relatório” foi modelado na ferramenta ArgoUML e os passos de seus cursos normal e alternativos descritos conforme regras detalhadas na Seção 6.4. O caso de uso foi exportado no formato XMI e então importado para o ambiente utilizando o módulo de importação de casos de uso a partir de arquivo XMI.

Passo 3:

Inseridos os 12 casos de uso no ambiente COCAR, pode-se obter grafos, tendo por base os passos do curso normal e adicionando-se os passos dos cursos alternativos do caso de uso. O grafo que representa os caminhos de execução do caso de uso "Realizar Atendimento" descrito no Passo 1 e conforme explicado no Capítulo 6, está apresentado na Figura 40.

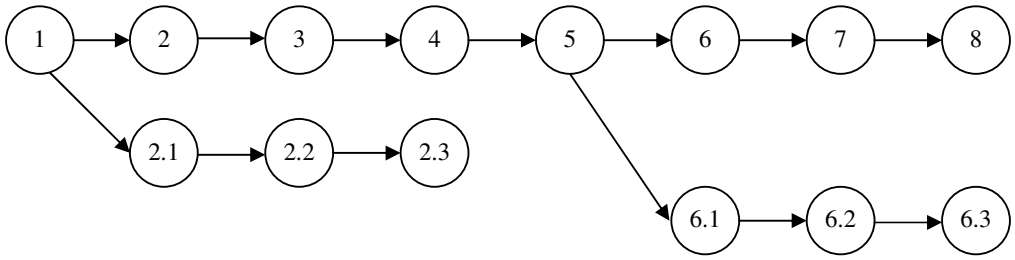


Figura 40: Grafo dos cursos normal e alternativos do Caso de Uso "Realizar Atendimento"

Percorrendo o grafo com início no passo 1 ("O Atendente solicita a interface de realizar atendimento"), temos os três possíveis cenários de teste unitários para o caso de uso "Realizar Atendimento" que são:

- Cenário de Teste Unitário 1: nós 1 – 2 – 3 – 4 – 5 – 6 – 7 – 8 (curso normal)
- Cenário de Teste Unitário 2: nós 1 – 2.1 – 2.2 – 2.3 (curso alternativo)
- Cenário de Teste Unitário 3: nós 1 – 2 – 3 – 4 – 5 – 6.1 – 6.2 – 6.3 (curso alternativo)

Assim sendo, ao substituirmos os nós percorridos por seus respectivos passos do curso a que se referem, os três cenários de teste unitários resultantes são:

- Cenário de Teste Unitário 1: os 8 passos do curso normal
- Cenário de Teste Unitário 2:

  1. O Atendente solicita a interface de realizar atendimento.
  2. O Atendente informou uma identificação de pessoa inválida.
  3. O sistema informa ao Atendente sobre o ocorrido.
  4. O caso de uso encerra.

- Cenário de Teste Unitário 3:

  1. O Atendente solicita a interface de realizar atendimento.
  2. O sistema garante que a identificação da pessoa que foi enviada é válida.
  3. O sistema carrega do banco de dados as informações solicitadas e exibe para o Atendente.
  4. O sistema garante que as informações que serão exibidas para o atendente estão de acordo com o tipo de entidade que está sendo trabalhada.
  5. O Atendente realiza o atendimento e submete ao sistema.
  6. As informações não estão de acordo com as regras de validação.
  7. O sistema informa ao Atendente sobre o ocorrido.
  8. O caso de uso encerra.

Passo 4:

Gerados os cenários de teste unitários, o próximo passo foi compará-los com os casos de teste que haviam sido elaborados pela Empresa X. Um desses casos de teste está apresentado na Figura 41, que mostra uma parte do arquivo de plano de teste.

39.1 – Realizar Atendimento (com sucesso) – atendimento geral

Pré-condição: O usuário deve estar autenticado como coordenador da entidade (3)
Pré-condição: O usuário deve estar autenticado como técnico da entidade (4)

URL:http://www.wiface.com.br/rede-das-redes/attendance/attendance/person/x

Step Name	Description	Expected Result
Step 1	* Usuário solicita ao sistema a interface para realizar atendimento.	* O sistema exibe a interface para realizar atendimento.
Step 2	* Usuário informa: - Tipo de atendimento oferecido por sua entidade válido (1 – existente na combo) Atenção: tester verifica se todos os atendimentos listados são realmente da entidade. * Usuário aciona o botão "Próxima".	- O sistema deve exibir uma interface para informar a data e se a informação é confidencial.
Step 3	* Usuário informa: - Uma data prevista para término válida (data válida, até 10 caracteres) - Informação confidencial válida. * Usuário aciona o botão "Salvar".	* Sistema deve exibir o formulário de atendimento da sua entidade.

Figura 41: Exemplo de Caso de Teste extraído do Plano de Teste da Empresa X para o Caso de Uso "Realizar Atendimento"

A Tabela 3 apresenta os cenários de teste que foram gerados no COCAR e os casos de teste que estão registrados no plano de teste da Empresa X para o caso de uso "Realizar Atendimento", relacionando-os sempre que seus objetivos fossem os mesmos.

Tabela 3: Comparação dos Cenários de Teste Unitários e dos Casos de Teste para o Caso de Uso "Realizar Atendimento"

Caso de Uso: Realizar Atendimento				
Cenário COCAR	**Descrição Geral**	**Caso de Teste Sistema X**	**Descrição Geral**	**Divergência**
1	Curso normal – atendimento realizado	39.1 / 39.2 / 39.3	Realizar atendimento com sucesso	-
2	Identificação de pessoa inválida	39.6	Identificação de pessoa inválida	-
3	Informações do atendimento inválidas	39.5	Tipo de atendimento inválido	-
-	-	39.4	Usuário sem permissão de acesso	Não está previsto nos cursos do caso de uso
		39.9 a 39.20	Testes de segurança, confiabilidade, sigilo	Não estão previstos nos cursos do caso de uso

Assim, analisando os dados da Tebela 3 observa-se que os 3 cenários de teste unitários gerados pelo ambiente COCAR possuem casos de teste relacionados. Entretanto, há casos de teste do plano de teste da Empresa X, como o 39.4 ("Usuário sem permissão de acesso"), para o qual não existe cenário de teste gerado pelo COCAR pois não há especificação nenhuma nos cursos normal e alternativos da documentação da empresa. Os casos de teste 39.9 a 39.20 também não possuem cenários correspondentes gerados pelo COCAR, pois eles se referem a requisitos não-funcionais, tais como confiabilidade, sigilo de informações, os quais também não foram especificados na documentação da empresa.

Para todos os outros 11 casos de uso o mesmo tipo de análise comparativa foi realizada, relacionando-se os cenários de teste unitários gerados no ambiente COCAR com os casos de teste do documento de testes fornecido pela Empresa X. Os resultados são exibidos na Tabela 4.

Tabela 4: Comparação dos cenários de teste unitários e dos casos de teste utilizados no estudo de caso

Caso de Uso: Autenticar Usuário				
Cenário COCAR	**Descrição Geral**	**Caso de Teste Sistema X**	**Descrição Geral**	**Divergência**
1	Curso normal – Operador autenticado no sistema	38.5	Autenticar usuário Operador com sucesso	-
2	Informações de usuário inválidas	38.6	Nome de usuário inválido	-
3	Informações válidas, mas usuário não existe	38.8	Usuário não existe no sistema	-
4	Usuário Inativo	38.7		-
5	Técnico da Entidade autenticado no sistema	38.4	Autenticar usuário Técnico da Entidade	-
6	Coordenador da Entidade autenticado no sistema	38.3	Autenticar usuário Coordenador da Entidade	-
7	Gerente da Rede autenticado no sistema	38.2	Autenticar usuário Gerente da Rede	-
8	Coordenador de Rede autenticado no sistema	38.1	Autenticar usuário Administrador	-
Caso de Uso: Visualizar Atendimento				
Cenário COCAR	**Descrição Geral**	**Caso de Teste Sistema X**	**Descrição Geral**	**Divergência**
1	Curso normal – atendimento realizado	-	Realizar atendimento com sucesso	Não há casos de teste especificados para o caso de uso
2	Identificação de pessoa inválida	-	Identificação de pessoa inválida	Não há casos de teste especificados para o caso de uso
Caso de Uso: Visualizar Detalhe do Atendimento				
Cenário COCAR	**Descrição Geral**	**Caso de Teste Sistema X**	**Descrição Geral**	**Divergência**
1	Curso normal – atendimento realizado	-	Realizar atendimento com sucesso	Não há casos de teste especificados para o caso de uso
2	Identificação de pessoa inválida	-	Identificação de pessoa inválida	Não há casos de teste especificados para o caso de uso
Caso de Uso: Cadastrar Pessoa				
Cenário COCAR	**Descrição Geral**	**Caso de Teste Sistema X**	**Descrição Geral**	**Divergência**
1	Curso normal – Cadastrar pessoa	26.1	Cadastrar pessoa com sucesso	-
2	Informações inválidas	26.3 a 26.13	Cadastrar pessoa com campos inválidos	-
3	Editar pessoa	26.1	Cadastrar pessoa com sucesso	-
-	-	26.2	Aciona o botão Voltar no browser	Não está previsto nos cursos do caso de uso
-	-	26.15	Executa a interface de cadastrar pessoa e clica no botão Salvar direto	Não está previsto nos cursos do caso de uso
Caso de Uso: Gerenciar Pessoa				
Cenário COCAR	**Descrição Geral**	**Caso de Teste Sistema X**	**Descrição Geral**	**Divergência**

