



UMA ABORDAGEM PARA ESPECIFICAÇÃO DE REQUISITOS DIRIGIDA POR
MODELOS INTEGRADA AO CONTROLE DA QUALIDADE DE APLICAÇÕES WEB

Jobson Luiz Massollar da Silva

Tese de Doutorado apresentada ao Programa de Pós-Graduação em Engenharia de Sistemas e Computação, COPPE, da Universidade Federal do Rio de Janeiro, como parte dos requisitos necessários à obtenção do título de Doutor em Engenharia de Sistemas e Computação.

Orientador: Guilherme Horta Travassos

Rio de Janeiro

Abril de 2011

UMA ABORDAGEM PARA ESPECIFICAÇÃO DE REQUISITOS DIRIGIDA POR
MODELOS INTEGRADA AO CONTROLE DA QUALIDADE DE APLICAÇÕES WEB

Jobson Luiz Massollar da Silva

TESE SUBMETIDA AO CORPO DOCENTE DO INSTITUTO ALBERTO LUIZ
COIMBRA DE PÓS-GRADUAÇÃO E PESQUISA DE ENGENHARIA (COPPE) DA
UNIVERSIDADE FEDERAL DO RIO DE JANEIRO COMO PARTE DOS REQUISITOS
NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE DOUTOR EM CIÊNCIAS EM
ENGENHARIA DE SISTEMAS E COMPUTAÇÃO.

Examinada por:

Prof. Guilherme Horta Travassos, D.Sc.

Prof. Toacy Cavalcante de Oliveira, D.Sc.

Prof. Geraldo Bonorino Xexéo, D.Sc.

Prof^a. Maria Emília Xavier Mendes, D.Sc.

Prof. Sérgio Castelo Branco Soares, D. Sc.

RIO DE JANEIRO, RJ – BRASIL

ABRIL DE 2011

Silva, Jobson Luiz Massollar da

Uma Abordagem para Especificação de Requisitos Dirigida por Modelos Integrada ao Controle da Qualidade de Aplicações Web / Jobson Luiz Massollar da Silva – Rio de Janeiro: UFRJ/COPPE, 2011.

XV, 253 p.: il.; 29,7 cm.

Orientador: Guilherme Horta Travassos.

Tese (doutorado) – UFRJ/COPPE/Programa de Engenharia de Sistemas e Computação, 2011.

Referências Bibliográficas: p. 184-196.

1. Engenharia de Aplicações Web. 2. Engenharia de Requisitos. 3. Especificação e Verificação de Requisitos. 4. Engenharia de Software Experimental. I. Travassos, Guilherme Horta II. Universidade Federal do Rio de Janeiro, COPPE, Programa de Engenharia de Sistemas e Computação. III. Título.

Ao meu filho Lucas, pelo seu sorriso, carinho e compreensão.
À minha esposa Jane, minha companheira e incentivadora, e sem a qual essa jornada
seria impossível.

Agradecimentos

Em primeiro lugar a Deus, por me dar saúde e iluminar meu caminho nesta e em tantas outras jornadas.

À minha mãe, Antônia, pelo seu amor e incentivo para que eu sempre siga em frente e busque novos desafios.

Ao meu pai, Luiz Caetano, que apesar de não estar mais entre nós, sempre lutou muito para me dar condições de chegar até aqui.

Ao meu orientador, prof. Guilherme Horta Travassos, que apesar da agenda geralmente apertada nunca deixou de estar presente e sempre se mostrou disponível e comprometido com qualidade final desta tese. Espero que este seja apenas o primeiro de muitos outros trabalhos que possamos realizar juntos.

À prof^a. Ana Regina Cavalcanti da Rocha que me recebeu com muito entusiasmo nesse retorno à COPPE, após ser minha orientadora no mestrado, e por fazer parte da minha banca de exame de qualificação.

Aos professores Emilia Mendes, Toacy Cavalcante, Geraldo Xexéo e Sérgio Soares por participarem da minha banca de defesa de doutorado.

Ao meus amigos e companheiros da COPPE, alguns já egressos, Tayana Conte, Rodrigo Spínola, Arilo Claudio, Marcos Kalinowski, Paulo Sérgio, Vitor Lopes, Marco Antônio, José Fortuna, Taisa Guidini, Breno França, Rafael Mello, Karen Nakazato, Silvia Santa Isabel, Wallace Martinho, Rafael Espírito Santo, Verônica Taquette, Priscila Pecchio e Lucas Paes pelo ótimo ambiente de trabalho e por todos os excelentes momentos vividos durante esses anos.

Ao pessoal administrativo do PESC, Claudia Prata, Maria Mercedes, Solange Santos e Sônia Galliano pelo carinho e atenção.

Aos alunos dos cursos de Engenharia de Software OO (ESOO), Interface Humano-Computador (IHC) e Verificação, Validação e Testes (VV&T) pela participação nos estudos que compõem esta tese.

À CAPES, pelo apoio financeiro durante o doutorado.

Resumo da Tese apresentada à COPPE/UFRJ como parte dos requisitos necessários para a obtenção do grau de Doutor em Ciências (D.Sc.)

UMA ABORDAGEM PARA ESPECIFICAÇÃO DE REQUISITOS DIRIGIDA POR MODELOS INTEGRADA AO CONTROLE DA QUALIDADE DE APLICAÇÕES WEB

Jobson Luiz Massollar da Silva

Abril/2011

Orientador: Guilherme Horta Travassos

Programa: Engenharia de Sistemas e Computação

Esta tese propõe uma abordagem para especificação estruturada de requisitos funcionais de aplicações Web que contempla questões relacionadas à garantia da qualidade dos requisitos e do produto final através de um arcabouço para análise, classificação e especificação de requisitos alinhado aos conceitos explorados pelos métodos Web contemporâneos. Essa abordagem explora o tratamento das perspectivas de projeto Web (conceituação, navegação e apresentação) desde a fase inicial do projeto e é apoiada por um metamodelo cuja finalidade é aprimorar o rigor da especificação funcional e apoiar atividades de garantia da qualidade da especificação dos casos de uso. A concepção da abordagem está baseada nos resultados de um estudo secundário anterior, de uma revisão da literatura técnica baseada nos conceitos de revisão sistemática e de três estudos primários (estudos de observação), enquanto a sua avaliação foi efetuada através de dois estudos primários (estudos de viabilidade). Os resultados da avaliação experimental indicam que o uso da abordagem proposta para especificação de requisitos de aplicações Web contribui para a redução de defeitos nas especificações de caso de uso quando comparada ao uso de uma abordagem *ad-hoc*.

Abstract of Thesis presented to COPPE/UFRJ as a partial fulfillment of the requirements for the degree of Doctor of Science (D.Sc.)

A MODEL-DRIVEN APPROACH FOR REQUIREMENTS SPECIFICATION
INTEGRATED TO WEB APPLICATIONS QUALITY ASSURANCE

Jobson Luiz Massollar da Silva

April/2011

Advisor: Guilherme Horta Travassos

Department: Computer Science and Systems Engineering

This thesis proposes a structured approach to support the specification of functional requirements aiming to assure their quality. It makes use of a framework for analysis, classification and specification of functional requirements aligned with the concepts promoted by contemporary Web development methods. This approach explores web design perspectives (concept, navigation and presentation) through the requirements specification. A metamodel designed to guarantee the specification accuracy and to support the specification quality assurance activities makes part of it. Its inception has been based on the results of a preliminary secondary study (systematic review), the conduction of a technical literature review based on systematic reviews concepts and three primary studies (observational studies), whilst its assessment has been conducted through two experimental studies (feasibility studies). The experimental studies have indicated that this approach contributes to the reduction of defects on Web application's use case specifications when compared to an *ad-hoc* approach.

ÍNDICE

1	Introdução.....	1
1.1	Motivação e Contexto.....	1
1.2	Problema.....	5
1.3	Questões de Pesquisa	6
1.4	Objetivos	7
1.5	Metodologia de Trabalho.....	8
1.6	Organização.....	10
2	Estudos Preliminares	11
2.1	Introdução	11
2.2	Primeiro Estudo de Observação.....	11
2.2.1	Objetivo	12
2.2.2	Contexto.....	12
2.2.3	Projeto do estudo	13
2.2.4	Instrumentação.....	16
2.2.5	Execução.....	16
2.3	Segundo Estudo de Observação.....	18
2.3.1	Objetivo	18
2.3.2	Contexto.....	18
2.3.3	Projeto do Estudo.....	18
2.3.4	Instrumentação.....	19
2.3.5	Execução.....	20
2.4	Terceiro Estudo de Observação	21
2.4.1	Objetivo	21
2.4.2	Contexto.....	21
2.4.3	Projeto do Estudo.....	22
2.4.4	Instrumentação.....	22
2.4.5	Execução.....	23
2.5	Análise dos Resultados dos Estudos.....	26
2.5.1	Análise dos Resultados Quantitativos.....	26
2.5.2	Análise dos Dados Qualitativos	29
2.6	Ameaças à Validade dos Estudos	30
2.7	Conclusão	32
3	Métodos de Desenvolvimento de Aplicações Web.....	34
3.1	Introdução	34
3.2	Revisão da Literatura sobre Métodos de Desenvolvimento de Aplicações Web	35
3.2.1	Protocolo da Revisão.....	35
3.2.2	Execução da Revisão.....	38
3.2.3	Resultados obtidos com a Análise dos Estudos.....	39
3.3	Resumo Comparativo dos Métodos Web	53
3.4	Contribuições da Revisão.....	54
3.5	Conclusão	55
4	Abordagem para Especificação de Requisitos	57
4.1	Introdução	57
4.2	Visão Geral da Abordagem	59
4.2.1	Perspectiva de Conceituação	60
4.2.2	Perspectiva de Navegação.....	61
4.2.3	Perspectiva de Apresentação.....	62
4.2.4	Linguagem de Modelagem	62

4.3	Processo de Engenharia de Requisitos.....	62
4.4	Classificação de Requisitos.....	65
4.5	Especificação de Requisitos.....	67
4.5.1	Modelos Conceituais	68
4.5.2	Regras de Negócio.....	71
4.5.3	Modelo de Casos de Uso	79
4.6	Inspeção de Requisitos	87
4.7	Geração de Casos e Procedimentos de Testes Funcionais.....	88
4.8	Conclusão	90
5	Metamodelo para Especificação de Casos de Uso	93
5.1	Introdução	93
5.2	Diagrama de Atividades e Casos de Uso	94
5.3	O Metamodelo UCMModel	96
5.3.1	Caso de Uso e seus Elementos Básicos	99
5.3.2	Regras.....	102
5.3.3	Fluxos do Caso de Uso	104
5.3.4	Ações do Caso de Uso.....	107
5.3.5	Combinando Ações na Descrição do Comportamento.....	112
5.3.6	Restrições do Metamodelo UCMModel	114
5.3.7	Perfil UML.....	121
5.4	Orientações para Especificação de Casos de Uso no UCMModel.....	122
5.4.1	Orientações Gerais.....	122
5.4.2	Orientações relacionadas à Ação do Ator.....	122
5.4.3	Orientações relacionadas à Ação do sistema	123
5.4.4	Orientações relacionadas à Resposta do Sistema.....	123
5.5	Exemplo de Caso de Uso especificado com o UCMModel	124
5.6	Conclusão	127
6	Infra-estrutura de Apoio à Especificação de Casos de Uso e Garantia da Qualidade	130
6.1	Introdução	130
6.2	Ferramenta UseCaseAgent.....	132
6.2.1	Cenários de uso da UseCaseAgent.....	135
6.2.2	Trabalhos Relacionados	144
6.3	Ferramenta ModelIT ²	146
6.4	Conclusão	156
7	Avaliação Experimental.....	157
7.1	Introdução	157
7.2	Primeiro Estudo.....	160
7.2.1	Objetivo	160
7.2.2	Seleção do Contexto	160
7.2.3	Seleção de Variáveis.....	162
7.2.4	Participantes.....	162
7.2.5	Projeto do Estudo	162
7.2.6	Instrumentação.....	164
7.2.7	Preparação.....	164
7.2.8	Execução.....	165
7.2.9	Análise Quantitativa.....	165
7.2.10	Análise Qualitativa	167
7.3	Segundo Estudo.....	168
7.3.1	Objetivo	168
7.3.2	Seleção do Contexto	168
7.3.3	Seleção de Variáveis.....	169

7.3.4	Participantes.....	169
7.3.5	Projeto do Estudo	169
7.3.6	Instrumentação.....	170
7.3.7	Preparação.....	170
7.3.8	Execução.....	171
7.3.9	Validação dos Dados.....	171
7.3.10	Resultados.....	171
7.4	Ameaças à Validade dos Estudos	175
7.5	Conclusão	177
8	Conclusão e Trabalhos Futuros	179
8.1	Considerações Finais	179
8.2	Contribuições da Pesquisa.....	179
8.3	Limitações	181
8.4	Questões em Aberto.....	181
8.5	Trabalhos Futuros	182
	REFERÊNCIAS BIBLIOGRÁFICAS.....	184
	APÊNDICE A – Instrumentos dos Estudos de Observação Preliminares.....	197
A.1.	Questionário do 1º estudo de observação preliminar.....	197
A.2.	Questionário do 3º estudo de observação preliminar.....	198
	APÊNDICE B – Análise Qualitativa.....	200
B.1	Introdução	200
B.2	Análise dos Dados	200
	APÊNDICE C – Manual do Usuário do UseCaseAgent.....	219
C.1	Introdução	219
C.2	Por que BOUML ?	220
C.3	Instalação e Configuração.....	220
C.4	Criação dos Atores	221
C.5	Criação de um Caso de Uso.....	222
C.6	Dados Gerais do Caso de Uso	224
C.7	Atores do Caso de Uso	225
C.8	Fluxo Principal.....	225
C.9	Fluxos Alternativos	227
C.10	Fluxos de Exceção	229
C.11	Regras	229
C.12	Verificação Sintática	230
C.13	Edição de um Caso de Uso	230
C.14	Impressão dos Casos de Uso.....	231
C.15	Verificação Sintática do Diagrama de Atividades.....	233
C.16	Restrições do UseCaseAgent.....	234
C.17	Combinação de Ações na Especificação do Caso de Uso	234
C.18	Orientações para Descrição de Casos de Uso	236
C.19	Mensagens de Erro	239
A.19.1	Mensagens de Erro na Verificação de Diagramas	239
A.19.2	Mensagens de Erro na Especificação do Caso de Uso.....	241
	APÊNDICE D – Exemplo de Casos e Procedimentos de Teste Funcional.....	244
1.	Test Cases	244
1.1.	Test Case 'TC001_TP1'	244
1.1.1.	Test Case Path	244

1.1.2. Test Case Steps	245
1.1.3. Test Case Data Variations	245
1.2. Test Case 'TC001_TP2'	246
1.2.1. Test Case Path	246
1.2.2. Test Case Steps	247
1.2.3. Test Case Data Variations	247
1.3. Test Case 'TC001_TP3'	248
1.3.1. Test Case Path	248
1.3.2. Test Case Steps	249
1.3.3. Test Case Data Variations	249
2.1. Class Diagram 'Category Diagram'	250
2.1.1. Category 'Senha1'	250
2.1.2. Category 'Opcao1'	250
2.1.3. Category 'DadosPagamento1'	250
2.1.4. Category 'A2'	250
2.1.5. Category 'Boleto1'	251
2.1.6. Category 'Erro1'	251

ÍNDICE DE FIGURAS

Figura 1-1 - Visão geral da metodologia adotada.....	9
Figura 3-1 - Metamodelo de requisitos do WRM.....	49
Figura 4-1 - Perspectivas de projeto x características dos modelos x fases de desenvolvimento (adaptada de SCHWINGER e KOCH (2006)).....	60
Figura 4-2 - Atividades previstas na abordagem proposta	64
Figura 4-3- Mapeamento entre a taxonomia de requisitos de ESCALONA e KOCH (2004) e as perspectivas de projeto Web e visões de modelagem.....	66
Figura 4-4 – Máquina de estado representando o ciclo de vida da entidade Solicitação de Movimentação	70
Figura 4-5 – Diagrama de atividades representando o processo de administração de solicitação de movimentação	71
Figura 4-6 - Exemplo de uso do gabarito proposto pela abordagem	82
Figura 4-7 – Diagrama de atividades com a especificação do caso de uso “Concluir compra”	85
Figura 4-8 - Diagrama de atividades detalhando o comportamento da ação "Calcular valor do pedido" do caso de uso da Figura 4-7	86
Figura 5-1- Visão parcial do metamodelo da UML 2 com os elementos do diagrama de atividades usados na definição do metamodelo UCMModel	97
Figura 5-2 - Exemplo da análise realizada para definir quais elementos da UML seriam especializados para criação dos elementos da UCMModel	97
Figura 5-3 - Metamodelo UCMModel e seu relacionamento com o metamodelo da UML	98
Figura 5-4 – Visão parcial do metamodelo UCMModel referente ao caso de uso e seus elementos básicos	100
Figura 5-5 – Padrão de representação de um caso de uso e seus elementos básicos usando o diagrama de atividades da UML 2	101
Figura 5-6 – Visão parcial do metamodelo UCMModel referente às regras de negócio/domínio e sua relação com o metamodelo da UML	103
Figura 5-7 – Padrão de representação de um caso de uso e suas regras de negócio/domínio usando o diagrama de atividades da UML 2	104
Figura 5-8 – Visão parcial do metamodelo UCMModel referente aos fluxos principal, alternativos e de exceção e sua relação com o metamodelo da UML.....	105
Figura 5-9 – Padrão de representação de um caso de uso e seus fluxos principal, alternativos e de exceção usando diagrama de atividades da UML	106
Figura 5-10 – Visão parcial do metamodelo UCMModel referente às ações que compõem os fluxos do caso de uso e a sua relação com o metamodelo da UML.....	110
Figura 5-11 - Transições básicas permitidas entre as ações ActorAction, SystemAction e SystemResponse.....	112
Figura 5-12 - Transições permitidas a partir de uma ActorAction com fluxo alternativo	113
Figura 5-13 - Transições permitidas a partir de uma SystemAction com fluxo alternativo	114
Figura 5-14 Transições permitidas a partir de uma SystemResponse com fluxo alternativo	114
Figura 5-15 – Estereótipos e tagged-values para definição de diagramas de atividades segundo o UCMModel	121
Figura 5-16 - Diagrama de atividades com a descrição do caso de uso da Figura 5-17	125
Figura 5-17 – Descrição textual do caso de uso “Autenticar usuário” de acordo com o gabarito apresentado na Tabela 4-10 – página 81	126
Figura 6-1 – Ciclo de uso das ferramentas previstas na abordagem com insumos e produtos.....	132