1	Curso normal – Carregar informações da pessoa	27.1	Realizar atendimento com sucesso	-
2	Editar as informações da pessoa	27.1	Identificação de pessoa inválida	-
-	-	27.2	Aciona o botão Voltar no browser	Não está previsto nos cursos do caso de uso
-	-	27.3 a 27.37	Informações Inválidas	Não está previsto nos cursos do caso de uso
Caso de Uso: Buscar Pessoa por Identificação				
Cenário COCAR	**Descrição Geral**	**Caso de Teste Sistema X**	**Descrição Geral**	**Divergência**
1	Curso normal – Buscar pessoa por documento	17.1	Buscar pessoa por documento com sucesso	-
2	Identificação de pessoa inválida	17.2 a 17.4	Busca por informações de documento inválidas	-
Caso de Uso: Buscar Pessoa por Nome				
Cenário COCAR	**Descrição Geral**	**Caso de Teste Sistema X**	**Descrição Geral**	**Divergência**
1	Curso normal – Buscar pessoa por nome ou apelido	18.1 a 18.3	Buscar pessoa por nome ou apelido com sucesso	-
2	Identificação de pessoa inválida	-	-	Não há caso de teste previsto
-	-	18.4 a 18.10	Tipo de atendimento inválido	Busca por outros campos não previstos no caso de uso
Caso de Uso: Visualizar Pessoa				
Cenário COCAR	**Descrição Geral**	**Caso de Teste Sistema X**	**Descrição Geral**	**Divergência**
1	Curso normal – Exibe as informações da pessoa	25.1	Carrega as informações da pessoa e exibe	-
2	Identificação de pessoa inválida	25.2	Identificação de pessoa inválida	-
Caso de Uso: Gerenciar Entidade				
Cenário COCAR	**Descrição Geral**	**Caso de Teste Sistema X**	**Descrição Geral**	**Divergência**
1	Curso normal – Exibe as informações da entidade	9.1	Exibe as informações da entidade e edita	-
2	Atendente edita as informações da entidade	9.1 / 9.21	Exibe as informações da entidade e edita, buscando endereço	-
3	Atendente cadastra a entidade	-	-	Não há caso de teste previsto
-	-	9.2 a 9.16	Informações de campos inválidas	Não está previsto nos cursos do caso de uso
Caso de Uso: Gerar Relatório				
Cenário COCAR	**Descrição Geral**	**Caso de Teste Sistema X**	**Descrição Geral**	**Divergência**
1	Curso normal – Exibe as informações em HTML	43.1	Realizar atendimento com sucesso	-
2	Informações de	-	-	Não há caso de

	solicitação inválidas			teste previsto
Caso de Uso: Buscar CEP				
Cenário COCAR	**Descrição Geral**	**Caso de Teste Sistema X**	**Descrição Geral**	**Divergência**
1	Curso normal – Exibe as informações sobre CEP informado	-	-	Não há caso de teste previsto
2	Informações de solicitação inválidas	-	-	Não há caso de teste previsto

Com base nos resultados coletados para cada caso de uso e exibidos na Tabela 4, chegou-se às seguintes conclusões em relação aos cenários de teste unitários gerados com o apoio do ambiente COCAR:

- Houve cenários de teste gerados pelo COCAR que contemplaram, em igual número, todos os casos de teste aplicados pela empresa para testar um caso de uso. Isso aconteceu com os casos de uso "Autenticar Usuário" e "Busca Pessoa por Identificação".
- Houve cenários de teste gerados pelo COCAR que não possuíram casos de teste relacionados, isto é, com os mesmos objetivos. Isso ocorreu com os casos de uso "Buscar CEP", "Visualizar Atendimento" e "Visualizar Detalhe do Atendimento". Como esses cenários de teste não foram exercitados por nenhum caso de teste, nada indica que não haja erros de implementação referentes a esses casos de uso.
- Houve alguns cenários de teste para um determinado caso de uso não possuíram casos de teste instanciados, exercitando apenas parcialmente os possíveis cenários de execução para o caso de uso. Isso aconteceu com os casos de uso "Buscar Pessoa por Nome" e "Gerenciar Entidade", indicando que pode haver falhas no sistema em alguma ação não testada.
- Houve alguns casos de teste instanciados no plano de teste da Empresa X para os quais nenhum cenários de teste foi gerado pelo ambiente COCAR. Isso pode ser observado para os casos de uso "Realizar Atendimento", "Cadastrar Pessoa", "Gerenciar Pessoa", "Buscar Pessoa por Nome" e "Gerenciar Entidade". A razão disso é que os passos desses casos de teste não foram previstos em nenhum dos passos dos cursos normais e alternativos dos casos de uso referidos, o que pode indicar alguma falha em suas especificações.
- Houve um cenário de teste gerado pelo COCAR que apresentou divergência de especificação em relação aos casos de teste relacionado a ele. Isso aconteceu com o

caso de uso "Autenticar Usuário", uma vez que se refere ao ator "Coordenador de Rede", o qual não existe na documentação da empresa descrita no Anexo I. O ator "Administrador de Rede", utilizado no caso de teste relacionado, é que deveria ser especificado nesse caso de uso.

Essa análise, além da questão de teste, evidenciou outros pontos também importantes para a qualidade de software como, por exemplo, requisitos que não tinham sido tratados na especificação dos casos de uso e nome de ator incorreto utilizado, indicando que o ambiente COCAR pode auxiliar na inspeção de outros artefatos que não o plano de teste apenas.

Se estivesse usando somente o ambiente COCAR, com base na documentação dos casos de uso da empresa, os cenários de teste unitários gerados no ambiente ajudariam a criar o plano de teste da empresa, o qual certamente não seria completo, uma vez que os cenários gerados pelo ambiente não explorariam todas as funcionalidades do sistema, pois há requisitos não especificados nos casos de uso pela Empresa X.

Se estivesse usando o COCAR para avaliar a atividade de teste, o plano de teste poderia ser corrigido, tornando-se mais completo, uma vez que os cenários de teste unitários gerados no COCAR e não exercitados no plano de teste da empresa, auxiliariam na elaboração de casos de teste adicionais para exercitá-los.

Passo 5:

Seguindo o procedimento adotado para avaliar o trabalho, o próximo passo foi gerar os cenários de teste de integração, com o objetivo de verificar se as relações de dependência existentes entre os casos de uso foram testadas. Essas relações foram construídas com base nas informações existentes nas especificações dos casos de uso, tais como as pré-condições, as descrições dos casos de uso e os passos registrados dos cursos normal e alternativos desses casos de uso, e geraram o grafo da Figura 42.

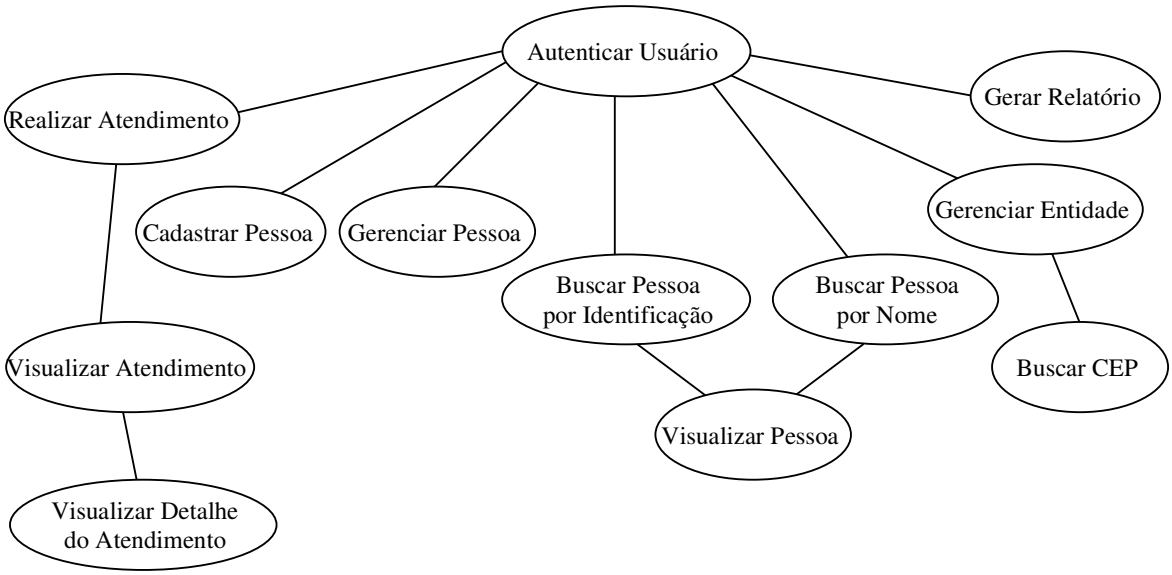


Figura 42: Grafo de dependência dos casos de uso utilizados no estudo de caso

Com base nesse grafo, foram gerados automaticamente 7 cenários de teste de integração no ambiente COCAR, conforme Figura 43.

Figura 43: Cenários de teste de integração dos casos de uso utilizados no estudo de caso

Passo 6:

Para caracterizar os cenários testados pela Empresa X, foi feita uma análise do Plano de Teste referente aos casos de teste executados para os 12 casos de uso selecionados como amostragem. Por meio das pré-condições dos casos de uso, da relação de inclusão (*include*) e extensão (*extend*), e também de referências nos passos dos cursos normal e alternativos, identificaram-se, indiretamente, os cenários de integração exercitados após a execução dos casos de teste para cada caso de uso. Vale ressaltar que o plano de teste da Empresa X possui

casos de teste criados para cada caso de uso especificado, com suas pré-condições, passos de execução e resultados esperados para cada um deles.

Os resultados da análise comparativa entre os cenários de teste de integração gerados no COCAR com os cenários de teste de integração deduzidos do plano de teste da Empresa X são mostrados na Tabela 5. A numeração dos cenários gerados é a mesma da Figura 43.

Tabela 5: Relação entre os cenários de teste de integração gerados pelo COCAR e utilizados no plano de teste da Empresa X

Cenário COCAR	**Avaliação em relação ao Plano de Teste da Empresa X**
1	A relação entre os casos de uso "Autenticar Usuário" e "Realizar Atendimento" é exercitada no caso de teste 9.4 e também nas pré-condições do caso de uso "Realizar Atendimento". Porém não há casos de teste para "Visualizar Atendimento" e "Visualizar Detalhe do Atendimento"
2	A relação é encontrada nas pré-condições do caso de teste do caso de uso "Cadastrar Pessoa", quando enfatiza a necessidade de autenticação do usuário, satisfazendo o cenário de teste por completo
3	A relação é encontrada nas pré-condições do caso de teste do caso de uso "Gerenciar Pessoa", quando enfatiza a necessidade de autenticação do usuário, satisfazendo o cenário de teste por completo
4	A relação entre os casos de uso "Autenticar Usuário" e "Buscar Pessoa por Identificação" é encontrada nas pré-condições do caso de uso "Buscar Pessoa por Identificação". Os casos de teste 25.1 e 25.2 parecem exercitar a relação entre "Buscar Pessoa por Identidade" e "Visualizar Pessoa", satisfazendo o cenário por completo
5	A relação entre os casos de uso "Autenticar Usuário" e "Buscar Pessoa por Nome" é encontrada nas pré-condições do caso de uso "Buscar Pessoa por Nome". Porém não há casos de teste que exercitam a relação entre "Buscar Nome por Nome" e "Visualizar Pessoa"
6	A relação entre os casos de uso "Autenticar Usuário" e "Gerenciar Entidade" é encontrada nas pré-condições do caso de uso "Gerenciar Entidade". Caso de Teste 9.21 exercita a relação entre "Gerenciar Entidade" e "Buscar CEP", ao testar a busca por endereço com base em include do caso de uso "Buscar CEP", satisfazendo assim o cenário por completo
7	A relação é encontrada nas pré-condições do caso de teste do caso de uso "Gerar Relatório", quando enfatiza a necessidade de autenticação do usuário, satisfazendo o cenário de teste por completo

Algumas conclusões podem ser extraídas da Tabela 5 e também da análise de toda a documentação de testes da Empresa X:

- 5 dentre os 7 cenários de teste gerados pelo COCAR para os 12 casos de uso selecionados foram exercitados, de alguma maneira, pelos casos de teste da

empresa. Esta conclusão pode ser observada para os cenários de teste número 2, 3, 4, 6 e 7.

- O cenário de teste de integração 1 não foi completamente testado, pois não há casos de teste previstos para os casos de uso "Visualizar Atendimento" e "Visualizar Detalhe do Atendimento". Isso mostra uma deficiência no plano de teste, uma vez que um ponto importante a ser testado, como a passagem de parâmetros entre os casos de uso "Realizar Atendimento" e "Visualizar Atendimento", e também entre os casos de uso "Visualizar Atendimento" e "Visualizar Detalhe do Atendimento", não foi exercitado, e esse ponto costuma ser causa de falhas em muitos casos.
- O cenário de teste de integração 5 não foi completamente testado porque não há nenhum caso de teste previsto ou pré-condição para o caso de uso "Visualizar Pessoa" que busque uma pessoa por nome, apenas por identificação. Dessa forma há um ponto falho no plano de teste no que se refere a esse cenário, pois somente a identificação da pessoa foi validada ao relacionar os casos de uso "Buscar Pessoa por Nome" e "Visualizar Pessoa". Seria necessária a inclusão de caso de teste adicional para o caso de uso "Adicionar Pessoa" exercitando a busca pelo nome da pessoa.

Assim, é possível perceber que os cenários de teste de integração gerados automaticamente no ambiente COCAR tendem a auxiliar a equipe de planejadores de teste, uma vez que mostra as possíveis combinações entre casos de uso dependentes. Isso contribui para a criação de um plano de teste mais completo, que contenha casos de teste para exercitar as relações de precedência existentes entre os casos de uso, evitando falhas como as identificadas na Tabela 5.

Para sistemas que possuem uma grande quantidade de casos de uso e, principalmente, se houver muitos relacionamentos entre eles, o fato de se obter, com antecedência, os possíveis cenários de execução do sistema, auxilia muito na elaboração do plano de teste e, consecutivamente, na instanciação dos casos de teste.

7.5. Considerações Finais

Apresentou-se neste capítulo um estudo de caso para analisar os resultados dos módulos implementados no ambiente COCAR, sobretudo o módulo de geração automática de cenários de teste unitários e de integração. Os resultados foram obtidos por meio de comparação entre os cenários de teste gerados no ambiente e os casos de teste utilizados em uma situação real, de uma empresa aqui chamada Empresa X.

Os resultados obtidos mostraram que a geração automática de cenários de teste pode contribuir para pessoas e empresas que façam especificação por meio de casos de uso e desejem gerar casos de teste mais completos, que exercitem a maior parte de cenários de teste possível.

CAPÍTULO 8

Conclusões

Este trabalho teve como objetivo gerar cenários de teste com base em casos de uso, para auxiliar, principalmente, na elaboração de um plano de teste mais eficaz, uma vez que o mesmo já pode ter início nas primeiras fases do ciclo de desenvolvimento de um software. Os trabalhos de Ryser & Glinz (1995; 1999a; 1999b; 1999c; 2000), de Briand & Labiche (2002), de Carniello (2003a; 2003b), de Hartmann et al. (2005), de Vieira et al. (2006) e de Nebut et al. (2006) são importantes nesse contexto e serviram de base para este trabalho.

A partir de um diagrama de dependência entre os casos de uso, como o proposto por Ryser & Glinz (1995), estabelece-se a relação de precedência entre os casos de uso e, então, um grafo de dependência pode ser criado a partir desse diagrama. Assim, ao percorrer esse grafo de dependência, é possível gerar cenários de integração, exercitando-se a relação de dependência entre os casos de uso. De maneira análoga, é possível criar grafos de execução para os passos dos cursos normal e alternativos de cada caso de uso, sendo que, ao percorrê-la, podem-se gerar cenários de teste unitários, para exercitar os possíveis caminhos de execução de um determinado caso de uso.

Como a atividade de teste, inclusive a elaboração do plano de teste, muitas vezes é trabalhosa, pensou-se na automatização de todo esse processo. Assim, o ambiente COCAR foi utilizado, uma vez que já possuía funcionalidades apoiadas em casos de uso. Um novo módulo foi implementado nesse ambiente, cuja funcionalidade é a de geração de cenários de teste com base em casos de uso, além de módulos adicionais, como o cadastro manual de casos de uso e a importação de casos de uso a partir de arquivos no formato XMI, que permitem a usuários que utilizem outras ferramentas de modelagem de casos de uso e queiram apenas gerar cenários de teste por meio do ambiente COCAR, tenham essa possibilidade.