Figura 6-2 - Tela principal da ferramenta BOUML com a estrutura de pacotes	134
Figura 6-3 - Tela inicial da UseCaseAgent com os elementos básicos do caso de uso	136
Figura 6-4 - Tela da UseCaseAgent para definição dos atores do caso de uso	136
Figura 6-5 - Tela da UseCaseAgent com o fluxo principal do caso de uso "Pagar boleto bancário"	137
Figura 6-6 - Tela da UseCaseAgent com o fluxo alternativo A2 do caso de uso "Pagar boleto bancário"	137
Figura 6-7 - Tela da UseCaseAgent com as regras do caso de uso "Pagar boleto bancário"	138
Figura 6-8 - Tela da UseCaseAgent com a lista de defeitos do caso de uso "Pagar boleto bancário"	138
Figura 6-9 - Diagrama de atividades com a especificação do caso de uso "Pagar boleto bancário"	140
Figura 6-10 - Tela da UseCaseAgent para seleção dos casos de uso que farão parte do documento de especificação.....	141
Figura 6-11 - Tela da UseCaseAgent para impressão das especificações de casos de uso em formato RTF	141
Figura 6-12 - Representação em formato texto da especificação do caso de uso "Pagar boleto bancário"	143
Figura 6-13 – Tela inicial da ModelT ² para seleção dos casos de uso	149
Figura 6-14 – Segunda tela da ModelT ² que indica os casos de uso que estão aptos a terem seus modelos de teste gerados	149
Figura 6-15 - Modelo de testes gerado pela ModelT2 referente ao caso de uso "Pagar boleto bancário"	150
Figura 6-16 - Modelo de testes referente ao caso de uso "Pagar boleto bancário" alterado pelo analista de testes	152
Figura 6-17 – Trecho do documento de casos e procedimentos de testes do caso de uso "Pagar boleto bancário", com um caminho e os valores para percorrer esse caminho.....	155
Figura 7-1 - xxx.....	157
Figura 7-2 – Visão geral dos estudos de avaliação.....	159
Figura 7-3 - Avaliação do nível de complexidade dos casos de uso selecionados....	162
Figura 7-4 - Boxplots relativos aos defeitos reportados nas especificações de caso de uso construídas com as ferramentas EDA e UCA (tela da ferramenta JMP v.4)	173

ÍNDICE DE TABELAS

Tabela 2-1 - Atividades e artefatos do 1º estudo de observação.....	14
Tabela 2-2 - Medidas diretas extraídas dos artefatos de projeto do 1º estudo	16
Tabela 2-3 - Atividades e artefatos do 2º estudo de observação.....	18
Tabela 2-4 - Dificuldades relatadas pelos participantes do 2º estudo.....	20
Tabela 2-5 - Dados quantitativos do 2º estudo.....	20
Tabela 2-6 - Atuação dos desenvolvedores por modelo	23
Tabela 2-7 - Percepção de esforço	23
Tabela 2-8 - Distribuição de percepção de esforço dos participantes que criaram os artefatos.....	24
Tabela 2-9 - Distribuição de importância percebida pelos participantes por diagrama	24
Tabela 2-10 - Fatores de dificuldade apontados pelos participantes.....	25
Tabela 3-1 - Resumo da execução da revisão da literatura	38
Tabela 3-2 - Distribuição dos 21 artigos aprovados pelos métodos Web	38
Tabela 3-3 - Estereótipos do metamodelo do WebRE	46
Tabela 3-4 - Resumo dos métodos Web analisados.....	53
Tabela 4-1 - Artefatos para representação dos requisitos na perspectiva de conceituação	61
Tabela 4-2 - Artefatos para representação dos requisitos na perspectiva de navegação	61
Tabela 4-3 - Artefatos para representação dos requisitos na perspectiva de apresentação.....	62
Tabela 4-4 – Algumas diferenças entre Requisitos Funcionais e Regras de Negócio .	72
Tabela 4-5 - Sugestão para prefixação de regras no SBVR (OMG, 2008; BOLLEM, 2008)	73
Tabela 4-6 - Exemplo de especificação de regras conceituais segundo a SBVR.....	74
Tabela 4-7 - Regras comportamentais e possíveis contextos de atuação.....	75
Tabela 4-8 - Exemplo de especificação de regras de navegação segundo a SBVR ...	76
Tabela 4-9 - Exemplo de especificação de regras de apresentação segundo a SBVR	78
Tabela 4-10 - Gabarito de caso de uso adotado na abordagem proposta.....	81
Tabela 5-1 – Subconjunto de elementos do diagrama de atividades da UML explorados no contexto deste trabalho	95
Tabela 5-2 - Mapeamento entre as informações básicas do caso de uso e os elementos do metamodelo UCMModel.....	99
Tabela 5-3 - Estereótipos usados na definição do caso de uso e seus elementos básicos	100
Tabela 5-4 – Classificação da regras de negócio/domínio de acordo com as perspectivas de projeto Web.....	102
Tabela 5-5 - Mapeamento entre as regras e os elementos do metamodelo UCMModel	103
Tabela 5-6 - Estereótipos usados na definição de regras de negócio/domínio.....	103
Tabela 5-7 - Mapeamento entre os fluxos do caso de uso e os elementos do metamodelo UCMModel	105
Tabela 5-8 - Principais tipos de ações previstos pelo metamodelo UCMModel.....	108
Tabela 5-10 - Tipos de ações para inclusão e extensão de casos de uso.....	109
Tabela 5-11 - Mapeamento entre as ações do caso de uso e os elementos do metamodelo UCMModel	109
Tabela 5-12 - Estereótipos usados na definição das ações	111
Tabela 5-13- Restrições aplicadas ao caso de uso "Autenticar usuário" da Figura 5-16	126
Tabela 6-1 – Transformação dos elementos do metamodelo UCMModel para o metamodelo TDE.....	148
Tabela 7-1 - Cálculo da complexidade dos casos de uso	161

Tabela 7-2 - Conjuntos de casos de uso balanceados pelo nível de complexidade ..	163
Tabela 7-3 - Conjuntos de casos de uso atribuídos a cada participante por rodada .	163
Tabela 7-4 - Tempos reportados pelos participantes para especificação dos casos de uso.....	166
Tabela 7-5 - Distribuição dos casos de uso pelos participantes do segundo estudo .	170
Tabela 7-6 - Número de defeitos reportados por caso de uso e ferramenta.....	172
Tabela 7-7 - Defeitos nos requisitos detectados pela ferramenta UseCaseAgent	174

1 Introdução

Neste capítulo são apresentados o contexto do trabalho, o que motivou esta pesquisa e a questão de investigação. São também apresentados os objetivos, a metodologia de pesquisa adotada e a organização deste texto.

1.1 Motivação e Contexto

Nos últimos anos a complexidade e o escopo de atuação das aplicações Web vêm crescendo rapidamente e, atualmente, essa categoria de software tem apoiado o relacionamento entre empresas e indivíduos em atividades do dia-a-dia relacionadas ao trabalho ou lazer (BERNARDI *et al.*, 2010). Atualmente várias indústrias ligadas ao comércio, manufatura, logística, sistema financeiro e educação exploram as características da Web como ubiqüidade, padrões abertos e facilidade de acesso a informações e serviços, para criar novas oportunidades de negócio e expandir seus mercados. Nesse contexto, o aumento da complexidade de tais aplicações vem sendo acompanhado pela crescente demanda por aplicações Web confiáveis, seguras e usáveis – três critérios de qualidade dominantes segundo OFFUT (2002) – tornando essa categoria de aplicações uma das mais importantes da indústria em geral e da indústria de software em particular.

No contexto deste trabalho é importante definir, primeiramente, o que são aplicações Web. Apesar de não existir um consenso da comunidade científica em torno de uma definição em comum do que seria uma aplicação Web, KAPPEL *et al.* (2006) apresentam a seguinte definição: “*Uma aplicação Web é um sistema de software baseado em tecnologias e padrões do World Wide Web Consortium (W3C)¹ que provê recursos específicos da Web, como conteúdo e serviços, através de uma interface de usuário, o browser Web*”. Contudo, essa definição parece restringir a classe de aplicações Web por incluir o *browser* como um elemento indispensável da interface dessas aplicações. Apesar da presença maciça do *browser* na grande maioria das aplicações Web, não podemos excluir dessa definição as aplicações que utilizam a infra-estrutura e os conceitos da Web, mas abrem mão do *browser* como

¹ www.w3c.org

componente principal para apresentação e exploração das informações disponibilizadas pela aplicação. Essa é uma realidade que tem sido observada na prática pelo grupo de Engenharia de Software Experimental da COPPE/UFRJ envolvido com projetos de pesquisa que demandam a especificação de aplicações Web envolvendo workflow científico e características de ubiqüidade computacional. Neste cenário foi possível identificar maiores preocupações relacionadas a questões arquiteturais, integração de serviços e manipulação de grandes volumes de dados não estruturados em detrimento de outras relacionadas puramente com a exploração da informação através do browser, apesar da existência destas com um menor grau de relevância.

Assim, no contexto deste trabalho, foi adotada uma definição adaptada de KAPPEL *et al.* (2006):

*“Uma aplicação Web é um sistema de software baseado em tecnologias e padrões do World Wide Web Consortium (W3C) que provê recursos específicos da Web, como conteúdo e serviços, através de um **cliente Web**”*

Essa definição pode ser complementada ressaltando que o foco deste trabalho está nas aplicações Web que representam sistemas de software complexos, orientados a processos ou a dados e que integram a exploração não-linear da informação com a capacidade de execução de processos de forma distribuída. É importante notar que, de acordo com essa definição, páginas HTML estáticas não são consideradas aplicações Web.

Aplicações Web se diferenciam de aplicações convencionais² por uma série de características que estão relacionadas com a própria aplicação, seu uso, sua evolução ou seu processo de desenvolvimento (SCHWINGER e KOCH, 2006):

- Integração de diferentes meios para apresentação das informações (textos, gráficos, áudio, vídeo, dentre outros);
- Características ligadas à segurança e privacidade demandam maior atenção dos desenvolvedores;
- Exploração do espaço da informação de forma não-linear;

² O termo software ou aplicação convencional será usado neste trabalho como sinônimo de software ou aplicação não-Web.

- Atenção especial quanto à usabilidade e acessibilidade, pois são normalmente projetadas visando atingir um público bem maior, com expectativas, habilidades e limitações diferentes;
- Necessidade de lidar com uma grande variedade de dispositivos de hardware (redes convencionais, redes sem fio, celulares, PDAs, desktops, dentre outros);
- Necessidade de adequação às questões culturais e lingüísticas, haja vista a possibilidade de acesso global;
- Atenção quanto à escalabilidade, pois as características da aplicação podem tornar difícil a previsão da quantidade real de usuários;
- Constante evolução de tecnologias que apóiam a construção de aplicações Web (novas linguagens, padrões e ferramentas);
- Integração de diversas tecnologias e padrões de software (linguagens de programação, linguagens de scripts, linguagens de marcação, bancos de dados, protocolos, *browsers*, servidores *web*, servidores de aplicação, *e-services*, dentre outros);
- Necessidade de equipes de desenvolvimento multidisciplinares (especialistas em software, programadores visuais, especialistas em infraestrutura, dentre outros);
- Integração com sistemas legados e com serviços disponibilizados fora do escopo da aplicação, e;
- Maior pressão do *time-to-market*, ou seja, ciclos de manutenção curtos em função de pressões de mercado.

A lista acima não significa que aplicações convencionais não tenham preocupações relativas a esses aspectos, mas que eles são normalmente mais significativos no contexto de desenvolvimento de aplicações Web. Tampouco essa lista pretende ser definitiva ou exclusiva. Muitas aplicações convencionais podem demandar um alto grau de atenção em uma ou mais dessas características, assim como outras características não citadas podem se tornar relevantes no contexto específico de uma determinada aplicação.

O aumento da complexidade das aplicações Web, aliada às características próprias dessas aplicações, levou à necessidade do uso de boas práticas no desenvolvimento de aplicações Web para que estas pudessem ser entregues dentro do prazo e orçamento planejados e com alto grau de qualidade e facilidade de manutenção. Esses requisitos motivaram o surgimento da Engenharia de Aplicações Web (*Web Engineering*), com o objetivo de aplicar os princípios da engenharia no

desenvolvimento de aplicações Web de qualidade (PRESSMAN, 2000a), onde a estrutura, funcionalidade, navegação e interação com o usuário sejam representadas de forma adequada (PASTOR, 2004). A Engenharia de Aplicações Web pode ser definida como “o uso de princípios científicos, da engenharia e de gestão e de abordagens sistemáticas com o propósito de desenvolver, implantar e manter de maneira bem-sucedida aplicações Web de alta qualidade” (MENDES *et al.*, 2005).

No contexto da Engenharia de Aplicações Web, vários métodos³ têm sido propostos com o objetivo de apoiar, de forma sistemática, o desenvolvimento de aplicações Web. Uma revisão *quasi*-sistemática (TRAVASSOS *et al.*, 2008) conduzida por CONTE *et al.* (2005) com o objetivo de caracterizar tais métodos possibilitou observar um conjunto de características comuns:

- Os métodos apontam para a necessidade de capturar e representar perspectivas de projeto relevantes no contexto do desenvolvimento de aplicações Web, com destaque para as perspectivas de conceituação (representação de elementos conceituais relacionados ao domínio do problema), apresentação (representação de características relativas à interface com o usuário) e navegação (representação de características relativas à exploração da informação de forma não-linear);
- O paradigma da orientação a objetos é o mais explorado pelos métodos;
- Os métodos têm procurado explorar modelos para representar as perspectivas de projeto, e;
- As preocupações com qualidade ou inexistem ou reaproveitam técnicas exploradas no contexto das aplicações convencionais sem a adaptação dessas técnicas às particularidades das aplicações Web.

Um estudo comparativo realizado anteriormente por ESCALONA e KOCH (2004) com 10 métodos Web, complementada por uma revisão da literatura conduzida no escopo desta pesquisa com outros 14 métodos Web, sobre o tratamento dispensado aos requisitos funcionais nesses métodos, acrescenta mais algumas características a essa lista:

³ A palavra “método” é aqui usada no sentido de “um procedimento organizado com o objetivo de conduzir a um determinado resultado”. Como os autores dos diversos métodos utilizam termos distintos em seus trabalhos, optamos pelo uso deste termo para referenciá-los de forma coletiva

- Pode ser observada uma clara evolução dos métodos no sentido de incorporar atividades relacionadas à engenharia de requisitos, destacando o tratamento diferenciado dos requisitos relevantes no contexto do desenvolvimento das aplicações Web, como navegação e apresentação, em relação aos demais requisitos;
- Para os requisitos que não envolvem navegação ou apresentação, os métodos analisados normalmente propõem o uso de técnicas tradicionais como casos de uso ou linguagem natural sem nenhuma adaptação às particularidades das aplicações Web;
- Não foi possível identificar preocupações em relação à estruturação do documento de requisitos de forma que os diversos modelos produzidos possam ser acessados e analisados de forma integrada;
- Com relação à validação dos requisitos, poucos métodos propõem o uso de técnicas de avaliação e, quando o fazem, se limitam à técnicas tradicionais como revisão/*walk-through* e prototipagem, e;
- Os métodos, de um modo geral, não tem demonstrado preocupações em integrar os modelos de especificação ao processo de garantia da qualidade do produto final.

1.2 Problema

O problema tratado nesta tese está relacionado à especificação dos requisitos funcionais das aplicações Web e seu desdobramento em termos de garantia da qualidade da aplicação.

Ao longo dos anos vários trabalhos têm tentado demonstrar que requisitos inadequados, inconsistentes ou ambíguos têm um impacto negativo na qualidade do software e que a adoção dos processos previstos na Engenharia de Requisitos (SAWYER e KOTONYA, 2004; SOMMERVILLE, 2006) pode reduzir significativamente esses impactos (BOEHM e BASILI, 2001). Nesse contexto, um estudo experimental recente realizado por HAN e HUANG (2007) observou seis dimensões de risco (usuários, requisitos, complexidade do projeto, planejamento e controle, equipe, ambiente organizacional) em 115 projetos de software e concluiu que a dimensão “requisitos” foi a que mais afetou o desempenho geral dos projetos analisados.

Uma das formas de minimizar defeitos em requisitos é a aplicação de técnicas estáticas de avaliação. Dentre essas, as técnicas de inspeção têm se mostrado um importante meio para aprimorar a qualidade do software (FAGAN, 1976; ACKERMAN *et al.*, 1989), pois apóiam a identificação de defeitos tão logo eles tenham sido

introduzidos, reduzindo o retrabalho em fases subseqüentes do desenvolvimento e falhas no produto final (TRAVASSOS *et al.*, 1999).