Todas essas funcionalidades foram apresentadas neste trabalho e exercitadas por meio de um estudo de caso com dados reais de uma empresa, com apoio do ambiente COCAR. Pelas informações coletadas, percebeu-se a sua contribuição em relação a auxiliar na geração de casos de teste mais eficazes, percorrendo os possíveis cenários de um sistema, bem como

na inspeção de artefatos existentes, como o Documento de Casos de Uso e o Plano de Teste, apontando divergências existentes entre eles.

8.1. Contribuições e limitações deste trabalho

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

- Estudo de propostas encontradas literatura sobre teste baseado em casos de uso e definição de estratégias de geração de cenários de teste unitários e de integração com base em modelos de casos de uso.

- Implementação do módulo de geração de cenários de teste com base em casos de uso. Esse módulo possibilita a geração de cenários de teste de unidade a fim de exercitar os casos de uso individualmente, e de cenários de teste de integração, para exercitar as relações de dependência entre os casos de uso. O objetivo é fornecer aos usuários os possíveis cenários de teste que envolvam os casos de uso de um sistema cadastrados no ambiente. Com isso, os usuários podem elaborar casos de teste quem contemplem e exercitem os possíveis cenários de um sistema, possibilitando a geração de um Plano de Teste mais bem elaborado.

- Implementação do módulo de cadastro manual dos Casos de Uso no ambiente COCAR. Esse módulo, composto pelas telas de cadastro do cabeçalho do caso de uso, tela de cadastro do curso normal e tela de cadastro dos cursos alternativos, permite a inserção de casos de uso no ambiente sem a necessidade de utilização da TUCCA. O objetivo é tornar o ambiente mais flexível e permitir que usuários que desejem apenas gerar cenários de teste para os casos de uso já documentados que possuem, consigam fazer uso do módulo de geração de cenários de teste.

- Implementação do módulo de importação de casos de uso a partir de arquivo no formato XMI. Esse módulo, assim como o de cadastro manual de caso de uso, tem como objetivo tornar o ambiente mais flexível, integrando-o com ferramentas CASE que possuam as funcionalidades de modelagem de casos de uso e de exportação dos modelos gerados para arquivo XMI. A ferramenta ArgoUML foi selecionada como exemplo e arquivos XMI gerados a partir dela foram importados

para o ambiente COCAR, comprovando seu funcionamento e fornecendo diretrizes de como futuras integrações com outras ferramentas podem ser realizadas.

- Identificação de defeitos em artefatos de software, em decorrência do uso dos cenários de teste gerados pelo COCAR para comparação com dados da atividade de teste de uma empresa real, mostrando que os cenários gerados pelo ambiente podem ser usados também com o propósito de avaliar, por exemplo, a completitude dos requisitos ou mesmo a ausência de cenários de teste relevantes.

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

- A integração entre o ambiente COCAR e outras ferramentas CASE, que modelam casos de uso e permitem a exportação das especificações dos casos de uso para arquivos no formato XMI, restringiu-se apenas à ferramenta ArgoUML neste trabalho.

- Para os casos de uso inseridos no ambiente por meio do módulo de cadastro manual de casos de uso ou via importação de arquivo XMI, os outros módulos do ambiente não podem ser utilizados, uma vez que a relação entre os requisitos e os casos de uso não fica estabelecida no ambiente. Tal relação é feita ao gerar o modelo de casos de uso com a aplicação da TUCCA.

- Para que o módulo de importação de Casos de Uso a partir de arquivo XMI funcione corretamente no ambiente COCAR, foram criadas regras de preenchimento dos passos dos cursos dos casos de uso que o usuário deve seguir. Isso faz com que os usuários tenham que se adequar a essas regras para conseguirem utilizar a funcionalidade, não podendo especificar os casos de uso como estavam acostumados habitualmente, pois na grande maioria das vezes, as regras estabelecidas não são usadas intuitivamente, ou seja, os desenvolvedores usam um texto corrente, sem formatação. Com isso, pode haver a necessidade de um esforço grande para inserir os casos de uso manualmente no ambiente COCAR.

- Embora o estudo de caso tenha sido realizado com dados reais de uma empresa, toda a manipulação do ambiente COCAR, seja a inserção dos casos de uso ou a geração dos cenários de teste, foi feita pelo próprio autor deste trabalho. O ideal

seria que o ambiente fosse instalado na empresa e manipulado pelos próprios funcionários.

- Em relação aos cenários de teste de integração, o módulo implementado gera apenas os cenários de teste válidos. Os cenários inválidos contribuiriam para a geração de casos de teste cujo resultado esperado é a execução sem sucesso a fim de garantir a consistência da execução. Porém, esses cenários de integração inválidos não são gerados na atual versão do ambiente.

8.2. Lições Aprendidas

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

Primeiramente é preciso comentar sobre o tempo de conversão do ambiente da versão Web para a versão Desktop. Os motivos e alguns detalhes da conversão foram especificados nas Considerações Iniciais do Capítulo 6. O tempo para converter os principais módulos do ambiente foi maior do que o previsto inicialmente, o que resultou em atrasos nos prazos de entrega deste trabalho e impossibilitou que o ambiente fosse utilizado e testado mais efetivamente, principalmente por empresas de desenvolvimento de software.

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.

8.3. Trabalhos Futuros

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

- Integrar o ambiente COCAR com outras ferramentas CASE que não sejam a ArgoUML por meio de arquivo no formato XMI. As diretrizes utilizadas no ambiente em relação à ArgoUML servem como base para essas futuras integrações.

- Implementar uma interface que relacione os requisitos com os casos de uso inseridos no ambiente COCAR por meio de cadastro manual ou por importação de arquivo XMI. Dessa forma, o usuário também poderia inserir os requisitos no ambiente de forma manual, e poderia relacioná-los aos respectivos casos de uso, permitindo assim que os demais módulos do ambiente relacionados à gerência dos requisitos e ao planejamento de projeto também possam ser utilizados.

- Realizar novos estudos de caso com outras empresas, avaliando a utilização do ambiente COCAR na indústria, com sistemas complexos e de diferentes portes. Ocorreu um contato tardio com o gerente da empresa Linkway, porém não houve tempo hábil de conseguir a documentação necessária dos casos de uso e dos casos de teste, a fim de realizar comparações, ou até mesmo de gerar os cenários de teste para sistemas que estejam nas fases iniciais de seu ciclo de vida, com o intuito de auxiliar na geração dos casos de teste.

- Implementar melhorias no módulo de geração de cenários de teste de integração, possibilitando que cenários de teste de integração inválidos sejam gerados também, assim como ocorre com a geração dos cenários de teste unitários.

- Converter os demais módulos do ambiente, como o de Rastreabilidade de Requisitos e o de Geração de Pontos de Caso de Uso da versão Web para a versão Desktop. Esses módulos não foram convertidos durante este trabalho por falta de tempo hábil, uma vez que eles não eram vitais para a implementação e análise das novas funcionalidades adicionadas ao ambiente.

- Gerar dados de teste a partir dos cenários de teste gerados no ambiente COCAR.

- Otimizar o trabalho, permitindo ao usuário selecionar possíveis combinações de cenários de teste, e não apenas garantir a execução de cada cenário de teste individualmente.

REFERÊNCIAS

[Andriole, 1986] ANDRIOLE, S. J. **Software Validation, Verification, Testing and Documentation**. New Jersey: Petrocelli Books, 1986, p. 389.

[Apache, 2010] **Documentação do Software Apache**, 2010. Disponível em: <http://www.apache.org/> Acesso em: 15 jun. 2010.

[ArgoUML, 2010] **Documentação da Ferramenta ArgoUML**, 2010. Disponível em: <http://argouml.tigris.org/> Acesso em: 12 dez. 2010.

[Astah, 2010] **Documentação da Ferramenta Astah**, 2010. Disponível em: <http://astah.net/editions/community/> Acesso em: 12 dez. 2010.

[Basili et al., 1996a] BASILI, V. R.; CALDIERA, G.; LANUBILE, F.; SHULL, F. **Studies on Reading Techniques**. In: ANNUAL SOFTWARE ENGINEERING WORKSHOP, 21., 1996a, Greenbelt, Maryland. NASA/Goddard Software Engineering Laboratory Series, December, 1996. p. 59-65.

[Basili et al., 1996b] BASILI, V. R.; GREEN, S.; LAITENBERGER, O.; LANUBILE, F.; SHULL, F.; SORUMGARD, S.; ZELKOWITZ, M. **The Empirical Investigation of Perspective-Based Reading. Empirical Software Engineering: An International Journal**, v.1, n.2, p. 133-164, 1996b.

[Basili et al., 1998] BASILI, V.; GREEN, S.; LAITENBERGER, O.; LANUBILE, F.; SHULL, F.; SORUMGARD, S.; ZELKOWITZ, M. **Lab Package for the Empirical Investigation of Perspective-Based Reading**, 1998. University of Maryland. Disponível em: <http://www.cs.umd.edu/projects/SoftEng/ESEG/manual/pbr_package/node19.html> Acesso em: 17 mar. 2009.

[Belgamo, 2004a] BELGAMO, A. **GUCCRA: Técnicas de Leitura para Construção de Modelos de Casos de Uso e Análise de Documentos de Requisitos**, 2004.

Dissertação (Mestrado em Pós Graduação em Ciência da Computação) – Universidade Federal de São Carlos, São Carlos.

[Belgamo et al., 2004b] BELGAMO, A.; FABBRI, S. C. P. F. **Constructing Use Case Model by Using a Systematic Approach: Description of a Study**. In: WER 2004 - Workshop de Engenharia de Requisitos, 2004, Tandil, Argentina.

[Belgamo et al., 2005a] BELGAMO, A.; FABBRI, S. C. P. F.; MALDONADO, J. C..**Avaliando a Qualidade da Técnica GUCCRA com Técnica de Inspeção**. In: WER 2005 - Workshop de Engenharia de Requisitos, 2005, Porto, Portugal.

[Belgamo et al., 2005b] BELGAMO, A.; FABBRI, S. C. P. F.; MALDONADO, J. C..**TUCCA: Improving the Effectiveness of Use Case Construction and Requirement** Analysis. In: ISESE 2005 International Symposium on Empirical Software Engineering, 2005, Noosa Heads, Austrália.

[Biolchini et al., 2005] BIOLCHINI J.; MIAN, P. G.; NATALI, A. C. C.; Travassos, G. H. **Systematic Review in Software Engineering**. Technical Report RT-ES 679 / 05, Rio de Janeiro, Brazil, 2005.

[Briand & Labiche, 2002] BRIAND, L.; LABICHE, Y. **A UML-Based Approach to System Testing**. Technical Report. Carleton University, 2002. Disponível em: <http://squall.sce.carleton.ca/pubs/journal/2002_Briand_%20Labiche.pdf>. Acesso em: 22 abr. 2009.

[Budd, 1981] BUDD, T. A. **Mutation Analysis: Ideas, Examples, Problems and Prospects**, Computer Program Testing, North-Holand Publishing Company, p. 129-148, 1981.

[Carniello, 2003a] CARNIELLO, A. **Teste baseado na Estrutura de Casos de Uso**, 2003, Dissertação (Mestrado em Pós Graduação em Engenharia Elétrica) – Departamento de Engenharia de Computação e Automação Industrial, UNICAMP, Campinas.

[Carniello, 2003b] CARNIELLO, A. **Teste baseado em Casos de Uso**, 2003, Boletim de Pesquisa e Desenvolvimento – Embrapa, Campinas, São Paulo.

[Chow, 1978] CHOW, T. S. **Testing Software Design Modeled by Finite-State Machines**. IEEE Transactions on Software Engineering, v. 4, n. 3, p. 178-187, 1978.

[Colanzi, 1999] COLANZI, T. E. **Uma Abordagem Integrada de Desenvolvimento e Teste de Software Baseada na UML**, 1999, 143 p. Dissertação (Mestrado em Ciência da Computação) - Instituto de Ciências Matemáticas e Computação, USP, São Carlos.

[DeMarco, 1979] DeMARCO, T. **Structured Analysis and System Specification**, Prentice-Hall, 1979.

[Eclipse, 2010] **Documentação do Software Eclipse**, 2010. Disponível em: <http://www.eclipse.org/> Acesso em: 12 mar 2010.

[Fagan, 1976] FAGAN, M. E. **Design and Code Inspections to Reduce Errors in Programa Development**. IBM Systems Journal, v. 15, n. 3, p. 182-211, 1976.

[Fagan, 1986] FAGAN, M. E. **Advances in Software Inspections**. IEEE Transactions Software Engineering, v. 12, n. 7, p. 744-751, 1986.

-[Fujiwara et al., 1991] FUJIWARA, S. et. al., **Test Selection Based on Finite-State Models**, IEEE Transactions on Software Engineering, v. 17, n. 6, June 1991, p. 591-603.

[Furlan, 1998] FURLAN, J. D. **Modelagem de Objetos através da UML**. São Paulo: Makron Books, 1998, 329 p.

-[Gönenç, 1970] GÖNENÇ, G., **A Method for Design of Fault-Detection Experiments**, IEEE Transactions on Computers, v. C-19, n. 6, 1970, p.551-558.

[Hartmann et al., 2005] HARTMANN, J.; VIEIRA, M.; FOSTER, H.; RUDER, A. **A UML-based approach to system testing**, Innovations Syst Softw Eng, USA, 2005.

-[IEEE, 1990] **IEEE Standard Glossary of Software Engineering Terminology**, Std 610.12-1990, 1990.

[IEEE, 1998] **IEEE Recommended Pratice for Software Requirements Specifications**, Std 830-1998, 1998.

[Jacobson et al., 1992] JACOBSON, I. **Object-Oriented Software Engineering: A Use Case Driven Approach**, Addison-Wesley Publish Company, 1992.

[Karner, 1993] KARNER, G. **Resource Estimation for Objectory Projects**, 1993. Disponível em: <http://www.bfpug.com.br/Artigos/UCP/Karner%20-%20Resource%20Estimation%20for%20Objectory%20Projects.doc>. Acesso em: 02 mai. 2009.

[Kitchenham, 2004] KITCHENHAM, B. **Procedures for Performing Systematic Reviews**. Joint Technical Report SE Group, Depart. Computer Science Keele University, United King and Empirical Software Engineering, National ICT, Australia, 2004.

[Kawai, 2005] KAWAI, K. K. **Diretrizes para Elaboração de Documento de Requisitos com ênfase nos Requisitos Funcionais**, 2005. Dissertação (Mestrado em Pós Graduação em Ciência da Computação) – Universidade Federal de São Carlos, São Carlos.

[Kulak & Guiney, 2000] KULAK, D.; GUINEY, E. **Use Cases: Requirements in Context**. Addison-Wesley, 2000. 329 p.

[Maldonado, 1991] MALDONADO, J.C. **Critérios Potenciais Usos: Uma Contribuição ao Teste Estrutural de Software**, 1991, 247 p. Tese (Doutorado em Engenharia Elétrica) - Faculdade de Engenharia Elétrica e Computação, UNICAMP, Campinas.

[Martins, 2007] MARTINS, M. D. C. **Uma Proposta de Ferramenta para o Cálculo dos Pontos de Caso de Uso**, 2007. Dissertação (Mestrado em Pós Graduação em Ciência da Computação) – Universidade Federal de São Carlos, São Carlos.

[Marucci, 2002a] MARUCCI, R. **Definição de uma Estratégia de Inspeção para um processo de desenvolvimento de Software Orientado a Objetos**. Dissertação de Mestrado. DC-UFSCar, Julho, 2002a.

[Marucci et al., 2002b] MARUCCI, R. A.; FABBRI, S. C. P. F.; MALDONADO, J. C.; TRAVASSOS, G. H. **OORTs/ProDeS: Definição de Técnicas de Leitura para um Processo de Software Orientado a Objetos**. In: SIMPÓSIO BRASILEIRO DE QUALIDADE DE SOFTWARE 2002, Gramado, Brasil.

[Naito & Tsunoyama, 1981] NAITO, S.; TSUNOYAMA, M. **Fault Detection for Sequential Machines by Transition-Tours**. In: Proceedings FCTS – Fault Tolerant Comput. Systems, p. 238-243, 1981.

[Nebut et al., 2006] NEBUT, C.; FLEUREY, F.; TRAON, Y. L.; Jézéquel, J. M. **Automatic Test Generation: A Use Case Driven Approach**. In: IEEE Transactions on Software Engineering, Vol. 32, Nº 3, 2006.

[OMG, 2010] **Unified Modeling Language Superstructure**. Disponível em: <http://www.omg.org/technology/documents/formal/uml.htm>. Acesso em: 18 nov. 2010.

[Porto et al., 2008] PORTO, D. P.; MENDONÇA, M.; FABBRI, S. C. P. F., **CRISTA - Code Reading Implemented with Stepwise Abstraction**. In: SIMPÓSIO BRASILEIRO DE ENGENHARIA DE SOFTWARE - SESSÃO DE FERRAMENTAS, XXII, Outubro, 2008, Campinas, Brasil. Anais... Campinas: SBC, 2008. p. 50-56.