Por outro lado, o teste de software tem por objetivo identificar defeitos presentes em determinado produto através da sua análise dinâmica, ou seja, através da sua execução. Essa execução tem como meta revelar falhas no produto, permitindo, assim, que as devidas correções tornem o produto mais confiável e garantindo sua qualidade (DIAS NETO e TRAVASSOS, 2006). Mais especificamente sobre o teste funcional, dois fatores que influenciam sobremaneira a qualidade desses testes são: 1) a própria qualidade dos requisitos usados como oráculo para o planejamento dos testes, e; 2) o procedimento adotado para geração dos casos e procedimentos de teste, ou seja, de que forma esses artefatos são gerados e qual os insumos necessário para essa geração.

Além de impactarem diretamente na qualidade do produto, os requisitos funcionais compõem a base para construção dos modelos Web. Logo, a estruturação dos requisitos com o objetivo de apoiar adequadamente a construção dos modelos Web deve ser tratada como um ponto crucial para obtenção de modelos de qualidade.

Podemos depreender, então, que a qualidade do produto final passa pela estruturação adequada e pelo controle de qualidade dos requisitos funcionais, uma vez que estes representam o oráculo a partir do qual os modelos Web e o plano de testes funcionais são derivados.

1.3 Questões de Pesquisa

A partir do problema exposto na seção anterior, a questão de pesquisa para esta tese pode ser formulada como:

É possível aprimorar a qualidade das aplicações Web através da estruturação da especificação dos requisitos funcionais dessas aplicações em torno de um arcabouço que promova a garantia da qualidade da própria especificação e do produto final ?

Contextualizando o aprimoramento da qualidade e a especificação de requisitos no cenário contemporâneo de desenvolvimento de aplicações Web, duas sub-questões podem ser formuladas:

Seria possível:

Q1) apoiar a redução de defeitos nas especificações funcionais de aplicações Web, e;

Q2) apoiar a redução do esforço necessário para planejamento dos casos e procedimentos de testes funcionais

através de uma abordagem para análise e especificação desses requisitos alinhada aos conceitos explorados pelos métodos Web contemporâneos em comparação a uma abordagem *ad-hoc*?

1.4 Objetivos

O objetivo principal deste trabalho consiste em propor uma abordagem para especificação estruturada de requisitos funcionais de aplicações Web, conforme definido na questão de pesquisa formulada na seção 1.3. Esse objetivo pode ser mais bem detalhado nos seguintes sub-objetivos:

- Estruturar um arcabouço para análise, classificação e especificação de requisitos funcionais alinhado aos conceitos explorados pelos métodos Web contemporâneos, oferecendo, assim, uma forma consistente para tratamento dos requisitos desde a fase inicial do projeto;
- Explorar modelos conceituais para representação das entidades relacionadas ao domínio do problema do ponto de vista estrutural e comportamental;
- Explorar modelos de casos de uso para especificação do comportamento esperado do sistema do ponto de vista dos seus usuários;
- Formalizar um arcabouço para especificação de casos de uso composto por um conjunto de definições, critérios e restrições e oferecer, a partir dele, uma estrutura sintática e semântica capaz de apoiar a garantia da qualidade da especificação e do produto final;
- Definir uma infra-estrutura computacional que apóie a especificação e verificação dos casos de uso, e;
- Definir uma infra-estrutura computacional que apóie o planejamento dos casos e procedimentos de testes funcionais a partir da exploração dos modelos de casos de uso.

Essa tese apresenta a abordagem proposta e as ferramentas de apoio criadas para viabilizar o seu uso, bem como discute os resultados obtidos a partir das avaliações e as limitações observadas. A metodologia de trabalho baseada em evidências (SHULL *et al.*, 2001; MAFRA *et al.*, 2006; SPÍNOLA *et al.*, 2008), adotada

nesta pesquisa, apoiou tanto a concepção quanto as avaliações da abordagem proposta através de um estudo secundário, que não foi conduzido no escopo deste trabalho, três estudos de observação e dois estudos de caracterização em busca de indicadores que mostrassem a viabilidade do seu uso.

1.5 Metodologia de Trabalho

A abordagem proposta neste trabalho teve sua construção e avaliação de viabilidade apoiada por experimentação, através da adoção dos conceitos preconizados por metodologias baseadas em evidências para desenvolvimento e introdução de tecnologias de software na indústria (SHULL *et al.*, 2001; MAFRA *et al.*, 2006, SPÍNOLA *et al.*, 2008). A Figura 1-1 apresenta um resumo das atividades realizadas, bem como os capítulos deste trabalho que reportam cada uma dessas atividades.

O arcabouço proposto por MAFRA *et al.* (2006) tem como objetivo principal apoiar o desenvolvimento de novas tecnologias de software, desde a sua concepção até a sua transferência para a indústria. Ele prevê, como primeiro passo, a execução de uma revisão sistemática (KITCHENHAM *et al.*, 2004) com o objetivo de coletar, com maior grau de rigidez e amplitude, o conhecimento existente na literatura acerca de um tema de pesquisa. No contexto desta tese essa revisão foi realizada por CONTE *et al.* (2005), cuja questão principal de pesquisa focava na caracterização dos métodos propostos na literatura técnica para desenvolvimento de aplicações Web.

A introdução de estudos primários logo após a revisão sistemática foi formalizada por SPÍNOLA *et al.* (2008), onde os autores propõem que o conhecimento obtido com a revisão sistemática seja avaliado através de *surveys* com pesquisadores externos da mesma linha de pesquisa. No escopo desta pesquisa esses estudos primários foram conduzidos como três estudos de observação preliminares, com o propósito de adquirir um entendimento mais refinado sobre a exploração de diversos conceitos coletados pela revisão sistemática (CONTE *et al.*, 2005). Assim, o principal objetivo desses estudos foi tentar compreender, de forma mais detalhada, o uso dos modelos propostos pelos métodos Web em um processo de desenvolvimento e coletar dados qualitativos acerca do seu uso. O capítulo 2 apresenta esses três estudos e discute os resultados obtidos.

Ao final dos três estudos de observação preliminares, a análise dos dados obtidos indicou a adequação dos requisitos e o apoio ferramental como dois fatores críticos para o sucesso da exploração dos modelos Web propostos na literatura técnica. Dessa forma, uma revisão da literatura foi conduzida com o objetivo de

caracterizar os métodos Web propostos na literatura técnica com respeito ao tratamento dos requisitos. O capítulo 3 apresenta o protocolo adotado nessa revisão e resume os resultados da análise de 24 métodos de desenvolvimento Web selecionados nessa revisão.

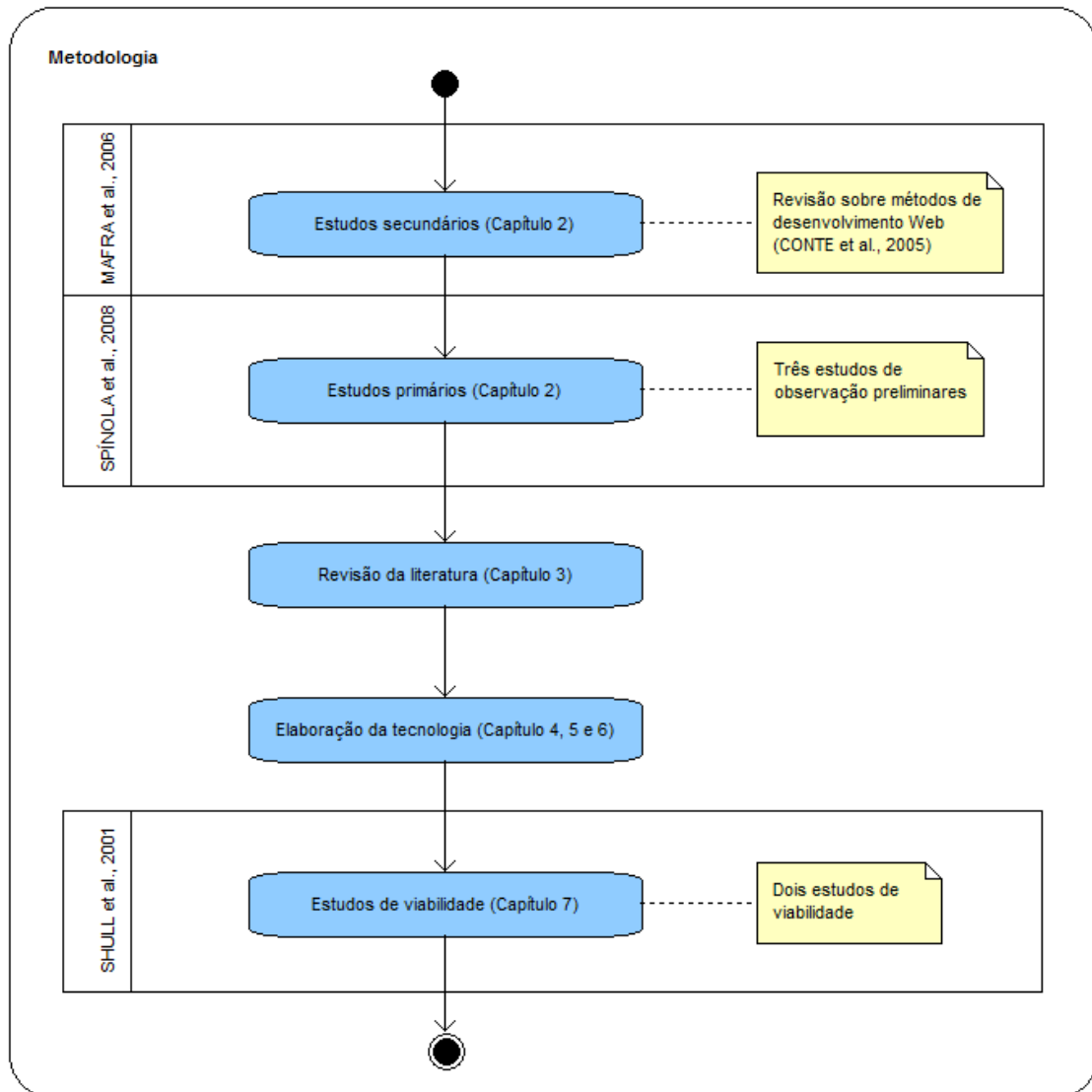


Figura 1-1 - Visão geral da metodologia adotada

Tanto os três estudos de observação preliminares (capítulo 2) quanto a revisão da literatura (capítulo 3) forneceram subsídios para a definição da abordagem apresentada nesta tese, que encontra-se detalhada, juntamente com as ferramentas que a apóia, nos capítulos 4, 5 e 6.

Por último foram realizados dois estudos de viabilidade com o objetivo de caracterizar a abordagem e verificar a viabilidade de seu uso. Esses estudo estão detalhados no capítulo 7.

1.6 Organização

Este capítulo apresentou o contexto e a motivação para realização dessa pesquisa, bem como as questões de pesquisa, os objetivos planejados e a metodologia utilizada na definição e avaliação da abordagem proposta. Estes tópicos serão refinados ao longo dos seguintes capítulos:

- **Capítulo 2 – Estudos Preliminares:** apresenta três estudos preliminares que foram conduzidos com o objetivo de observar a utilização dos conceitos preconizados pelos métodos Web em cenários de desenvolvimento. Os resultados desses estudos forneceram indicações que apoiaram a definição da abordagem proposta nesta tese;
- **Capítulo 3 – Métodos de Desenvolvimento de Aplicações Web:** caracteriza os métodos que tratam requisitos das aplicações Web a partir de uma revisão da literatura técnica e do estudo comparativo realizado por ESCALONA e KOCH, 2004;
- **Capítulo 4 – Abordagem para Especificação de Requisitos:** apresenta detalhadamente a abordagem proposta e seu escopo de atuação, juntamente com o processo e os artefatos previstos para especificação dos requisitos;
- **Capítulo 5 – Metamodelo para Especificação de Casos de Uso:** apresenta o arcabouço criado com o objetivo de apoiar a descrição de casos de uso segundo um conjunto de critérios e restrições bem definido, juntamente com os princípios que nortearam a criação desse arcabouço;
- **Capítulo 6 – Infra-estrutura de apoio à Especificação de Casos de Uso:** apresenta a infra-estrutura computacional organizada para apoiar a abordagem de especificação e garantia da qualidade;
- **Capítulo 7 – Avaliação Experimental:** descreve dois estudos conduzidos no contexto da abordagem, cujos resultados nos trazem indicações de que o uso da abordagem pode auxiliar na redução de defeitos na especificação de requisitos, e;
- **Capítulo 8 – Conclusão e Trabalhos Futuros:** apresenta as conclusões, contribuições da pesquisa, limitações, questões não respondidas e discute algumas perspectivas de trabalhos futuros.

2 Estudos Preliminares

Neste capítulo são apresentados com detalhes três estudos preliminares conduzidos com o objetivo de observar a utilização dos conceitos preconizados pelos métodos Web em cenários de desenvolvimento. Os resultados desses estudos forneceram indicações que apoiaram a definição da abordagem proposta nesta tese.

2.1 Introdução

A análise dos métodos de desenvolvimento Web propostos na literatura técnica apresentados na revisão sistemática conduzida por CONTE *et al.* (2005) nos permitiu ter uma visão crítica de suas principais características. Entretanto, mesmo com as informações obtidas a partir das descrições desses métodos, tornou-se necessário observar na prática algumas idéias e questões iniciais que surgiram dessa análise, mais precisamente com relação a: (1) relevância dos modelos de projeto Web em função do tipo da aplicação; (2) viabilidade da construção dos modelos de projeto Web a partir de um conjunto de requisitos; (3) dificuldades relacionadas à geração desses modelos dentro de um processo de desenvolvimento; (4) tipos de modelos mais adequados para representação da perspectiva de apresentação, e; (5) modelagem da informação no contexto da perspectiva de navegação.

Seguindo a metodologia apresentada na seção 1.5, foram realizados três estudos com o objetivo de observar essas questões, que são apresentados nas seções 2.2, 2.3 e 2.4. A seção 2.5 apresenta uma análise global dos três estudos, tanto do ponto de vista quantitativo quanto qualitativo, enquanto a seção 2.6 apresenta as ameaças à validade dos três estudos. Finalmente, na seção 2.7 é apresentada a conclusão, mostrando como esses estudos foram úteis para a formalização da abordagem de desenvolvimento Web proposta neste trabalho e no apêndice A são apresentados alguns instrumentos utilizados nesses estudos.

2.2 Primeiro Estudo de Observação

Esse estudo ocorreu no contexto da disciplina de graduação Engenharia de Software Orientada a Objetos ministrada no 2º semestre de 2005 na UFRJ.

2.2.1 Objetivo

O objetivo do estudo, segundo o paradigma GQM (*Goal-Question-Metric*) (BASILI, 1988), pode ser formalizado como:

<u>Analisar</u> <u>com o propósito de</u> <u>com respeito à</u> <u>do ponto de vista do</u> <u>no contexto</u>	o uso de modelos no desenvolvimento de aplicações Web, caracterizar, relevância das perspectivas (percepção de importância) e viabilidade dos modelos (percepção de dificuldade), pesquisador, de desenvolvimento de diferentes tipos de aplicações Web por alunos de graduação em Ciência da Computação e Engenharia de Computação e Informação da UFRJ.
--	---

2.2.2 Contexto

O foco do trabalho foi o desenvolvimento de aplicações Web de pequena escala e diferentes tipos, seguindo o paradigma de orientação a objetos. A seleção das aplicações foi baseada em demandas reais observadas por situações do dia-a-dia referentes ao contexto institucional e de pesquisa:

1. **Ferramenta para Configuração e Alocação de serviços em Ambientes de Engenharia de Software Experimental:** a Engenharia de Software tem explorado a construção de ambientes através de integração de ferramentas para gerenciar as informações e apoiar as atividades do processo de desenvolvimento. Com o mesmo intuito a equipe de Engenharia de Software Experimental propôs a definição e construção de um ambiente que pudesse apoiar o Processo de Experimentação em ES. A execução de estudos experimentais em ES visa investigar, dentre outros aspectos, a viabilidade, custo/benefício e emprego de técnicas, tecnologias e metodologia utilizadas no processo de desenvolvimento. O objetivo deste trabalho é desenvolver um dos componentes do Ambiente responsável pela configuração dos Processos de Experimentação a serem executados.
2. **Ferramenta para Construção e Manutenção de Matriz de Rastreabilidade de Requisitos em Projetos de Software:** o propósito desta ferramenta é auxiliar o Engenheiro de Software a gerenciar sua *baseline* de requisitos. Para isso é necessário estabelecer a rastreabilidade (vínculos) entre os requisitos e os outros produtos de trabalho do desenvolvimento (modelos de análise e projeto, código, casos de teste...). A rastreabilidade é importante para auxiliar a avaliar o impacto de

mudanças nos requisitos e também para demonstrar a completa satisfação dos requisitos.

3. **Aplicação para a Gerência de Configuração e Manutenção de Computadores:** esta aplicação Web tem o objetivo de realizar o gerenciamento e o controle das configurações de computadores instalados em laboratórios de pesquisa. A idéia é permitir aos usuários do laboratório identificar as facilidades existentes em cada máquina, otimizando assim sua utilização e evitando retrabalho de instalação. Além disso, é interessante também poder ter um controle de manutenção (hardware e software) o que possibilitará aprimorar o planejamento de substituição e renovação.
4. **Sistema para Controle de Inventário Patrimonial:** gerenciar e controlar o inventário, ou seja os bens de uma determinada organização, representa um conjunto de atividades importantes e relacionadas com a evolução patrimonial da empresa. Desta forma este sistema intenciona permitir que as informações relacionadas aos bens adquiridos por uma empresa possam ser registrados e tenham seu histórico de movimentação também armazenado de forma a permitir que os administradores possam saber, a qualquer instante, qual a posição do inventário (valor) bem como onde e quando foram adquiridos, seu valor venal considerando a depreciação anual, o estado atual dos bens e onde se encontram na companhia.
5. **Conjunto de Componentes e Serviços Web para a Realização de Votação Eletrônica:** este conjunto de componentes deverá apoiar a consulta de opinião de grupos de indivíduos, realizando um controle sobre a unicidade e privacidade do voto. A flexibilização de organização das consultas, bem como a tabulação e exportação dos resultados em formato padronizado permitirá sua utilização em diferentes contextos e domínios de aplicação.