[Porto, 2009] PORTO, D. P. **CRISTA: Um Apoio Computacional para Atividades de Inspeção e Compreensão de Código**. Dissertação de Mestrado. DC-UFSCar, Maio, 2009.

[Pressman, 2006] PRESSMAN, R. S. **Engenharia de Software**. 6ª edição. McGraw Hill, 2006, 720 p.

[Rapps & Weyuker, 1982] RAPPS, S; WEYUKER, E. J., **Data Flow Analysis Techniques for Test Data Selection**. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 1982, Tokio, Japan, p.272-278.

[Rational, 1999] **Documentação Oficial da UML**. Versão 1.3, junho, 1999, p. 808. Disponível em: <http://www-01.ibm.com/software/rational/uml/ documentation.html>. Acesso em: 31 mar. 2009.

[Rational, 2009] **Rational Unified Process**. Disponível em: <http://www-01.ibm.com/software/awdtools/rup/>. Acesso em: 04 mar. 2009.

[Runeson et al., 1999] RUNESON, P.; REGNELL, B.; THELIN, T. **Experimental and Simulation Analysis of Perspective Difference in Scenario-based Requirements Inspection**. REFSQ International Workshop on Requirements Engineering: Foundations of Software Quality, 1999, Lund, Suécia.

[Russel, 1991] RUSSEL, G. W. **Experience with Inspection in Ultralarge-Scale Developments**. In: IEEE SOFTWARE, Vol. 8. No. 1, 1991, p. 25-31.

[Ryser, 1995] RYSER, J. **An Integrated Formal Model of Scenarios Based on Statecharts**. In: Software Engineering – ESEC'95, 1995. Proceedings of the 5th European Software Engineering Conference. Springer (Lecture Notes in Computer Science 989)

[Ryser & Glinz, 1999a] 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: 04 jun. 2006.

[Ryser & Glinz, 1999b] RYSER, J.; GLINZ, M. **A Practical Approach to Validating and Testing Software Systems Using Scenarios**. In: Quality Week Europe, 1999b.

[Ryser & Glinz, 1999c] RYSER, J.; GLINZ, M. **A Scenario-Based Approach to Validating and Testing Software Systems Using Statecharts**. In: 12th International Conference on Software and Systems Engineering and their Applications, 1999c. Proceedings CNAM, Paris, France.

[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, 2000. Washington, D.C.

[Sabnani & Dahbura, 1998] SABNANI, K. K.; DAHBURA, A. **A Protocol Test Generation Procedure**, Computer Networks and ISDN Systems, v. 15, 4, 1998, p. 285-297.

[Schneider & Winters, 2001] SCHNEIDER, G.; WINTERS, J. P. **Applying Use Cases: A Practical Guide**. Second Edition, Addison-Wesley, 2001. 245 p.

[Shull et al., 2000] SHULL, F.; RUS, I.; BASILI, V. R. **How Perspective-Based Reading can Improve Requeriments Inspections**. IEEE Computer, v. 33, n.7, p. 73-79, 2000.

[Software Público, 2010] **Documentação de softwares públicos**, 2010. Disponível em: <http://www.softwarepublico.gov.br/>. Acesso em: 05 dec. 2010.

[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://net.educause.edu/ir/library/pdf/NCP08083B.pdf>. Acesso em: 13 mai. 2009.

[Standish Group, 2000] **The Chaos Report – 2000 – Standish Group**. Disponível em: <http://www.vertexlogic.com/processOnline/processData/documents/pdf/extreme_chaos.pdf>. Acesso em: 13 mai. 2009.

[Standish Group, 2004] **2004 Third Quarter Research Report - Standish Group**. Disponível em: <http://www.kean.edu/~rmelworm/3040-00/StandishGroup-04-q3-spotlight.pdf>. Acesso em: 13 mai. 2009.

[Standish Group, 2006] **The Chaos Report - 2006**. Disponível em: <http://www.sdtimes.com/link/30247>. Acesso em: 15 out. 2010.

[Sun, 2009a] **SUN**. Disponível em: <http://www.sun.com/java/everywhere/#facts>. Acesso em: 13 dez. 2009.

[Sun, 2009b] **SUN**. Disponível em: <http://www.sun.com/java/everywhere/#awards>. Acesso em: 13 dez. 2009.

[Thommazo, 2007] THOMMAZO, A. D. **Uma Proposta de Ferramenta para Rastreabilidade de Requisitos Baseada em Modelo de Casos de Uso**, 2007. Dissertação (Mestrado em Pós Graduação em Ciência da Computação) – Universidade Federal de São Carlos, São Carlos.

[Travassos et al., 1999a] TRAVASSOS, G. H.; SHULL, F.; CARVER, J.; BASILI, V.R. **Reading Techniques for OO Design Inspections**. In: ANNUAL SOFTWARE ENGINEERING WORKSHOP, 24., Greenbelt, USA, December, 1999a.

[Travassos et al., 1999b] TRAVASSOS, G. H.; SHULL, F.; FREDERICKS, M.; BASILI, V. R. **Detecting Defects in Object-Oriented Designs: Using Reading Techniques to Improve Software Quality**. In: CONFERENCE ON OBJECT-ORIENTED

PROGRAMMING, SYSTEMS, LANGUAGES, AND APLICATIONS (OOPSLA), 24., 1999b, Denver, Colorado, USA. Proceedings. Disponível em: <http://www.cs.umd.edu /projects/SoftEng/ESEG/papers/oopsla99.pdf>. Acesso em: 16 dez. 2002.

[Travassos et al., 2000] TRAVASSOS, G.H.; SHULL, F.; CARVER, J. **A Family of Reading Techniques for OO Design Inspections**. In: WORKSHOP DE QUALIDADE DE SOFTWARE, 7., 2000, João Pessoa, Brasil. Anais... João Pessoa: SBC, 2000, p. 225-237.

[Travassos et al., 2002] TRAVASSOS, G. H.; SHULL, F.; CARVER, J; BASILI, V. R. **Reading Techniques for OO Design Inspections**, 2002, 56 p. Technical Report CS-TR-4353, UMIACS-TR-2002-33, University of Maryland, Maryland. Disponível em: http://www.cs.umd.edu/Library/TRs/CS-TR-4353/CS-TR-4353.pdf. Acessado em 16/12/2002.

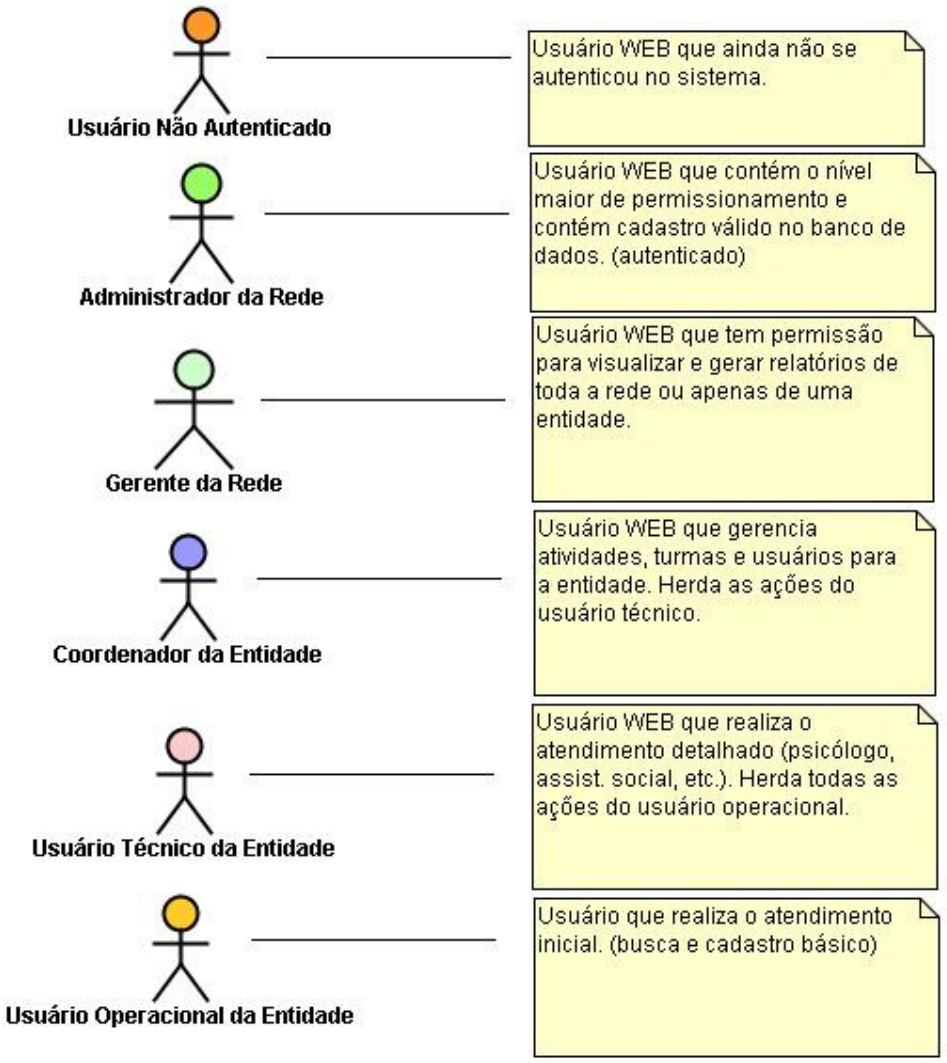
[Vieira et al., 2006] VIEIRA, M.; LEDUC, J.; HASLING, B.; SUBRAMANYAN, R.; KAZMEIER, J. **Automation of GUI Testing Using a Model-driven Approach**, First International Workshop on Automation of Software Test (*AST'06*), 2006.

[Wheeler et al., 1996] WHEELER, D. A.; BRYKCZYNSKI, B.; MEESON, R. N. J. **Introduction to Software Inspection**. In: Software Inspection: An Industry Best Practice. IEEE Computer Society Press, 1996, pp. 1-18.

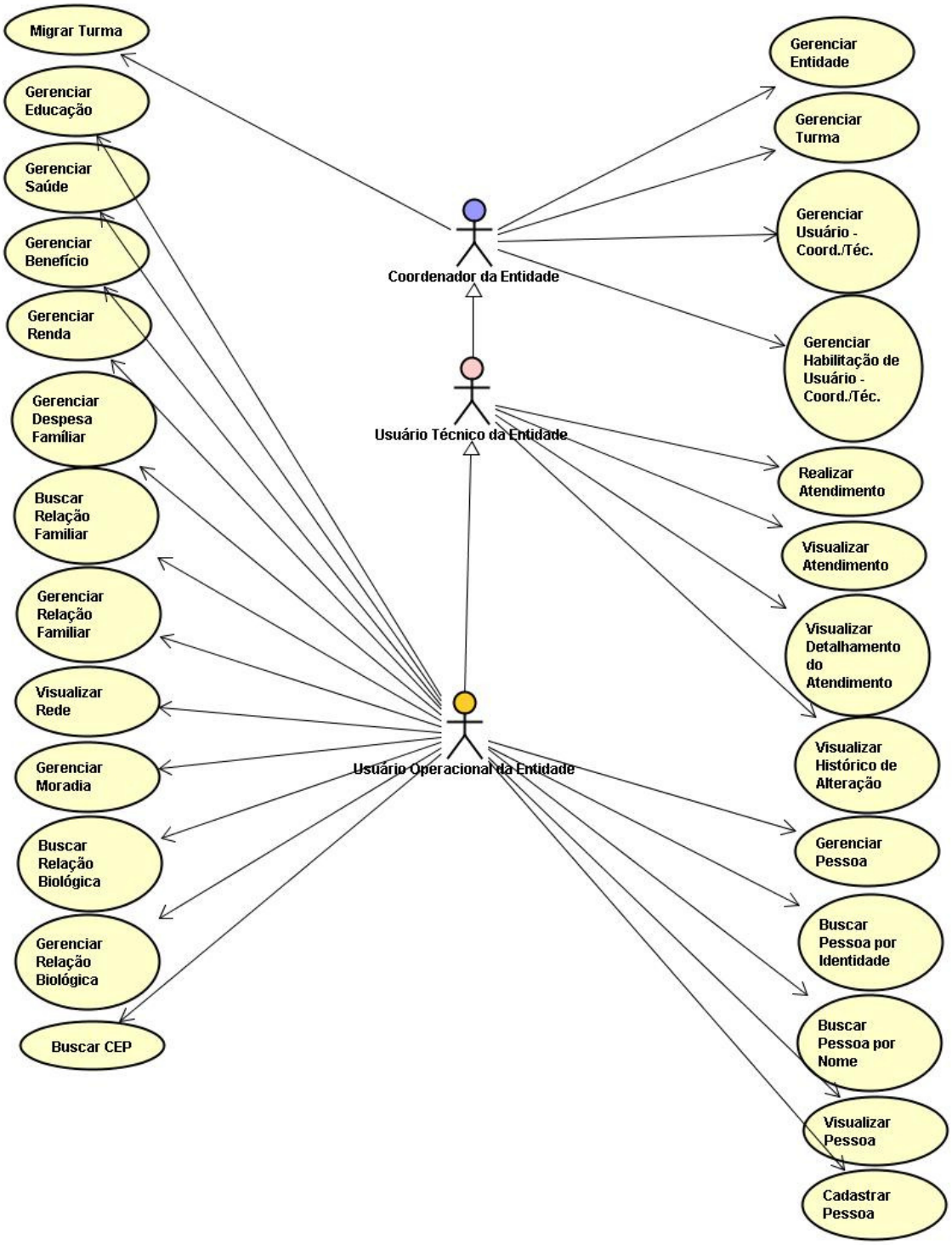
ANEXO I

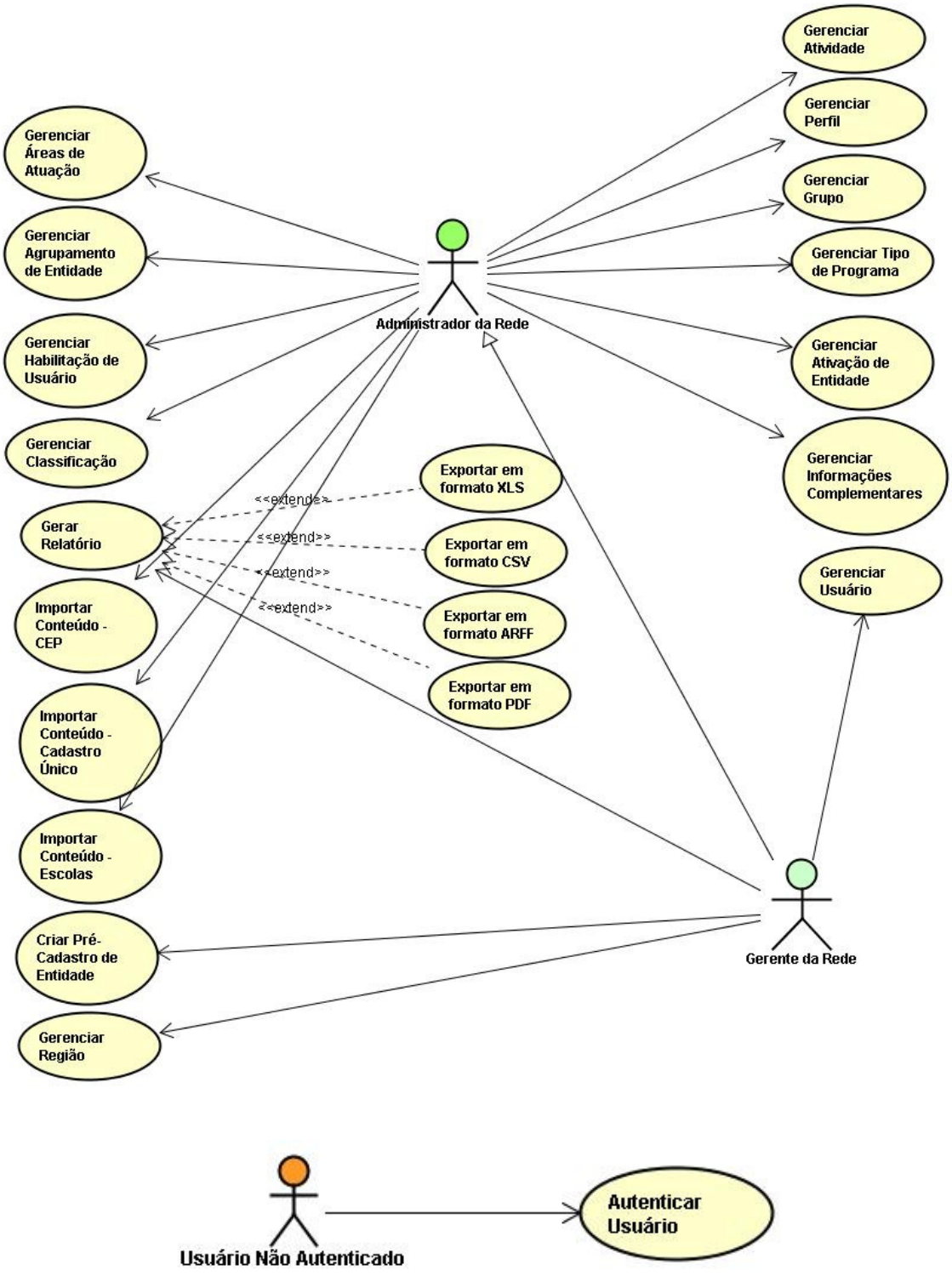
Exemplos de Documentos Fornecidos pela Empresa X