2.2.3 Projeto do estudo

Todos os quinze participantes (alunos de graduação do 8º período do curso de Engenharia da Computação e Informação da UFRJ) já haviam cursado a disciplina de Engenharia de Software anteriormente e, adicionalmente, estavam cursando a disciplina de Engenharia de Software Orientada a Objetos. Nos períodos anteriores, os alunos tiveram oportunidade de participar de projetos em sala de aula envolvendo os conceitos da orientação a objetos, sendo que alguns deles já atuavam na indústria como estagiários ou programadores na área de Tecnologia da Informação. No decorrer da disciplina foi ensinado aos participantes como desenvolver o projeto da

aplicação utilizando as representações propostas para cada uma das perspectivas. A concordância em participar do estudo foi formalizada através de um formulário de consentimento e a caracterização dos participantes foi realizada através de um formulário preenchido pelos próprios participantes, que foi posteriormente analisado por três pesquisadores para classificação final da experiência em desenvolvimento de cada participante (escala ordinal com valores alta, média ou baixa).

Os quinze participantes foram organizados em cinco equipes de três desenvolvedores, de forma que todas as equipes tivessem, aproximadamente, um nível equivalente de experiência. Nesse estudo as equipes deveriam realizar as atividades e entregar os artefatos conforme descrito na Tabela 2-1.

Tabela 2-1 - Atividades e artefatos do 1º estudo de observação

Atividade	Artefatos gerados
Especificação de requisitos	<ul style="list-style-type: none"> • Documento contendo o escopo da aplicação, o glossário de termos e a lista de requisitos funcionais, não-funcionais e de dados.
Especificação de Casos de Uso	<ul style="list-style-type: none"> • Documento contendo a lista de Atores e a descrição de cada Caso de Uso
Modelagem das Perspectivas	<ul style="list-style-type: none"> • Modelo Conceitual • Modelo de Navegação: Mapa de Atores, Diagrama de Contexto Navegacional e Diagrama de Navegação • Modelo de Apresentação: <i>Templates</i> de Apresentação e Diagrama de Apresentação
Projeto detalhado	<ul style="list-style-type: none"> • Modelos de Classes detalhado

Na execução desse estudo foram adotados modelos retirados dos métodos OOWS (PASTOR *et al.*, 2001) e UWE (KOCH & KRAUS 2002) para representar as perspectivas de navegação e apresentação. A seleção desses modelo ocorreu pelos seguintes motivos:

- O Mapa de Atores e o Diagrama de Contexto Navegacional estão, na nossa avaliação, conceitualmente próximos dos Casos de Uso, o que poderia facilitar a construção daqueles modelos a partir destes.
- O *Template* de Apresentação permite a representação da interface de uma forma bastante abstrata, focando apenas no tipo da informação que será veiculada em uma determinada área. Essa forma de representação pode ser útil caso a interface tenha baixa complexidade ou se deseje apenas delimitar essas áreas, deixando o detalhamento dos componentes da interface para fases posteriores.
- O Diagrama de Apresentação do método UWE (KOCH & KRAUS 2002) permite a representação da estrutura da interface em termos dos elementos

conceituais que a compõem. Esse modelo pode ser usado como uma alternativa ao *Template* de Apresentação do método OOWS.

- Disponibilidade de material para treinamento, o que facilitou tanto o entendimento dos pesquisadores quanto dos participantes do estudo
- Possibilidade de usar uma ferramenta UML para construir os diagramas.

Os modelos solicitados durante o processo de desenvolvimento foram:

- Modelo Conceitual: representação dos elementos do domínio e suas associações segundo a abordagem orientada a objetos. Usa o diagrama de classes da UML.
- Mapa de Atores: foi extraído do método OOWS e é utilizado para representar a taxonomia de atores no contexto da aplicação.
- Diagrama de Contexto Navegacional: também foi extraído do OOWS e é utilizado para representar os contextos de navegação acessados direta ou indiretamente por cada um dos atores.
- Diagrama de Navegação: representa o conteúdo de cada contexto de navegação usando os conceitos de nós e links. Um nó representa um conjunto de informações que podem ser originadas de serviços ou visões sobre o modelo conceitual. Os links representam as formas de exploração das informações ou serviços.
- *Templates* de Apresentação: também foram extraídos do OOWS e representam o particionamento da área de apresentação de acordo com o tipo da informação a ser veiculada em cada área.
- Diagrama de Apresentação: foi extraído do método UWE (HENNICKER e KOCH, 2000) e se propõe a representar a estrutura das páginas Web através de um modelo de interface abstrata que mostra como os elementos abstratos da interface humano-computador estão distribuídos pela página.
- Modelo de Classes Detalhado: modelo de classes representando a solução computacional para o problema proposto.

Das cinco aplicações definidas no contexto do estudo, somente a primeira foi alocada por conveniência a uma equipe, devido à complexidade do tema e ao conhecimento prévio do domínio por dois integrantes da equipe (que já haviam trabalhado com este tema em projetos de pesquisa do grupo de Engenharia de Software Experimental da COPPE/UFRJ). As demais aplicações foram sorteadas entre as outras quatro equipes.

2.2.4 Instrumentação

Os próprios pesquisadores, nesse caso 3 indivíduos, desempenharam o papel de *stakeholders* e foram associados a cada um dos cinco projetos, de forma a permitir uma avaliação completa do problema e da solução proposta. A ferramenta Poseidon for UML Community Edition⁴ foi disponibilizada para construção dos modelos.

2.2.5 Execução

Durante a disciplina os participantes elaboraram a especificação de requisitos e realizaram a inspeção dos mesmos usando técnica de inspeção *ah-hoc*. Em seguida as equipes tiveram cerca de três semanas para a elaboração e entrega dos modelos referentes às perspectivas de projeto Web e ao projeto detalhado. Após a entrega final do trabalho, foi distribuído aos participantes um questionário, para ser respondido individualmente, contendo uma avaliação sobre o trabalho realizado. Todas as equipes entregaram todos os artefatos solicitados, com exceção das equipes 2 e 5 que não entregaram o projeto detalhado. A Tabela 2-2 lista algumas medidas diretas extraídas desses artefatos.

Tabela 2-2 - Medidas diretas extraídas dos artefatos de projeto do 1º estudo

Projeto	Qtde. de Requisitos	Qtde. de Casos de Uso	Qtde. de Atores	Qtde. de Classes Conceituais	Qtde. de Nós Navegacionais	Qtde. de Classes de Projeto
1	21	4	2	11	4	21
2	15	19	4	10	17	-
3	9	5	3	11	17	32
4	16	12	3	10	21	13
5	19	9	3	4	16	-

Quanto ao questionário, onze dos quinze participantes responderam às duas questões formuladas:

1. Facilidades/dificuldades provocadas pelo uso das perspectivas: quatro participantes responderam que as perspectivas facilitaram, três responderam que as perspectivas não trouxeram nem dificuldades nem

⁴ www.gentleware.com

facilidades, dois apontaram dificuldades e um considerou as perspectivas dispensáveis.

2. Aplicabilidade do modelo navegacional e do modelo de apresentação: sete consideraram que os modelos puderam ser desenvolvidos e aplicados ao projeto, sendo que um destes destacou a alta complexidade dos diagramas de navegação e outro considerou o mapa de atores sem utilidade em aplicações de pequeno porte. Dois participantes questionaram a aplicabilidade dos diagramas de navegação e apresentação em aplicações de pequeno porte. Um dos participantes questionou a aplicabilidade dos diagramas de navegação e dos *templates* e diagramas de apresentação enquanto elementos de comunicação com o *web designer*.

De acordo com o que foi descrito no item 1, houve uma dispersão de opinião com relação ao benefício do uso das perspectivas de projeto. A baixa complexidade das aplicações pode ter influenciado nessa avaliação, já que, nesse caso, a predisposição do participante em usar ou não as perspectivas pode suplantar a real necessidade de representá-las. Por outro lado, essa dispersão pode ser um indício de que a exploração de todas essas perspectivas só faça sentido caso alguma característica relacionada à aplicação direcione para essa demanda. A última opinião da pergunta 2 revela um comportamento que pode ser recorrente: o participante considerou os modelos dispensáveis porque não enxergou como esses modelos poderiam auxiliar em atividades posteriores. Além disso, a análise dos diagramas de apresentação construídos pelas equipes mostrou que eles foram explorados de forma muito superficial em relação ao detalhamento apresentado nos demais artefatos, como casos de uso e o modelos de classe. As razões para esse desbalanceamento não puderam ser capturadas neste estudo, porém a abstração proporcionada pelos diagramas de apresentação pode ter influenciado nesse ponto.

Um item que não pode ser negligenciado nesse estudo foi o pouco apoio proporcionado pela ferramenta Poseidon for UML for Community Edition à construção de modelos baseados na UML que adotam semânticas adicionais para os elementos dos diagramas através do uso de estereótipos (esse apoio existe, mas está disponível somente nas versões comerciais da ferramenta). O ideal, nesse caso, seria a utilização de uma ferramenta que permitisse o uso de perfis UML, para definição dos estereótipos e *tagged-values* adequados à modelagem navegacional, o que não foi possível nesse estudo devido ao desconhecimento dos pesquisadores acerca de uma ferramenta gratuita com tal funcionalidade.

Apesar dos indícios pontuais, a informalidade e o próprio tamanho da amostra não nos permitiram consolidá-los. Um segundo estudo de observação se fez, então, necessário.

2.3 Segundo Estudo de Observação

Esse estudo de observação ocorreu no contexto da disciplina de graduação Interface Humano-Computador ministrada no 1º semestre de 2007 na UFRJ.

2.3.1 Objetivo

Analisar com o propósito de com respeito à do ponto de vista do no contexto o uso de modelos no desenvolvimento de aplicações Web, caracterizar, viabilidade dos modelos (percepção de dificuldade), pesquisador, do desenvolvimento de um portal Web por alunos de graduação do curso de Engenharia de Computação e Informação da UFRJ.

2.3.2 Contexto

Nesse estudo os participantes focaram em uma única aplicação Web para que fosse possível isolar a percepção de dificuldade do tipo da aplicação e observar, de forma mais isenta, as diferenças entre os artefatos gerados pelas diversas equipes. A escolha da aplicação baseou-se no interesse dos próprios alunos em gerar a especificação e o projeto para o portal do curso de Engenharia da Computação e Informação da UFRJ sendo, para eles, uma aplicação real.

2.3.3 Projeto do Estudo

Os quatorze participantes tiveram a liberdade de se organizarem livremente, desde que formassem quatro equipes de desenvolvimento, sendo duas com quatro participantes e outras duas com três participantes. Além disso, o aparente desinteresse dos participantes do primeiro estudo no desenvolvimento dos diagramas de apresentação fez com que os pesquisadores lançassem mão do uso de protótipos como uma alternativa para a representação das interfaces humano-computador. As atividades executadas durante o desenvolvimento e os artefatos gerados podem ser vistos na Tabela 3-3.

Tabela 2-3 - Atividades e artefatos do 2º estudo de observação

Atividade	Artefatos gerados
Especificação de Casos de Uso	<ul style="list-style-type: none">Documento contendo a lista de requisitos, a lista de atores e a descrição dos casos de uso

Modelagem das Perspectivas	<ul style="list-style-type: none"> • Modelo conceitual • Modelo de Navegação: Mapa de Atores, Diagrama de Contexto Navegacional e Diagrama de Navegação
Criação do protótipo	<ul style="list-style-type: none"> • Protótipos da interface dos Casos de Uso

Neste segundo estudo, diferentemente do primeiro, utilizamos somente os diagramas da perspectiva de navegação previstos no método OOWS (PASTOR et al., 2001). Os diagramas relacionados à perspectiva de apresentação foram totalmente substituídos por protótipos da interface humano-computador devido: (1) às observações feitas pelos participantes do 1º estudo, que levantaram dúvidas quanto à utilidade desses diagramas como um instrumento de comunicação entre os membros da equipe e na criação da interface, e; (2) o aparente desinteresse dos participantes do primeiro estudo no desenvolvimento desses diagramas. Optamos por deixar a cargo de cada equipe a definição de qual ferramenta ou abordagem utilizar na representação desses protótipos (ferramentas de autoria ou apresentação, *storyboards*, *mockups*, páginas HTML ou outro mecanismo).

Os modelos previstos nesse estudo foram:

- Modelo Conceitual: representação dos elementos do domínio e suas associações segundo a abordagem orientada a objetos. Usa o diagrama de classes da UML.
- Mapa de Atores: foi extraído do método OOWS e é utilizado para representar a taxonomia de atores no contexto da aplicação.
- Diagrama de Contexto Navegacional: também foi extraído do OOWS e é utilizado para representar os contextos de navegação acessados direta ou indiretamente por cada um dos atores.
- Diagrama de Navegação: também foi extraído do OOWS e é utilizado para representar o conteúdo de cada contexto de navegação a partir das visões sobre as classes e associações do modelo conceitual.

2.3.4 Instrumentação

Os requisitos do portal foram definidos por dois pesquisadores e disponibilizados aos participantes. Também foi disponibilizada a ferramenta StarUML⁵ juntamente com um perfil UML (OMG, 2010a) para construção do modelo navegacional, segundo o método OOWS. Duas razões motivaram a troca da ferramenta do primeiro para o segundo estudo: (1) a não evolução das funcionalidades

⁵ www.staruml.com

oferecidas pela ferramenta Poseidon for UML Community Edition, que continuava oferecendo a possibilidade de explorar perfis UML somente para as versões comerciais, e; (2) o surgimento da ferramenta StarUML que, à época, oferecia mais flexibilidade para construção dos modelos desejados, principalmente por permitir a definição de perfis UML e a renderização customizada dos elementos dos modelos, facilitando a criação de modelos aderentes, semântica e visualmente, ao método OOWS. Como o objetivo do estudo não estava associado à avaliação da ferramenta, a estratégia adotada para seleção desta baseou-se na sua adequação à construção dos modelos especificados no projeto do estudo.

2.3.5 Execução

As equipes tiveram cerca de três semanas para a elaboração e entrega dos artefatos citados na Tabela 2-4. Ao final desse prazo cada equipe fez uma apresentação de aproximadamente 30 minutos na qual explicaram os modelos gerados e as decisões de projeto adotadas. Nessa apresentação, cada equipe também teve que descrever quais dificuldades foram encontradas durante o processo de desenvolvimento. O resumo das dificuldades relatadas está listado na Tabela 4-4.

Tabela 2-4 - Dificuldades relatadas pelos participantes do 2º estudo

Equipe	Desenv.	Dificuldades
Diamante	4	Definição do Modelo navegacional, falta de uma visão geral da aplicação, confusão entre modelo OO e modelo ER, trabalho distribuído e diagramação dos modelos
Zafira	3	Divisão do trabalho entre os membros causou confusão, modelos navegacionais e controle da versão dos modelos devido ao trabalho distribuído
Rubi	4	Desenvolvimento separado dos Casos de uso pelos membros levou a dificuldades na hora de unir os modelos, modelo conceitual não foi bem compreendido, falta de exemplos de modelos prontos e falta de reuniões presenciais
Topázio	3	Confusão entre modelo conceitual e modelo ER e dificuldade para definir as classes

A partir dos artefatos entregues foi possível extrair algumas medidas diretas apresentadas na Tabela 2-5.

Tabela 2-5 - Dados quantitativos do 2º estudo

Equipe	No. de participantes	No. de UCs	No. de classes conceituais	No. atores	No. de contextos navegacionais	No. de UCs prototipados
Diamante	4	30	14	6	33	6
Zafira	3	36	15	8	21	3
Rubi	4	27	12	5	18	19

Topázio	3	33	13	5	77	15
---------	---	----	----	---	----	----

Com relação ao uso dos protótipos da interface humano-computador, pudemos observar que a sua criação durante o desenvolvimento foi mais bem assimilada pelas equipes do que os diagramas de apresentação usados no primeiro estudo de observação (seção 3.1). Isso já era esperado, pois os protótipos materializam a estrutura da interface com os usuários de uma forma concreta e mais natural para os desenvolvedores. Contudo, ainda persistiram dúvidas quanto à forma de representar os aspectos comportamentais relacionados à perspectiva de apresentação.

Algumas dificuldades relatadas pelos participantes do primeiro estudo surgiram também neste segundo estudo, mais notadamente as questões relacionadas ao modelo navegacional (relatado por 2 das 4 equipes), enquanto outras dificuldades vieram à tona somente nesse estudo, como a divisão dos modelos entre os membros da equipe (relatado por 3 das 4 equipes). Entretanto, apesar dessa repetição, não foi possível estabelecer com maior precisão nenhuma sugestão de possível causa e efeito. Assim, um terceiro estudo de observação foi elaborado no sentido de tentar capturar os dados de forma mais rigorosa e objetiva a fim de possibilitar essa análise.

2.4 Terceiro Estudo de Observação

Esse estudo de observação ocorreu no contexto da disciplina de graduação Engenharia de Software Orientada a Objetos ministrada no 2º semestre de 2008 na UFRJ.

2.4.1 Objetivo

Analisar com o propósito de com respeito à do ponto de vista do no contexto o uso de modelos no desenvolvimento de aplicações Web, caracterizar, viabilidade e relevância dos modelos (percepção de esforço, dificuldade e importância), pesquisador, do desenvolvimento de uma aplicação Web para Controle de Inventário Patrimonial por alunos de graduação do curso de Engenharia de Computação e Informação da UFRJ.

2.4.2 Contexto

Nesse estudo os participantes focaram novamente em uma única aplicação Web. A escolha da aplicação baseou-se, principalmente, no interesse dos pesquisadores em aumentar o grau de complexidade da aplicação a ser modelada. Assim, a especificação da aplicação de Controle de Inventário Patrimonial (usada no primeiro estudo) foi revista e ampliada para que novos casos de uso fossem

incorporados. A nova versão da especificação passou a ser composta por 17 casos de uso.

2.4.3 Projeto do Estudo

A divisão das equipes seguiu dois critérios: a experiência dos participantes e a assiduidade nas aulas, de forma a manter os alunos mais assíduos em um mesmo grupo. O critério da assiduidade foi adotado para que os alunos que tiveram menor participação nas aulas teóricas não influenciassem negativamente no desempenho daqueles que participaram de todas as discussões envolvendo a construção dos modelos. Foram definidas dez equipes com três participantes e uma equipe com quatro participantes. As equipes tiveram a liberdade de se organizarem internamente e não foi exigido que nenhum processo de desenvolvimento específico fosse seguido. Foi sugerido que as equipes utilizassem os diagramas de classes, seqüência, atividades e estado, o mapa de atores e o mapa navegacional. Entretanto, cada equipe poderia abdicar do uso de qualquer um desses modelos, se assim desejasse. Ao final, cada equipe deveria entregar um documento de projeto com os modelos construídos, além de realizar uma apresentação de 20 minutos ressaltando os principais pontos do projeto. É importante ressaltar que a perspectiva de apresentação foi excluída desse estudo porque os seus modelos se demonstraram os menos aderentes aos padrões de modelagem aplicados neste trabalho, particularmente no que diz respeito ao uso da UML. Novas investigações são necessárias no sentido de aprimorar a percepção de como os aspectos estruturais e comportamentais dessa perspectiva podem ser representados em um contexto de desenvolvimento dirigido por modelos.

2.4.4 Instrumentação

Foram entregues aos participantes a especificação da aplicação e um *template* para preparação da apresentação da equipe, direcionando a apresentação para as questões de interesse dos pesquisadores. Também foi disponibilizada a ferramenta BOUML (PAGÉS, 2011) juntamente com perfis UML (OMG, 2010a) para construção do modelo navegacional, segundo o método OOWS (PASTOR *et al.*, 2001). Novamente houve mudança da ferramenta em relação o estudo anterior. Nesse caso, a mudança foi motivada pela descontinuação da ferramenta StarUML, que naquele momento não implementava inovações importantes da especificação UML 2.0, principalmente em relação aos diagramas de estado e atividades.

2.4.5 Execução

As equipes tiveram quatro semanas para realizar a modelagem da aplicação proposta e, após a entrega dos trabalhos, os participantes foram convidados a responderem, individualmente, um questionário sobre as suas percepções acerca dos modelos utilizados.

O questionário final foi respondido por 23 dos 34 participantes e continha quatro questões quantitativas e uma qualitativa sobre o uso dos modelos. Para que as respostas do questionário pudessem ser analisadas de forma isenta, a primeira pergunta procurou definir o nível de participação que cada integrante da equipe teve em relação à elaboração dos modelos. A Tabela 2-6 sumariza esse nível de participação.

Tabela 2-6 - Atuação dos desenvolvedores por modelo

Tipo de participação	Classe	Seqüência	Estado	Atividade	Ator	Navegação
Não construiu nem revisou	0,0%	0,0%	13,1%	34,8%	26,1%	17,4%
Só revisou	8,7%	17,4%	21,7%	26,1%	34,8%	39,1%
Só construiu	8,7%	17,4%	8,7%	0,0%	8,7%	13,1%
Construiu e revisou	82,6%	65,2%	56,5%	39,1%	30,4%	30,4%

Para as demais questões quantitativas, as opiniões dos participantes que não tiveram nenhum contato com o modelo foram expurgadas das estatísticas, ou seja, os indivíduos que nem construíram e nem revisaram o modelo tiveram a sua opinião removida da estatística.

A segunda questão procurou capturar o esforço de construção de cada modelo na percepção dos participantes. As repostas possíveis, nesse caso, eram Alto, Médio ou Baixo. A Tabela 2-7 resume a percepção de esforço dos participantes, além de destacar essa percepção para as três equipes que apresentaram os melhores projetos e para as três equipes que apresentaram os piores projetos. É possível notar que, independente da qualidade do projeto, o esforço percebido pelas equipes segue um comportamento semelhante.

Tabela 2-7 - Percepção de esforço

Diagrama	Esforço Percebido	Equipes com melhores projetos	Equipes com piores projetos	Total	Total de opiniões
Classe	Alto	8	7	15	23
	Médio	4	3	7	
	Baixo	0	1	1	
Seqüência	Alto	11	6	17	23
	Médio	1	5	6	
	Baixo	0	0	0	

Estado	Alto	0	0	0	21
	Médio	3	2	5	
	Baixo	9	7	16	
Atividade	Alto	0	1	1	17
	Médio	10	4	14	
	Baixo	1	1	2	
Ator	Alto	0	0	0	19
	Médio	1	0	1	
	Baixo	11	7	18	
Navegação	Alto	9	8	17	22
	Médio	3	1	4	
	Baixo	0	1	1	

Levando em consideração somente a opinião dos participantes que criaram os artefatos, a Tabela 2-8 apresenta o percentual de distribuição do esforço percebido por diagrama.

Tabela 2-8 - Distribuição de percepção de esforço dos participantes que criaram os artefatos

Esforço percebido	Classe	Seqüência	Estado	Atividade	Ator	Navegação
Alto	61,9%	73,7%	0,0%	11,1%	0,0%	90,0%
Médio	33,3%	26,3%	26,7%	66,7%	11,1%	0,0%
Baixo	4,8%	0,0%	73,3%	22,2%	88,9%	10,0%
Total	100%	100%	100%	100%	100%	100%

A terceira questão teve como foco a importância de cada modelo na percepção dos participantes do estudo, onde as respostas possíveis variavam de 0 (nenhuma importância) a 6 (muito importante). Visando realizar uma avaliação mais justa, a resposta de cada participante foi multiplicada por um peso que indicava como o participante manipulou o artefato:

- Não criou nem revisou: peso 0;
- Só revisou: peso 1;
- Só criou: peso 2, e;
- Criou e revisou: peso 3.

A Tabela 2-9 resume a distribuição de importância percebida pelos participantes por diagrama, destacando para cada diagrama as importâncias que obtiveram votação mais expressiva.

Tabela 2-9 - Distribuição de importância percebida pelos participantes por diagrama

Importância Percebida	Classe	Seqüência	Estado	Atividade	Ator	Navegação
0 – sem importância	0,0%	0,0%	1,4%	0,9%	0,6%	0,0%
1	0,0%	0,0%	12,4%	2,6%	66,9%	9,2%
2	0,0%	0,0%	11,1%	6,8%	9,6%	51,5%
3	0,0%	5,0%	55,3%	10,3%	9,6%	0,6%

4	0,0%	10,6%	18,4%	61,5%	0,0%	30,7%
5	0,2%	68,8%	0,0%	17,9%	13,4%	6,1%
6 – muito importante	99,8%	15,6%	1,4%	0,0%	0,0%	1,8%
Total	100%	100%	100%	100%	100%	100%

Nessa questão os participantes puderam apontar até três fatores de uma lista pré-definida de nove fatores, a saber:

1. Treinamento (exemplos, qualidade do material disponibilizado entre outros);
2. Complexidade do sistema (dificuldade de entendimento do domínio);
3. Documento de especificação (documentação confusa e/ou inadequada);
4. Falta de experiência na criação do modelo;
5. Ferramenta case (componentes inadequados para construção do modelo);
6. Complexidade do diagrama;
7. Ordem com que os modelos foram construídos;
8. Divisão de tarefas/responsabilidades entre a equipe, e;
9. Perspectiva de projeto inconsistente usada pela equipe.

A criação dessa lista de fatores se baseou nas análises feitas nos questionários dos primeiro e segundo estudos de observação anteriormente descritos. A Tabela 2-10 sumariza os três fatores que trouxeram mais dificuldades para cada modelo na percepção dos participantes do estudo.

Tabela 2-10 - Fatores de dificuldade apontados pelos participantes

Dificuldade	Classe	Seqüência	Estado	Atividade	Ator	Navegação
Falta de experiência na criação do modelo	✓	✓	✓	✓		✓
Ferramenta CASE		✓			✓	✓
Divisão de tarefas	✓			✓	✓	
Treinamento		✓			✓	✓
Documento de especificação			✓	✓		
Complexidade do diagrama	✓					
Ordem de construção dos modelos			✓			

A quinta e última questão foi concebida com o intuito de capturar o que poderia ser feito, na opinião dos participantes, para diminuir o esforço ou eventuais dificuldades no uso desses modelos. A pergunta formulada foi:

Na sua opinião de engenheiro de software, o que poderia ser feito para diminuir o esforço/dificuldade com estes modelos?

Foi solicitada uma resposta separada para cada um dos modelos envolvidos no estudo. Por se tratar de uma questão aberta as respostas dos participantes foram analisadas de acordo com as codificações aberta e axial propostas por STRAUSS e CORBIN (1998) para o método *Grounded Theory*. Como a pergunta formulada era bastante específica, o processo de codificação/categorização procurou interpretar as respostas em busca dos fatores facilitadores propostos pelos participantes. As respostas dos participantes, bem como análise dessas respostas dados e os códigos extraídos estão detalhados no apêndice B. Um resumo dos códigos e categorias obtidos nessa análise será apresentado na próxima seção.

A próxima seção apresenta uma análise sobre o uso de modelos em um cenário de desenvolvimento Web à luz dos resultados obtidos nos três estudos de observação realizados.

2.5 Análise dos Resultados dos Estudos

Os dados obtidos com os estudos nos deixam estabelecer algumas indicações sem permitir, contudo, nenhuma afirmação ou conclusão acerca do uso dos modelos no contexto de desenvolvimento Web. Para facilitar essa análise vamos tratar os dados quantitativos para cada um dos modelos separadamente e, em seguida, analisar os dados qualitativos de uma forma geral. É importante destacar que a análise a seguir foi influenciada, na maior parte dos casos, pelos dados do terceiro estudo de observação. Entretanto, os dados obtidos no primeiro e segundo estudos reforçam, em vários momentos, os resultados obtidos no terceiro, se observados isoladamente.

2.5.1 Análise dos Resultados Quantitativos

Diagrama de Classes

Esse modelo foi o mais manipulado pelas diversas equipes nos três estudos, o que já era de certa forma esperado, haja vista o seu papel de destaque no desenvolvimento de software OO. Os participantes puderam perceber a grande importância de construir esse modelo (Tabela 2-9), mas relataram uma percepção de esforço alta (com viés para média) talvez influenciada pela falta de experiência dos participantes, já que a aplicação não tinha alta complexidade. Essa dificuldade também pode estar relacionada com o entendimento do paradigma OO, pois alguns participantes dos estudos 2 e 3 relataram certa “confusão” entre os paradigmas OO e os modelos ER (Entidades-Relacionamentos). A divisão das tarefas também foi

apontada como um fator que trouxe dificuldades no uso desse modelo, já que ela levou à divisão da especificação em casos de uso que foram tratados individualmente pelos desenvolvedores que, por sua vez, perderam a visão geral do domínio necessária para construção e evolução do modelo. O reflexo disso foi que muitos participantes apontaram as reuniões presenciais como um instrumento para mitigar esse risco. Outro ponto levantado foi a dificuldade de criar o modelo conceitual de classes, pois a visão dos casos de uso não proporcionava uma visão geral dos processos de negócio. Daí surgiram algumas sugestões no sentido aprimorar o documento de especificação a fim de incorporar essa informação.

Diagrama de Seqüência

Esse modelo também foi bastante explorado pelos desenvolvedores, porém o grau de importância atribuído pelos participantes foi bem menor se comparado com o modelo de classes (Tabela 2-9). Esse fato surpreende até certo ponto, pois essas duas perspectivas são totalmente complementares e fazem parte da essência do paradigma OO. Esse fato pode ser explicado pela falta de experiência e/ou treinamento/estudo insuficiente (Tabela 2-10), o que pode fazer com que os desenvolvedores tendam a tratar as questões comportamentais prioritariamente através da codificação usando uma linguagem de programação. Seguindo esse raciocínio, a modelagem da estrutura seria importante e essencial como ponto de partida apenas para o processo de codificação. Por outro lado, essa tendência também pode ser alimentada pelo alto grau de esforço percebido na construção desse modelo (Tabela 2-8), o que poderia levar os desenvolvedores a abandoná-lo em função de outra atividade que atinge o mesmo objetivo, mas cuja percepção de esforço é menor.

Diagrama de Atividades

Analisando os trabalhos entregues pelas equipes, verificamos que ora esse diagrama era citado como essencial ora como descartável. Talvez pela falta de um foco bem definido, como ocorre com os diagramas de classe e seqüência, a exploração desse diagrama tenha sido pequena e as percepções tenham sido tão dispersas. Analisando o uso desse diagrama pelas quatro equipes mais bem colocadas, constatamos que ele foi usado como representação gráfica dos casos de uso por duas equipes e como representação gráfica dos processos de negócio por uma equipe (uma das equipes abdicou do seu uso). Entretanto, nenhuma delas declarou que o diagrama era essencial ou ao menos importante. Esse fato nos remete a uma questão mais geral: a percepção da importância pode estar diretamente

relacionada à definição de um foco claro e objetivo para o modelo, a partir do qual o desenvolvedor consegue perceber os benefícios e os desdobramentos do seu uso.

Diagrama de Estado

Esse diagrama foi razoavelmente explorado pelas equipes e o seu grau de importância teve um viés de baixa (Tabela 2-9), assim como o esforço percebido pelos participantes (Tabela 2-8). Apesar do domínio da aplicação desenvolvida realmente não exigir máquinas de estado muito aprimoradas, constatamos, através da análise dos diagramas gerados pelas equipes, que o grau de detalhamento desses diagramas não acompanhou o grau de detalhamento do modelo de classes e seqüência, causando um desbalanceamento entre esses modelos em termos de abstração. Isso pode ter sido causado pela falta de experiência dos desenvolvedores e/ou treinamento inadequado (o mesmo ocorreu com os diagramas de seqüência, já que ambos focam no aspecto comportamental). Entretanto, uma dificuldade relatada chamou a atenção: o documento de requisitos. Através da análise dos dados qualitativos foi possível entender a razão dessa dificuldade: alguns participantes relataram que as transições de estado definidas na especificação estavam nas “entrelinhas” das descrições dos casos de uso e sugeriram que tanto a definição dos estados quanto os eventos que disparam a sua transição sejam representados de uma forma mais objetiva para facilitar a construção inicial deste modelo.

Mapa Navegacional

A Tabela 2-6 indica que um número razoável de participantes tentou explorar esse modelo. Contudo, a percepção de esforço alta (Tabela 2-8) aliada ao baixo grau de importância percebido (Tabela 2-9), fez com que as equipes classificassem esse modelo apenas como desejável ou até mesmo descartável. A falta de experiência e/ou treinamento adequadas aliadas à baixa aderência da ferramenta CASE à semântica do mapa navegacional, de fato podem ser fatores críticos que explicam essas dificuldades. Porém, o que realmente chama a atenção nesse caso é que os participantes não conseguiram perceber a importância de representar essa perspectiva no desenvolvimento de uma aplicação Web. Em última análise, isso poderia indicar que os participantes não sentiram a necessidade de representar a perspectiva navegacional, ou que as informações relevantes foram ou poderiam ser capturadas em outros modelos. Analisando por outro ângulo, a importância pode não ter sido percebida porque os desdobramentos que a criação desse modelo poderia ter no produto final, em termos de garantia da qualidade ou até mesmo da confecção do produto em si, não foram percebidos pelos participantes. Talvez esse seja um

mecanismo natural de corte implicitamente usado pelos desenvolvedores: não vale a pena criar um modelo quando este não é usado de forma direta ou indireta na obtenção do produto final.

Mapa de Atores

A análise dos trabalhos nos permite afirmar que esse diagrama foi efetivamente pouco explorado no contexto de desenvolvimento proposto. Na realidade o mapa de atores só tem sentido se explorado juntamente com o mapa navegacional, já que ambos fazem parte do mesmo método (OOWS). Contudo, as equipes tentaram explorar esse diagrama de forma isolada, o que resultou em um baixo nível de importância percebida.

2.5.2 Análise dos Dados Qualitativos

Os dados qualitativos foram analisados de forma conjunta, sem levar em consideração opiniões específicas acerca de um modelo em particular. A idéia por trás dessa análise foi poder obter um conjunto de categorias e códigos que indicassem facilitadores em um processo de desenvolvimento de aplicações Web dirigido por modelos, que poderiam indicar, em última análise, características a serem observadas e/ou adotadas nesses processos.

A seguir são listadas as categorias, obtidas com a aplicação da codificação aberta e da codificação axial sugeridas por STRAUSS e CORBIN (1998) para o método *Grounded Theory*, e as principais sugestões relacionadas a elas.