1. Lista de Atores



2. Diagramas de Casos de Uso




Gerenciar Atividade
Gerenciar Perfil
Gerenciar Grupo
Gerenciar Tipo de Programa
Gerenciar Áreas de Atuação
Gerenciar Agrupamento de Entidade
Gerenciar Habilitação de Usuário
Gerenciar Classificação
Administrador da Rede
Gerenciar Ativação de Entidade
Gerenciar Informações Complementares
Exportar em formato XLS
Exportar em formato CSV
Exportar em formato ARFF
Exportar em formato PDF
<<extend>>
<<extend>>
<<extend>>
<<extend>>
Gerar Relatório
Importar Conteúdo - CEP
Importar Conteúdo - Cadastro Único
Importar Conteúdo - Escolas
Gerenciar Usuário
Criar Pré-Cadastro de Entidade
Gerenciar Região
Gerente da Rede
Usuário Não Autenticado
Autenticar Usuário

3. Exemplo de Especificação de Caso de Uso

Caso de Uso

Realizar Atendimento

Histórico de Revisão

Data	Versão	Descrição	Responsáveis pela Versão
29/01/2008	1	Criação do documento	Jefferson Lima – W3S

1 DESCRIÇÃO

Esse caso de uso descreve uma ação concreta (funcionalidade do sistema) que permite ao Atendente realizar o atendimento no sistema.

2 ATORES

Ator	Descrição
Atendente	Usuário com atribuições de grupo: Operador, Técnico, Coordenador da Entidade, Gerente da rede e Administrador da rede.

3 PRÉ-CONDIÇÕES

Pré-condições
O Atendente deve estar autenticado e ter permissão para realizar atendimento. O Atendente deve ter feito o procedimento de busca de usuários e selecionado um para a visualização das informações.

4 FLUXO PRINCIPAL

1. O Atendente solicita a interface de realizar atendimento.
2. O sistema garante que a identificação da pessoa que foi enviada é válida.
3. O sistema carrega do banco de dados as informações solicitadas e exibe para o Atendente **[Ver Dicionário de Dados – Seção Realizar Atendimento].**
4. O sistema garante que as informações que serão exibidas para o atendente estão de acordo com o tipo de entidade que está sendo trabalhada.
5. O Atendente realiza o atendimento e submete ao sistema **[Ver Dicionário de Dados – Seção Realizar Atendimento] [Ver Especificação Suplementar – Seção Realizar Atendimento].**
6. O sistema valida as informações fornecidas e garante que estão de acordo com as regras de validação **[Ver Dicionário de Dados – Seção Realizar Atendimento].**
7. O sistema persiste as informações fornecidas no banco de dados.
8. O caso de uso encerra.

5 FLUXOS ALTERNATIVOS

Passo	Condição	Fluxo
2	O Atendente informou uma identificação de pessoa inválida.	1. O sistema informa ao Atendente sobre o ocorrido [FTN0004]. 2. O caso de uso encerra.
6	As informações não estão de acordo com as regras de validação.	1. O sistema informa ao Atendente sobre o ocorrido [FTN0003]. 2. O caso de uso encerra.
3, 7	Falha na comunicação com o banco de dados.	1. O sistema informa ao Atendente sobre o ocorrido. [FTN0002]. 2. O caso de uso encerra.

6 CASOS DE USO ESTENDIDOS

Caso de Uso Estendido	Passo no Fluxo Principal do Caso de Uso Estendido	Condição
Não há		

7 PÓS-CONDIÇÕES

Pós-condições
Sucesso: As informações relacionadas ao atendimento persistidas no banco de dados.
Erro: O sistema informa ao Atendente do problema ocorrido.

8 REQUISITOS NÃO-FUNCIONAIS

Classificação do Requisito	Medida

4. Exemplo do Plano de Teste e de Caso de Teste

39. Realizar atendimento: .....
39.1 – Realizar Atendimento (com sucesso).....130
39.2 – Realizar Atendimento (sem sucesso) – usuário não tem permissão de acesso .....
39.3 – Realizar Atendimento (sem sucesso) – tipo de atendimento inválido.....
39.4 – Realizar Atendimento (sem sucesso) – identificação da pessoa inválida .....
39.5 – Realizar Atendimento (sem sucesso) – banco de dados indisponível no passo 3.....
39.6 – Realizar Atendimento (sem sucesso) – banco de dados indisponível no passo 2.....
39.7 – Realizar Atendimento (com sucesso) – interfaces de atendimento.....
39.8 – Realizar Atendimento (com sucesso) – atendimento geral.....
39.9 – Realizar Atendimento (com sucesso) – confidencialidade por entidade.....
39.10 – Realizar Atendimento (com sucesso) – parte de grupo de usuário que pode alterar um nível de sigilo
39.11 – Realizar Atendimento (com sucesso) – sigilo público, entidade e perfil
39.12 – Realizar Atendimento (com sucesso) – perfil diferente do permitido .....
39.13 – Realizar Atendimento (com sucesso) – atendimento especial.....
39.14 – Realizar Atendimento (com sucesso) – atendimento grupo .....
39.15 – Realizar Atendimento (com sucesso) – atendimento encerrado.....

39.1 – Realizar Atendimento (com sucesso) – atendimento geral

Pré-condição: O usuário deve estar autenticado como coordenador da entidade (3)
Pré-condição: O usuário deve estar autenticado como técnico da entidade (4)

URL:http://www.wiface.com.br/rede-das-redes/attendance/attendance/person/x

Step Name	Description	Expected Result
Step 1	* Usuário solicita ao sistema a interface para realizar atendimento.	* O sistema exibe a interface para realizar atendimento.
Step 2	* Usuário informa: - Tipo de atendimento oferecido por sua entidade válido (1 – existente na combo) Atenção: tester verifica se todos os atendimentos listados são realmente da entidade. * Usuário aciona o botão “Próxima”.	- O sistema deve exibir uma interface para informar a data e se a informação é confidencial.
Step 3	* Usuário informa: - Uma data prevista para término válida (data válida, até 10 caracteres) - Informação confidencial válida. * Usuário aciona o botão “Salvar”.	* Sistema deve exibir o formulário de atendimento da sua entidade.

39.2 – Realizar Atendimento (com sucesso) – atendimento especial

Pré-condição: O usuário deve estar autenticado como coordenador da entidade (3)
Pré-condição: O usuário deve estar autenticado como técnico da entidade (4)

URL:http://www.wiface.com.br/rede-das-redes/attendance/attendance/person/x

Step Name	Description	Expected Result
Step 1	* Usuário solicita ao sistema a interface para realizar atendimento.	* O sistema exibe a interface para realizar atendimento.
Step 2	* Usuário informa: - Tipo de atendimento oferecido por sua entidade válido (1 – existente na combo) Atenção: tester verifica se todos os atendimentos listados são realmente da entidade. * Usuário aciona o botão “Próxima”.	- O sistema deve exibir uma interface para informar a data e se a informação é confidencial.
Step 3	* Usuário informa: - Uma data prevista para término válida (data válida, até 10 caracteres) - Informação confidencial válida. * Usuário aciona o botão “Salvar”.	* Sistema deve exibir o formulário de atendimento da sua entidade.

39.3 – Realizar Atendimento (com sucesso) – atendimento grupo

Pré-condição: O usuário deve estar autenticado como coordenador da entidade (3)
Pré-condição: O usuário deve estar autenticado como técnico da entidade (4)

URL:http://www.wiface.com.br/rede-das-redes/attendance/attendance/person/x

Step Name	Description	Expected Result
Step 1	* Usuário solicita ao sistema a interface para realizar atendimento.	* O sistema exibe a interface para realizar atendimento.
Step 2	* Usuário informa: - Tipo de atendimento oferecido por sua entidade válido (1 – existente na combo) Atenção: tester verifica se todos os atendimentos listados são realmente da entidade. * Usuário aciona o botão “Próxima”.	- O sistema deve exibir uma interface para informar a data e se a informação é confidencial.
Step 3	* Usuário informa: - Uma data prevista para término válida (data válida, até 10 caracteres) - Informação confidencial válida. * Usuário aciona o botão “Salvar”.	* Sistema deve exibir o formulário de atendimento da sua entidade.