Requisitos

Foram citadas sugestões relacionadas à adequação do documento de requisitos como insumo principal para geração inicial dos modelos. Essas sugestões dizem respeito tanto à necessidade de aprimoramento no nível de detalhamento necessário quanto à estrutura do próprio documento. Também foram citadas a necessidade de apresentar inicialmente uma visão geral dos processos de negócio, assim como o escopo do projeto e as perspectivas de futuras evoluções.

Ferramenta CASE

Foram citadas recorrentemente três sugestões: adequação sintática e semântica aos modelos, interface que privilegie a agilidade na manipulação dos modelos e trabalho colaborativo.

Garantia da Qualidade

Foram citadas a necessidade de processos para verificação da consistência entre os modelos, processos para inspeção dos requisitos e processos para validação dos modelos de navegação junto aos clientes.

Modelos

Foram sugeridas a exploração de transformações (semi) automáticas de modelos, a geração de outros artefatos a partir dos modelos e o reuso de modelos e padrões de projeto.

Processo de Modelagem

A maior parte das sugestões se enquadraram nessa categoria, tais como: definição de heurísticas de apoio à construção dos modelos, critérios para seleção dos modelos mais relevantes de acordo com o contexto de desenvolvimento, critérios para definição do nível de detalhamento a ser utilizado nos modelos de acordo com a complexidade existente em determinada parte da aplicação e a definição de uma ordem para construção/refinamento dos modelos.

Treinamento em Modelagem

As diversas sugestões podem ser consolidadas como exploração de exercícios e exemplos com ênfase na simulação de situações reais de projeto.

2.6 Ameaças à Validade dos Estudos

Foram consideradas cinco ameaças à validade recorrentes nos três estudos preliminares realizados:

- **Treinamento das equipes:** a mitigação desse risco envolveu a apresentação e discussão, durante as disciplinas ministradas, sobre todos os artefatos explorados nos estudos, embora a ausência de alguns alunos nessas apresentações possa ter desequilibrado o nível de conhecimento entre eles. Por outro lado, como as tarefas eram sempre realizadas em grupo, esse desequilíbrio pode ser minimizado;
- **Definição das aplicações Web:** embora a seleção das aplicações Web a serem desenvolvidas tenha sido realizada pelos pesquisadores (estudos 1 e 3), a definição dessas aplicações e suas respectivas funcionalidades surgiram de demandas reais observadas pelos pesquisadores no dia-a-dia;

- **Alocação das aplicações Web pelas equipes (somente 1º estudo):** embora a alocação de uma das aplicações do primeiro estudo tenha sido direcionada para uma equipe específica, é importante ressaltar que a modelagem dessa aplicação demandava conhecimento específico sobre Engenharia de Software Experimental, para o qual somente uma das equipes estava apta. É importante ressaltar também que, apesar de conhecer o domínio, esta equipe ainda não havia modelado essa aplicação utilizando os conceitos solicitados no estudo;
- **Mudança das ferramentas:** o uso de três ferramentas diferentes nos três estudos relatados pode ter influenciado nos resultados devido à eventuais facilidades ou dificuldades proporcionadas por essas ferramentas. Por outro lado, as três ferramentas utilizadas seguem o padrão de interface normalmente adotado nesse tipo de aplicação, como barra de ferramentas apresentada de acordo com o tipo do diagrama, janelas *pop-up* para edição das propriedades dos elementos, *drag-and-drop* de elementos gráficos, dentre outros. Adicionalmente, nos três estudos conduzidos os participantes tiveram tempo considerado suficiente para mitigar questões relacionadas ao aprendizado dessas ferramentas, minimizando esse efeito no resultado final;
- **Medição da experiência dos participantes:** a classificação da experiência foi feita, inicialmente, pelos próprios participantes utilizando um questionário que continha perguntas gerais sobre experiência em desenvolvimento tanto no meio acadêmico quanto industrial e perguntas mais específicas sobre modelagem. Entretanto, os pesquisadores se reuniram posteriormente e analisaram essas classificações levando em conta suas observações acerca desses indivíduos durante a disciplina (assiduidade, participação em sala de aula, trabalhos realizados no contexto da disciplina ao longo do semestre, dentre outros);
- **Definição dos grupos:** a definição dos grupos no primeiro e terceiro estudo procurou criar equipes da forma mais equilibrada possível levando em consideração os diversos critérios adotados na classificação dos participantes. Na definição dos grupos foi tomado um cuidado específico para que não houvesse, em um mesmo grupo, participantes que demonstraram assiduidade ou interesse pela matéria muito dissonantes (em detrimento da própria experiência do participante). No segundo estudo,

havia um interesse muito grande dos participantes na definição dos requisitos da aplicação Web, que aliado ao equilíbrio entre as experiências dos participantes, permitiu que a organização dos grupos pudesse ser feita pelos próprios participantes;

- **Participação de estudantes:** estudantes não representam de forma adequada os desenvolvedores do mundo real, assim como o ambiente acadêmico não simula de forma adequada o ambiente industrial. Entretanto, a aplicação usada no terceiro estudo apresenta características bastante semelhantes às aplicações de pequeno a médio porte encontradas na indústria. Adicionalmente, CARVER *et al.* (2003) apresentam uma série de benefícios que podem ser obtidos com a participação de estudantes em estudos experimentais, e;
- **Uso das ferramentas:** a falta de familiaridade com as ferramentas usadas durante os estudos podem trazer algum tipo de dificuldade na construção dos modelos solicitados. Entretanto, como as tarefas realizadas não eram pontuais nem individuais e não requeriam medição de tempo ou esforço, as eventuais dificuldades podem ser desconsideradas.

2.7 Conclusão

No que diz respeito às atividades de modelagem em projetos de software de uma forma geral, as dificuldades apontadas nos estudos reforçam a idéia de que a modelagem de software não é uma atividade trivial. Apesar dos participantes dos estudos serem alunos de graduação, que por sua vez leva ao argumento da inexperiência dos desenvolvedores, é fato que as aplicações desenvolvidas não tinham alta complexidade. Esse cenário indica que, no contexto do desenvolvimento de aplicações Web, que tem procurado explorar intensamente o uso de modelos, deve haver uma preocupação concreta com relação ao apoio ferramental e ao treinamento dos desenvolvedores na abordagem a ser utilizada. A mudança de paradigma e de foco proposta pela maioria dos métodos de desenvolvimento Web (orientação a modelos) traz mudanças significativas ao estado-da-prática, tradicionalmente focado em questões tecnológicas (STANDING, 2002). Esse argumento é reforçado se levarmos em consideração que a formação tradicional dos desenvolvedores de software prioriza mais as fases relacionadas à manufatura (*coding*) do que ao projeto (*design*) ou ao tratamento dos requisitos. Mais especificamente com relação ao tratamento dos requisitos, os resultados dos estudos forneceram indícios da

necessidade de estruturação adequada desses documentos com o objetivo de apoiar de forma efetiva a elaboração dos modelos de projeto.

Em paralelo a esses estudos, o grupo de Engenharia de Software Experimental da COPPE/UFRJ tem estado envolvido com o desenvolvimento de um sistema de informação Web larga-escala denominado SiGIC. Esse sistema, apesar de aparentemente representar um sistema tradicional orientado a dados, é responsável pela integração de todas as áreas de negócio da organização, tanto do ponto de vista gerencial quanto operacional. A criticidade e a abrangência dessa atuação lhe conferem uma complexidade semelhante aos sistemas de gestão corporativa ou ERP (*Enterprise Resource Planning*) (LOZINSKY, 1998 *apud* PRESSMAN, 2000b). A automação de processos de negócio, muitas vezes envolvendo workflows entre áreas da organização ou entre a organização e entidades externas, segurança e auditoria nas operações realizadas, integração com sistemas legados, entre outros, são alguns dos requisitos demandados no contexto desse sistema. O cenário de desenvolvimento do SiGIC possibilitou, com o apoio da experimentação, a exploração de algumas idéias e temas de pesquisa relacionados à Engenharia de Aplicações Web, como:

- Aplicação de técnicas de inspeção em módulos do SiGIC com características distintas, de onde foi possível observar tendências sobre a origem dos defeitos identificados;
- Adaptação do processo de engenharia de requisitos adotado no projeto, com relação às atividades, técnicas e artefatos gerados, em virtude da análise do impacto desse processo na qualidade dos requisitos;
- Configuração (desenvolvimento e/ou adaptação) de técnicas de inspeção e teste aplicadas para garantia da qualidade de aplicações Web, e;
- Identificação de indicadores (métricas) que permitam avaliar, antecipadamente, tendências relativas à qualidade dos produtos sendo desenvolvidos.

Nesse contexto, surgiu a necessidade de mapear como os requisitos têm sido tratados pelos métodos de desenvolvimento Web contemporâneos e o apoio ferramental proposto para esse tratamento. Para esse mapeamento foi realizada uma revisão da literatura, que será apresentada com detalhes no capítulo 3.

3 Métodos de Desenvolvimento de Aplicações Web

Neste capítulo é apresentada uma revisão da literatura técnica para caracterização de métodos de desenvolvimento que tratam os requisitos de aplicações Web. Foram analisados 24 métodos, ao longo dos últimos anos, e os resultados dessa análise auxiliaram no entendimento do estado-da-arte em termos de tratamento de requisitos por parte dos métodos Web e forneceram indicações para a concepção da abordagem de especificação proposta nesta tese.

3.1 Introdução

Com o objetivo de identificar trabalhos relevantes com relação ao tratamento de requisitos de aplicações Web foi realizado um levantamento sobre quais métodos de desenvolvimento que tratam requisitos de aplicações Web têm sido propostos na literatura técnica.

Revisões sistemáticas da literatura (KITCHENHAM, 2004) baseiam-se em uma estratégia de pesquisa bem definida, com o objetivo de selecionar o máximo de material bibliográfico relevante sobre um tema. O protocolo da revisão sistemática é um documento que especifica a questão central da pesquisa e os métodos que serão utilizados para executar a revisão. Ele deve documentar a estratégia de busca e definir, de forma explícita, os critérios de inclusão e exclusão para avaliar cada estudo primário potencial. Assim, o protocolo permite que leitores (e outros pesquisadores) possam conhecer seu grau de rigor e a completeza da revisão (BIOLCHINI *et al.*, 2007). Uma revisão sistemática propõe uma avaliação mais justa do tópico de pesquisa à medida que utiliza uma metodologia de revisão rigorosa, confiável e passível de auditagem (KITCHENHAM, 2004).

A revisão realizada no contexto desta tese apoiou-se em uma série de características das revisões sistemáticas da literatura, como a definição das questões de pesquisa, de critérios para seleção dos estudos e da estratégia para extração de dados. Por outro lado, algumas características que fortalecem a formalização e o rigor científico das revisões sistemáticas não foram observadas na sua totalidade:

- O protocolo da revisão, com os termos selecionados para busca, critérios de inclusão/exclusão, dentre outros, não foi avaliado por outros pesquisadores;
- O resultado da seleção preliminar dos estudos primários não foi revisado por nenhum pesquisador, e;
- A busca pelos estudos foi realizada em somente uma fonte de pesquisa, embora esta fonte indexe várias outras fontes relevantes.

Adicionalmente, ESCALONA e KOCH (2004) publicaram um estudo comparativo sobre 10 métodos de desenvolvimento Web destacando como estes métodos tratavam os requisitos das aplicações naquela época. O trabalho não deixa explícito os critérios utilizados na seleção desses 10 métodos, mas 9 deles podem ser considerados métodos bem divulgados em eventos e publicações da área de Engenharia de Aplicações Web. Devido à sobreposição de interesses entre a revisão pretendida nessa pesquisa e o estudo comparativo de ESCALONA e KOCH (2004), optou-se por realizar a revisão da literatura a partir do ano de 2004 (inclusive) e acrescentar à lista dos métodos Web a serem analisados, os 10 métodos estudados por ESCALONA e KOCH (2004). O objetivo desse arranjo é obter uma visão mais atualizada sobre o tratamento dos requisitos nos métodos Web e, ao mesmo tempo, complementar essa análise com a avaliação de métodos anteriores ao ano de 2004 e que se demonstravam mais relevantes àquela época, segundo a visão de ESCALONA e KOCH (2004). Nesse cenário, vale ressaltar que aqueles métodos analisados por ESCALONA e KOCH (2004) que continuaram evoluindo ao longo dos últimos 7 anos poderiam ser selecionados nessa revisão e, caso isso acontecesse, seriam re-analisados à luz dessa evolução.

3.2 Revisão da Literatura sobre Métodos de Desenvolvimento de Aplicações Web

As seções a seguir definem o protocolo adotado na revisão da literatura, os dados da execução da revisão e os resultados obtidos com a análise dos estudos selecionados nessa revisão.

3.2.1 Protocolo da Revisão

3.2.1.1 Questão de Pesquisa

O contexto de aplicação dessa revisão refere-se aos métodos de desenvolvimento que tratam requisitos de aplicações Web.

- **Questão:** quais métodos de desenvolvimento que tratam requisitos de aplicações Web têm sido propostos na literatura técnica?
- **População:** projetos de desenvolvimento Web
- **Intervenção:** métodos de desenvolvimento Web que tratam requisitos de aplicações Web
- **Resultado:** métodos de desenvolvimento Web

3.2.1.2 Estratégia para Pesquisa dos Estudos Primários

Escopo da Pesquisa

A pesquisa será realizada exclusivamente em bases de dados eletrônicas, utilizando-se da máquina de busca disponibilizada na web pelas bibliotecas digitais. A pesquisa será realizada pelos campos *título*, *palavras-chave* e *resumo*. Serão pesquisados somente estudos cujo ano de publicação seja maior ou igual a 2004.

A fonte de pesquisa será a Scopus⁶, que indexa estudos das fontes IEEE Xplore⁷, ACM Digital Library⁸ e Springer⁹. Nenhuma restrição adicional será aplicada. Dessa forma, todos os artigos disponíveis nessa base são considerados artigos potenciais.

Termos utilizados na Pesquisa

- *web*
- *requirements*
- *analysis specification modeling modelling*
- *process method methodology approach technique metamodel*

A *string* de busca foi definida através do uso dos conectores *AND* e *OR* juntamente com o grupo de termos definidos anteriormente. Assim, a *string* de busca foi definida como:

web and requirements and (analysis or specification or modeling or modelling) and (process or method or methodology or approach or technique or metamodel)

⁶ www.scopus.com

⁷ ieeexplore.ieee.org

⁸ portal.acm.org

⁹ www.springerlink.com

É importante ressaltar que, na construção da *string* de busca evitou-se o uso de termos compostos como “*web requirements*” ou “*requirements analysis*” com o objetivo de evitar excluir estudos que não usassem exatamente essa terminologia, mesmo que isso implicasse em esforço maior na análise preliminar dos estudos recuperados.

3.2.1.3 Critérios e procedimentos de seleção de estudos

Os critérios de inclusão/exclusão dos artigos são:

- Os artigos devem estar disponíveis na web;
- Os artigos devem estar escritos em inglês;
- Os artigos devem descrever um método de desenvolvimento que trate, ou que inclua fases que tratem, dos requisitos de aplicações Web, independente da cobertura parcial ou total do ciclo de vida de desenvolvimento, e;
- Os artigos não devem descrever métodos que utilizam os requisitos somente para obter outros artefatos ou produtos, sem descrever como tratá-los.

A partir da leitura do título e do resumo do artigo, o pesquisador fará a seleção preliminar e gerará uma lista de artigo para leitura. Os artigos que suscitarem dúvidas nessa fase também deverão ser selecionados para leitura.

Para todos os artigos selecionados para leitura, serão recuperados os textos completos na web. Caso algum desses artigos não esteja disponível na web, por qualquer motivo, ele será marcado como “indisponível”. Após a leitura completa, os artigos que não atenderem a todos os critérios de inclusão/exclusão serão descartados.

Não foram definidos critérios ou procedimentos adicionais em relação à qualidade dos estudos. Somente os critérios de inclusão/exclusão serão usados como referência para incluir ou excluir um artigo da lista de artigos aprovados.

3.2.1.4 Estratégia para Extração de Dados

Para cada artigo aprovado pelos critérios e procedimentos de seleção deverão ser extraídos:

- Descrição do método;
- Artefatos gerados durante o tratamento de requisitos;
- Técnicas adotadas para análise/especificação/validação dos requisitos;
- Ferramenta para apoiar a fase de tratamento de requisitos, e;

- Existência de apoio para geração de testes a partir dos requisitos.

3.2.2 Execução da Revisão

A *string* de busca foi executada por ano (de 2004 a 2010), para facilitar a recuperação e gerenciamento da lista de artigos candidatos. A Tabela 3-1 e a Tabela 3-2 resumem os dados da execução:

Tabela 3-1 - Resumo da execução da revisão da literatura

Ano	No. de artigos recuperados	No. de artigos selecionados	No. de artigos indisponíveis	No. de artigos aprovados
2004	234	6	0	1
2005	260	2	0	2
2006	324	7	0	5
2007	404	7	0	3
2008	388	8	2	2
2009	392	6	0	4
2010	359	11	4	4
Total	2361	47	6	21

Tabela 3-2 - Distribuição dos 21 artigos aprovados pelos métodos Web

Ano	Método	No. de artigos	Referências
2004	WebGen	1	LOH e ROBEY, 2004
2005	AWDP	1	WOOKJIN <i>et al.</i> , 2005
	OOWS	1	VALDERAS <i>et al.</i> , 2005
2006	WebML	1	MORENO <i>et al.</i> , 2006
	WebRE	2	ESCALONA e KOCH, 2006 KOCH <i>et al.</i> , 2006
	HM3	1	CÁCERES <i>et al.</i> , 2006
	Modelagem de Aplicações Orientadas a Dados	1	ADAMKÓ, 2006
2007	Ariadne	1	MONTERO <i>et al.</i> , 2007
	WebML	1	MORENO <i>et al.</i> , 2007
	OOWS	1	VALDERAS <i>et al.</i> , 2007
2008	WRM	1	MOLINA <i>et al.</i> , 2008
	NDT	1	ESCALONA e ARAGÓN, 2008
2009	Abordagem para Análise de Requisitos com i*	1	GARRIGÓS <i>et al.</i> , 2009
	Modelagem de Navegação Orientada a Aspectos	1	CASALÁNGUIDA e DURÁN, 2009
	Análise de Requisitos com Geração de Protótipos	1	OGATA e MATSUURA, 2009
	Desenvolvimento Web por Usuários Finais	1	DE SILVA <i>et al.</i> , 2009
2010	WebTDD	1	LUNA <i>et al.</i> , 2010 ^a
	UWA	1	BERNARDI <i>et al.</i> , 2010
	Análise de Requisitos com Geração de Protótipos	1	OGATA e MATSUURA, 2010

	Abordagem para Análise de Requisitos com i*	1	AGUILAR <i>et al.</i> , 2010
Total	16	21	

3.2.3 Resultados obtidos com a Análise dos Estudos

A partir dos 21 artigos aprovados foram identificados 16 métodos que lidam com requisitos para aplicações Web. A essa lista foram mesclados os 10 métodos analisados anteriormente por ESCALONA e KOCH (2004), perfazendo um total de 24 métodos. É importante ressaltar que 2 métodos analisados por ESCALONA e KOCH (2004) (WebML e NDT) também foram selecionados pela revisão da literatura, indicando que esses métodos ainda continuam ativos no que se refere ao tratamento de requisitos de aplicações Web.

Nas seções a seguir os 24 métodos aprovados na revisão da literatura técnica são apresentados de forma resumida.

3.2.3.1 WSDM – Web Site Design Method

O WSDM (DE TROYER e LEUNE, 1997) é uma abordagem centrada no usuário cuja fase inicial se concentra na modelagem dos usuários. Nessa fase os diferentes tipos de público-alvo são agrupados de acordo com interesses relacionados às funcionalidades ou informações a serem providas pelo sistema. O resultado final dessa análise é uma hierarquia de grupos de usuários onde, para cada grupo, são definidos, em linguagem natural, os requisitos funcionais, de informação, de usabilidade, além de outras características como idioma e experiência. Em seguida é executada a modelagem conceitual, onde são criados os modelo de informação, o modelo funcional e o modelo navegacional. Por fim, os modelos de implementação detalham as páginas da aplicação juntamente com o projeto da apresentação e o modelos de dados.

3.2.3.2 SOHDM – Scenario-based Object-oriented Hypermedia Design Methodology

O SOHDM (LEE *et al.*, 1998) foi o primeiro método Web a incluir uma fase específica de análise de requisitos, onde são explorados cenários para a descrição desses requisitos. Cenários são descritos graficamente através de uma notação proprietária chamada Gráfico de Atividade do Cenários (SAC – *Scenario Activity Chart*) e descrevem a interação entre o usuário e o sistema quando um evento ocorre. Nesse gráfico são especificados o fluxo de atividades, os objetos envolvidos e as transações executadas. A partir desses cenários são obtidos os modelos conceituais (modelos de

classes) e, no próximo passo, as classes são agrupadas em unidades navegacionais e o modelo de navegação é gerado. Por fim, as páginas Web, a interface com o usuário e o banco de dados são implementados.

3.2.3.3 RNA – Relationship Navigation Analysis

RNA (BIEBER *et al.*, 1998) não é um método para análise, mas sim um método que prevê uma série de atividades a serem executadas durante a fase de análise:

Análise do ambiente: características do público-alvo são avaliadas e classificadas (de forma semelhante ao que ocorre na modelagem de usuários do WSDM);

- Análise dos elementos de interesse: são definidas telas, documentos, relatórios, dados, dentre outros;
- Análise do conhecimento: definição do esquema da aplicação com a identificação de objetos, processos e operações e seus relacionamentos;
- Análise da navegação: o esquema definido na fase anterior é enriquecido e incrementado com componentes específicos para dar suporte à navegação, e;
- Implementação da análise: define como os modelos produzidos serão convertidos para uma linguagem de programação.

O RNA não define uma notação nem tampouco orientações para modelagem, apenas orientações sobre as ações a serem executadas em cada fase do método.

3.2.3.4 HFPM – Hypermedia Flexible Process Modeling

O método HFPM (OLSINA, 1998) cobre todo o ciclo de vida do desenvolvimento, mas não apresenta novas técnicas ou modelos. Por isso, utiliza muitas das técnicas e modelos propostos pelo OOHDM. Ao todo, o processo de desenvolvimento do HFPM possui 13 fases, sendo que a fase de Modelagem dos Requisitos é composta de 5 atividades:

- Descrição do problema: não prevê nenhuma técnica específica, logo o uso da linguagem natural;
- Descrição de funcionalidades: propõe a utilização de casos de uso;
- Modelagem de dados: propõe o uso de diagramas de classes;
- Modelagem da interface com o usuário: usa protótipos ou *sketches* para apresentação ao cliente, e;
- Descrição de requisitos não-funcionais: também em linguagem natural.

Após essa fase, são criados os modelos conceituais, de navegação e de apresentação, seguindo os preceitos do métodos OOHDM.

3.2.3.5 OOHDM – Object-Oriented Hypermedia Design Model

O OOHDM (SCHWABE et al., 1996) é um método baseado em modelos e orientados a objetos para especificação e construção de aplicações hipermídia concentrando-se em dois aspectos: navegacional e estrutural. Na sua versão original o OOHDM não previa atividades relacionadas ao tratamento de requisitos. Entretanto, esse recurso foi introduzido por VILAIN et al. (2000) através dos UIs (*User Interaction Diagram*). O método OOHDM compreende cinco etapas que combinam um estilo de desenvolvimento incremental, iterativo e baseado em protótipos:

- Identificação e definição de requisitos: utiliza uma abordagem padrão de identificação de atores e casos de uso. Os Casos de Uso são, então, representados através de Diagramas de Interação do Usuário (UID), que provêem uma representação gráfica proprietária da interação ator x sistema;
- Modelo conceitual: são gerados a partir dos UIs através da aplicação de uma série de regras. Utiliza a abordagem orientada a objetos (classes, associações, atributos, generalização/especialização e agregação) para representação dos conceitos relacionados ao domínio da aplicação;
- Modelo navegacional: descreve a visão navegacional elaborada a partir do modelo conceitual. Uma notação própria é usada para descrever primitivas de acesso e de navegação entre as classes;
- Modelo de interface abstrata: define quais objetos da interface são perceptíveis ao usuário, possibilitando a criação de diferentes interfaces para o mesmo modelo navegacional. O OOHDM utiliza a abordagem de visão abstrata de dados para descrever a interface do usuário de uma aplicação hipermídia, e;
- Implementação: consiste no mapeamento dos modelos de interface abstrata e do modelo navegacional em objetos concretos na plataforma de implementação escolhida.

3.2.3.6 UWE – UML-based Web Engineering

O UWE (HENNICKER e KOCH, 2000) é um processo de desenvolvimento para aplicações Web com foco na sistematização, customização e geração semi-automática da aplicação. O UWE prevê a classificação dos requisitos em: conteúdo, estrutural, apresentação, adaptação e modelo do usuário. O processo de desenvolvimento de aplicações Web consiste em quatro passos:

- Análise de requisitos: construção do modelo de casos de uso contendo todas as informações relevantes para implementação dos casos de uso, que podem ser representados em linguagem natural ou diagramas de atividades;
- Projeto conceitual: construção do modelo conceitual contendo as abstrações relacionadas ao domínio da aplicação.
- Projeto navegacional: contendo inicialmente o modelo de espaço navegacional, onde são definidos quais conceitos podem ser visitados durante a navegação, e posteriormente evoluindo para o modelo de estrutura navegacional, onde são definidas as estruturas de acesso às informações (menus, paginação e índices).
- Projeto de apresentação: construção do modelo de apresentação, contendo os componentes da interface responsáveis pela interação com o usuário (formulários, imagens, botões e todos os demais elementos gráficos).

3.2.3.7 WebML – Web Modeling Language

A WebML (CERI et al., 2000) é uma linguagem de especificação de alto nível voltada para o desenvolvimento de aplicações Web que manipulam grandes quantidades de dados (*data-intensive Web applications*). Faz parte do ciclo de vida proposto pela WebML uma fase de especificação de requisitos, onde são definidos os casos de uso, complementados por uma descrição semi-estruturada em linguagem natural. A utilização de diagramas de atividades para casos de uso complexos também é sugerida. A partir da especificação dos requisitos são desenvolvidos o modelo de dados, o modelo de hipertexto e o modelo de apresentação. Finalmente, os testes de aceitação propostos são voltados para verificação de requisitos não-funcionais.

3.2.3.8 W2000

O W2000 (BARESI et al., 2001) é um método originado do método HDM (GARZOTTO et al., 1993) que tenta promover o refinamento gradativo dos modelos desde a fase da análise. O ciclo de vida proposto pelo W2000 é composto de 3 fases: análise de requisitos, projeto hipermídia e projeto funcional. Na fase de análise de requisitos são previstas duas atividades:

- Análise de requisitos funcionais: define o modelo de casos de uso e seus atores, e;
- Análise de requisitos navegacionais: também usa o modelo de casos de uso para representar as possibilidades de navegação do ator. Nesse caso, usa uma notação gráfica que estende a notação UML original.

A partir desses casos de uso são projetados os modelos conceitual e navegacional e, a partir desses modelos, são definidos os diagramas de seqüência que descrevem as funcionalidades do sistema.

3.2.3.9 Design-driven Requirements Elicitation (DDDP¹⁰)

O DDDP é parte do processo proposto por LOWE e EKLUND (2002) para desenvolvimento de aplicações Web. Esse método prevê a identificação, definição e validação dos requisitos durante a fase de projeto (*design*), ou seja, o processo é organizado de tal forma que as atividades de engenharia de requisitos e projeto ocorrem simultaneamente. Esse processo baseia-se na prototipação a fim de explorar possíveis soluções para o problema. Os usuários definem os requisitos a partir da avaliação dos protótipos e estes são alterados para refletir os requisitos, dentro de um processo iterativo que procura reduzir as dúvidas dos usuários. Um ciclo completo nesse processo possui três fases: avaliação, especificação e construção. É importante destacar que, nesse processo, a especificação do sistema é o próprio protótipo. A concepção do processo foi baseada na análise das “melhores práticas” de desenvolvimento de aplicações Web comerciais. Os requisitos são classificados de várias formas, mas são tratados da mesma maneira.

3.2.3.10 UWA – Ubiquitous Web Application

O método UWA (PERRONE e PAOLINI, 2003; BERNARDI *et al.*, 2010) divide o processo de desenvolvimento nas seguintes fases:

- Elicitação de requisitos: nesta fase os *stakeholders* e os requisitos são identificados através de uma metodologia orientada a objetivos, ou seja, são identificados os objetivos de cada *stakeholders* no uso do sistema. Nesse momento os requisitos são associados aos objetivos e são classificados em: Conteúdo (a informação que está sendo disponibilizada), Acesso (estratégias para localização da informação), Navegação (características relacionadas ao hipertexto), Apresentação (organização das páginas), operação do usuário (operações disponíveis ao usuário) e operação do sistema (operações que o sistema deve executar);
- Projeto hipermídia: nessa fase são desenvolvidos o modelo da informação (entidades e seus atributos e relacionamentos), modelo de navegação (como a informação é organizada em nós e os links entre esses nós) e

¹⁰ Os autores não definiram um nome para a metodologia, por isso essa sigla foi usada nesta tese para referenciá-la.

modelagem da publicação (como a informação será apresentada ao usuário);

- Projeto das operações: define quais operações estarão disponíveis ao usuário com suas respectivas pré e pós condições;
- Projeto transacional: define as transações, ou seja, uma seqüência de operações que define um processo de negócio, e;
- Projeto da customização: especifica regras que definem como a aplicação irá se adaptar a diferentes contextos de uso.

3.2.3.11 NDT – Navigational Development Techniques

O NDT (ESCALONA *et al.*, 2003) é um método que inclui apenas as fases de tratamento de requisitos e análise, onde são aplicadas técnicas específicas para obtenção de modelos de análise. No NDT cada requisito é classificado em requisito de armazenamento, do usuário, funcional, interação ou não-funcional. Para cada tipo de requisito existe um gabarito específico que direciona o desenvolvedor no seu preenchimento, criando um processo sistemático. Com o apoio dos gabaritos os requisitos são descritos em linguagem natural estruturada. Requisitos funcionais são descritos através de um gabarito para casos de uso. Após a especificação, os requisitos são revisados e é criada uma matriz de rastreabilidade para verificar se todos os requisitos foram especificados. A fase de análise tem início com a geração dos modelos preliminares de análise a partir dos modelos de especificação. Esses modelos são, então, alterados pelos desenvolvedores para obter os modelos de análise. A partir desse ponto, o NDT sugere o uso dos modelos do método UWE no restante do processo de desenvolvimento.

3.2.3.12 WebGen

O WebGen (LOH e ROBEY, 2004) propõe uma abordagem para geração do modelo navegacional e de trechos de código para a aplicação final a partir da descrição de casos de uso. O processo se inicia com a especificação textual dos casos de uso, onde cada ação descrita deve ser classificada como uma ação do ator ou uma ação do sistema. Em seguida, para cada cenário do caso de uso são definidas páginas com as informações que cada uma deve conter. Para cada cenário o desenvolvedor mapeia os ciclos “ação do ator/ação do sistema” em um ciclo *request/response*, que representa, no jargão do protocolo HTTP, uma solicitação ao sistema e uma resposta do sistema. Por fim, para cada ciclo *request/response* são definidos os eventos necessários para que a resposta prevista no cenário do caso de uso seja gerada a partir da requisição enviada. Adotando o paradigma da orientação a

objetos, os eventos podem ser comparados às mensagens trocadas entre os objetos para que a resposta esperada seja gerada a partir da requisição. Quando todos os cenários de um caso de uso são especificados segundo esta abordagem, as páginas Web e os eventos que descrevem esses cenários formam uma árvore que representa todos os possíveis eventos e páginas Web que realizam o caso de uso. A abordagem também prevê a geração parcial do código da aplicação a partir dessa árvore de eventos.

3.2.3.13 Agile Web Development Process (AWDP¹¹)

A metodologia ágil AWDP (WOOKJIN *et al.*, 2005) inclui o uso de modelos orientados a objetos e ferramentas aliados a um processo ágil e sistemático para o desenvolvimento de aplicações Web. A agilidade, nesse contexto, visa atender às solicitações constantes de mudança em virtude de pressões do mercado. O processo é dividido em duas grandes fases: Percepção e Produção. Na fase de Percepção são definidos os modelos conceituais relacionados ao domínio do problema e os casos de uso, descritos em linguagem natural. A fase de Produção é sub-dividida em 2 fases: Sofisticação e Construção. A fase de Sofisticação representa o detalhamento da análise e projeto e a criação dos testes. Para a análise e projeto são utilizados: (1) *storyboards*, que definem a interação da aplicação com o usuário; (2) modelo navegacional e modelo de componentes; (3) prototipação da interface, e; (4) validação dos *storyboards* e protótipos pelo cliente. Tanto o modelo navegacional quanto o modelo de componentes tratam elementos diretamente ligados à tecnologia utilizada no desenvolvimento, ou seja, o modelo navegacional define as páginas e os links entre elas, enquanto o modelo de componentes representa a troca de mensagens entre as classe de implementação (diagrama de seqüência). Por fim, como afirmam os próprios autores, o AWDP é um método voltado para desenvolvimento e manutenção de sistemas de pequeno porte.

3.2.3.14 OOWS – Object-Oriented Web Solution

O método OOWS (PASTOR *et al.*, 2001) em sua versão original não previa atividades relacionadas ao tratamento dos requisitos. A abordagem proposta por VALDERAS *et al.* (2005) preenche essa lacuna usando a metáfora de tarefas. São identificados os usuários do sistema e uma série de tarefas que esses usuários devem executar usando o sistema. Posteriormente, essas tarefas são refinadas até a

¹¹ Os autores não definiram um nome para a metodologia, por isso essa sigla foi usada nesta tese para referenciá-la.

geração de um grupo de tarefas elementares. Uma tarefa elementar é aquela composta de tarefas associadas ao ator ou do sistema, mas não a ambos. Os autores propõem o uso do CTT (PATERNO *et al.*, 1997) para descrição dessa taxonomia de tarefas. Para descrição das tarefas elementares é usada uma notação inspirada no diagramas de atividade da UML (OMG, 2010a), mas que lança mão de vários construtores específicos e com semântica própria para definir as ações básicas e as informações que elas manipulam. Nessa fase também são construídos *templates* para definir a informação a ser armazenada no sistema, ou seja, os requisitos de dados. A partir da descrição das tarefas elementares e dos requisitos de dados são aplicadas transformações para geração dos modelos navegacionais conforme preconizados pelo OOWS. Os modelos navegacionais representam visões do modelo conceitual e especificam as estruturas de acesso à informação para cada tipo de usuário identificado na elicitação de requisitos. A criação do modelo navegacional é dividida em duas atividades: (1) Gestão de Usuários, onde são capturados todos os possíveis atores da aplicação e suas associações, e (2) Especificação de Propriedades Navegacionais, onde são definidos os contextos navegacionais para cada ator, ou seja, que informações e serviços estarão acessíveis através da navegação.

3.2.3.15 WebRE - Web Requirements Engineering

O WebRE (ESCALONA e KOCH, 2006; KOCH *et al.*, 2006) reúne conceitos dos métodos NDT e UWE, e define um metamodelo que representa os requisitos de aplicações Web sob duas óticas: comportamental e estrutural. Os estereótipos para classificação dos requisitos nessas duas visões são apresentados na Tabela 3-3.

Tabela 3-3 - Estereótipos do metamodelo do WebRE

Visão	UML	WebRE	Definição
Comportamental	Caso de uso	<i>Navigation</i>	Modelo casos de uso que possuem somente navegação, ou seja, ações do tipo <i>Browse</i> e <i>Search</i>
		<i>WebProcess</i>	Modela casos de uso que possuem transações
	Ação	<i>Browse</i>	Modela o acionamento de um link
		<i>Search</i>	Modela uma consulta realizada pelo ator
		<i>User Transaction</i>	Modela uma transação iniciada pelo ator
	Ator	<i>WebUser</i>	Ator que interage com a aplicação
Estrutural	Classe	<i>Node</i>	Unidades de informação com as quais o usuário interage
		<i>Content</i>	Informações manipuladas

			pelo sistema
		<i>WebUI</i>	Páginas que são apresentadas ao usuário. Podem conter vários <i>Nodes</i>

Casos de uso são representados graficamente, através de diagramas de atividades onde são aplicados os estereótipos *Browse*, *Search* e *User Transaction* (Tabela 3-3). Os insumos e produtos das ações, nesse diagrama, são estereotipados como *Content*. A partir desses modelos são aplicadas transformações para geração do Modelo de Conteúdo ou Modelo de Domínio e do Modelo Navegacional no padrão definido pelo método UWE. Casos de uso também podem ser representados textualmente, usando o gabarito de casos de uso proposto pelo método NDT, porém, nesse caso, não podem ser usados pelos mecanismos de transformação.

3.2.3.16 HM³ - Hypertext Modeling Method of MIDAS

O método HM³ (CÁCERES *et al.*, 2006), que faz parte do método MIDAS (CASTRO *et al.*, 2004), defende que o modelo navegacional seja obtido a partir dos serviços demandados pelo usuário, ao invés de ser estruturado a partir do modelo conceitual, como ocorre na maioria dos métodos Web. Para tal é usado o modelo de casos de uso, que representa a visão comportamental do sistema, e o modelo conceitual de dados, que representa a visão estrutural do sistema. Casos de uso são decompostos, usando os conceitos de *include* e *extend*, em serviços estruturais, responsáveis somente pela visualização dos dados, ou funcionais, quando ocorre algum tipo de interação entre o ator e o sistema. Cada serviço que compõe o caso de uso pode ser visto como um caminho alternativo que descreve uma possibilidade de execução do caso de uso e que representa um “serviço” na visão do usuário. Para especificar o encadeamento entre os serviços estruturais e funcionais é criado um diagrama de atividades da UML (OMG, 2010a) para cada caso de uso onde as ações do diagrama são os serviços do caso de uso. Ao final, cada serviço do caso de uso, ou seja, cada ação do diagrama de atividades, é descrito no modelo navegacional como um nó, ou *slice* na terminologia do método, que contem os dados (retirados do modelo conceitual) a serem tratados nesse ponto. Os links entre os nós do modelo navegacional são definidos a partir dos fluxos de controle existentes no diagrama de atividades. O comportamento do sistema é implementado a partir dos serviços do caso de uso explorando o conceito e *web services*, como prevê a metodologia, ou através de outra tecnologia. A partir dos artigos analisados não foi possível observar atividades de validação dos modelos ou relacionadas à garantia da qualidade do produto.

3.2.3.17 Modelagem de Aplicações Orientadas a Dados

A metodologia proposta por ADAMKÓ (2006) é voltada para a modelagem de aplicações de pequeno e médio porte, como afirma o próprio autor, orientadas a dados. O desenvolvimento tem início com a análise do domínio e a construção de casos de uso que são detalhados através de diagramas de atividades. A partir desses elementos são construídos os modelo estruturais, ou seja, modelos de classes que definem as entidades a serem tratadas no contexto da aplicação. O modelo navegacional é construído a partir das classes do modelo estrutural e é posteriormente decorado com componentes de navegação (*index*, *query* e *menu*). Nesse caso, o modelo navegacional adotado é bastante semelhante ao usado no métodos UWE (HENNICKER e KOCH, 2000). A partir desses modelos serão gerados o esquema do banco de dados e as páginas Web descritas segundo a tecnologia XForms¹². Para implementar restrições adicionais relacionados ao negócio, além daquelas restrições estruturais já previstas no modelo de dados, o método prevê algumas possibilidades, como a introdução de trechos de código através de restrições associadas às classes. Esses trechos seriam implementados, posteriormente, nas classes responsáveis pelo gerenciamento dos dados ou diretamente no SGBD (Sistema de Gerenciamento de Banco de dados).

3.2.3.18 ADM - Ariadne Development Method

O método ADM foi inicialmente proposto por DIAZ *et al.* (2005), mas contemplava somente as fases relativas ao projeto (*design*) da aplicação Web. MONTERO *et al.* (2007) acrescentam o tratamento dos requisitos e propõem um processo flexível e detalhado que permite que os desenvolvedores tratem diferentes questões de projeto em 6 níveis de abstração: como o sistema está estruturado (estrutura), quais capacidades de navegação são oferecidas (navegação), como o sistema reage a eventos externos (comportamento), como o sistema funciona internamente (processo), quais as características da apresentação da informação (apresentação) e quais regras de acesso serão aplicadas para prover um ambiente seguro (acesso). A construção dos modelos em cada um desses níveis está dividida nas 3 fases do processo proposto: projeto conceitual, projeto detalhado e avaliação. O projeto conceitual envolve a definição da estrutura abstrata do sistema (um tipo de WBS – *Work Breakdown Structure*), a descrição dos principais serviços da aplicação, a definição das opções de navegação, a especificação das entidades e a definição de

¹² <http://www.w3.org/MarkUp/Forms/>

políticas de segurança. O projeto detalhado envolve o detalhamento de todos os elementos definidos na fase anterior e a especificação da apresentação. Na fase de avaliação são criados e avaliados os protótipos da aplicação. A completude, consistência e integridade dos vários artefatos gerados durante as fases de projeto conceitual e detalhado são garantidas através de um conjunto de regras. Todo o processo é apoiado por uma ferramenta denominada AriadneTool.

3.2.3.19 WRM - Web Requirements Metamodel

O WRM (MOLINA *et al.*, 2008) é uma abordagem para modelagem e gerenciamento de requisitos funcionais e não-funcionais, onde cada requisito está sempre associado a um objetivo que, por sua vez, é definido por um *stakeholder*. O WRM define que o requisito seja descrito em linguagem natural, mas, para prevenir ambigüidades, essa descrição tem que ser feita a partir de termos armazenados no glossário do projeto. O WRM prevê que cada requisito pode ser refinado através de várias descrições. Assim, o desenvolvedor pode armazenar nessas descrições casos de uso, cenários, *templates*, modelos ou qualquer outro artefato desejado. O WRM também prevê um catálogo para armazenamento de leis, regras ou orientações, que pode ser reusado entre vários projetos. Apesar dos autores declararem que o WRM é destinado à modelagem de requisitos de sistema de informação Web, esse arcabouço pode ser utilizado para modelagem e gerenciamento de aplicações de software em geral.

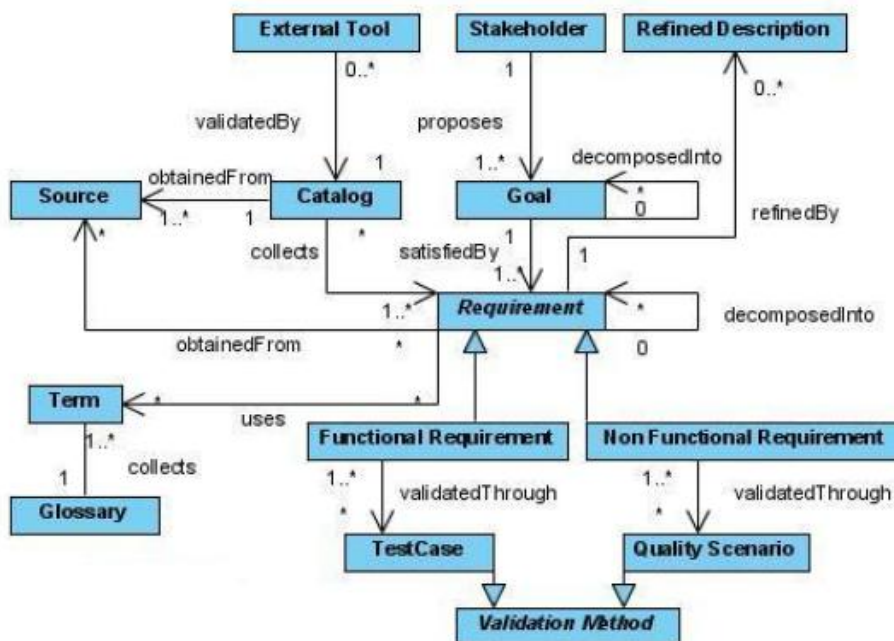


Figura 3-1 - Metamodelo de requisitos do WRM

3.2.3.20 Abordagem para Análise de Requisitos com i*

A abordagem apresentada por GARRIGÓS *et al.* (2009) e AGUILAR *et al.* (2010) é voltada para a análise de requisitos de aplicações Web a partir de uma versão estendida do *framework* i* (YU, 1997). O i* é uma linguagem de modelagem voltada para as fases iniciais do entendimento do domínio do problema. Ele usa o modelo de dependência estratégica (SD - *Strategic Dependency*) para descrever a dependência entre os vários atores do contexto organizacional e o modelo de raciocínio estratégico (SR - *Strategic Rationale*) para capturar, para cada ator, o seu objetivo (*Goal*), as tarefas (*Task*) para atingir esse objetivo, os recursos (*Resource*) necessários e os objetivos secundários (*Softgoals*).

Para adequar o *framework* i* à modelagem de requisitos Web foram acrescentados novos conceitos para representar os requisitos de conteúdo (conceitos do domínio), serviço (funcionalidades do sistema), navegação, apresentação e personalização. Assim, o conceito de *Task* foi estendido para *Service*, *Navigational*, *Layout* e *Personalization* e *Layout*, enquanto o conceito de *Resource* foi estendido para *Content*. O modelo SR do i* é, então, construído a partir desses novos conceitos, ou seja, especializando as tarefas e os recursos segundo os tipos de requisitos relacionados às aplicações Web. A abordagem prevê que, a partir dos modelos SR sejam gerados os modelos preliminares de domínio e navegação. Os autores declaram que é possível gerar esses modelos para vários métodos Web, como OO-H, OOWS ou UWE.

3.2.3.21 Modelagem de Navegação Orientada a Aspectos

CASALÂNGUIDA e DURAN (2009) propõem que modelagem navegacional seja realizada levando-se em consideração as tarefas que o usuário deseja executar na aplicação, modelando a complexidade das entradas de dados necessárias a essas tarefas, e não somente como uma simples visão do modelo conceitual. O método proposto baseia-se totalmente nos conceitos e processos do método WebRE e propõe que os casos de uso sejam modelados segundo uma abordagem orientada a aspectos proposta por SOUZA *et al.* (2004). O objetivo é modelar casos de uso *crosscutting*, ou seja, casos de uso que fornecem funcionalidades que são reusadas em vários outros casos de uso.

Os casos de uso são classificados, segundo o método WebRE, em *Navigation* ou *WebProcess*. O método propõe que o modelo navegacional dos casos de uso do tipo *WebProcess* sejam criados com um novo conjunto de estereótipos que visam detalhar como as transações definidas no caso de uso ocorrem em termos de interface

com o usuário e funcionalidades do sistema. Essa visão detalhada não é prevista no método WebRE.

3.2.3.22 Análise de Requisitos com Geração de Protótipos

O método proposto por OGATA e MATSUURA (2009) explora a especificação de requisitos através de casos de uso que são descritos usando diagramas de atividades. Esses diagramas são explorados em um processo iterativo onde protótipos são gerados diretamente a partir dos diagramas e são usados para validar/refinar os requisitos, que por sua vez alteram novamente os diagramas gerando novos protótipos. Para que seja possível gerar protótipos a partir dos diagramas, estes contam com as seguintes características:

- O diagrama possui raias para determinar de quem é a responsabilidade pela ação (ator ou sistema);
- Fluxos alternativos e de exceção são representados como nós de decisão;
- Os dados de entrada ou saída da aplicação são representados como fluxos de objetos, e;
- Os campos de entrada são definidos como ações com palavras-reservadas como *input <nome>*, que cria uma caixa de texto para entrada do campo *nome*, ou *select-single-from-list <nota>*, que cria uma lista de seleção com os valores de *nota*.

O método conta, ainda, com outros diagramas:

- Diagrama de classes (modelo de domínio);
- Diagrama de objetos: responsável por definir os valores a partir dos quais o protótipo é avaliado;
- Diagrama de cenários: define os cenários que serão executados pelos clientes juntamente com os dados usados nessa execução, e;
- Diagrama Navegacional: é representado por diagramas de atividades que representam a seqüência de invocação dos serviços e quem pode acessá-los. Nesse caso, cada ação corresponde a um serviço, os fluxos de controle representam os links entre esses serviços e as raias representam os atores que podem acessar esses serviços.

3.2.3.23 Desenvolvimento Web por Usuários Finais

DE SILVA *et al.* (2009) propõem uma abordagem iterativa baseada em casos de uso que permite o envolvimento dos usuários finais na especificação da aplicação e a sua geração semi-automática a partir dessa especificação. A especificação da aplicação é construída a partir da especificação dos casos de uso, da definição do

modelo de objetos (modelo do domínio) e do desenho da interface com o usuário. Uma ferramenta apóia todo esse processo. A especificação e a geração da aplicação são apoiadas por um framework que estende um conjunto de componentes chamado CBEADS[®]. (*Component Based E-Application Development and Deployment Shell*). O metamodelo criado com a extensão do CBEADS[®] tenta reaproveitar conceitos que são comuns a um conjunto de domínios e define uma estrutura a partir da qual a aplicação pode ser especificada referenciando diretamente esses conceitos e suas características, de forma bastante semelhante ao uso de uma DSL (Domain Specific Language). Como os próprios autores afirmam, essa é uma abordagem voltada para organizações de pequeno e médio porte.

3.2.3.24 WebTDD

O método WebTDD (LUNA *et al.*, 2010a) aplica o mesmo estilo adotado no desenvolvimento dirigido por testes (TDD – *Test Driven Development*), porém, ao invés de se basear na codificação, baseia-se na modelagem para geração da aplicação. No WebTDD os requisitos são adicionados incrementalmente, em ciclos de desenvolvimento curtos, através de diagramas descritos na linguagem específica de domínio WebSpec (LUNA *et al.*, 2010b). O diagrama da WebSpec tem dois elementos-chave:

- Interação: representa uma página Web que pode conter elementos gráficos como botões, listas, rótulos, dentre outros, e;
- Navegação: são ligações entre as interações, ou seja links entre páginas, que podem estar associados à restrições ou trechos de código escritos em WebSpec.

Com esses diagramas, os comportamentos do sistema durante a interação ator-sistema são definidos visualmente. Esses diagramas são explorados no processo WebTDD, que é composto das seguintes fases:

- 1) Captura de requisitos: é feita através de *mockups* e diagramas WebSpec;
- 2) Geração de testes: são criados testes para verificar a interação a partir dos diagramas WebSpec;
- 3) Execução dos testes: assim como no TDD, os testes são executados para verificar se a especificação atende aos requisitos do usuário;
- 4) Criar e aprimorar modelos: nessa fase um conjunto de modelos são criados para a geração da aplicação. O WebTDD não define qual abordagem usar nessa fase, e;
- 5) Execução dos testes na aplicação: os mesmo testes definidos na fase 2 são re-executados na aplicação final. Caso seja reprovado volta ao passo 4.

Caso seja aprovado esse conjunto de requisitos é liberado e o processo reinicia com um novo conjunto de requisitos.

3.3 Resumo Comparativo dos Métodos Web

A Tabela 3-4 resume as principais características dos métodos selecionados pela revisão da literatura técnica, conduzida no escopo desta tese, e dos métodos analisados por ESCALONA e KOCH (2004). Como a análise realizada por ESCALONA e KOCH (2004) não abrangia a verificação de ferramentas nem a existência de apoio a testes, os artigos que relatam esses métodos também foram lidos e analisados para verificação desses itens. Assim, os valores resumidos na Tabela 3-4 foram extraídos dos artigos aprovados na revisão da literatura técnica e dos artigos que relatam os 10 métodos analisados por ESCALONA e KOCH (2004), de acordo com a estratégia de extração de dados estabelecida no protocolo da revisão.

Tabela 3-4 - Resumo dos métodos Web analisados

Método	Requisitos			Apoio a teste
	Análise/Especificação	Validação	Ferramenta	
WSDM	Linguagem natural	-	-	-
SOHDM	DFD Cenários Lista de eventos	-	-	-
RNA	Linguagem natural	-	-	-
HFPM	Casos de uso Linguagem natural Glossário Esboços da interface	Protótipo	-	-
OOHDM	Casos de uso Linguagem natural UIDs	-	-	-
UWE	Casos de uso Linguagem natural Diagrama de atividades Cenário Glossário	Protótipo Revisão <i>ad-hoc</i>	MagicDRAW	-
WebML	Casos de uso Linguagem natural semi-estruturada Diagrama de atividades Esboços da interface	-	WebRatio	-
W2000	Casos de uso Linguagem natural	-	-	-
DDDP	Protótipo	Protótipo	-	-
UWA	Casos de uso Linguagem natural	-	-	-
NDT	Casos de uso Linguagem natural semi-estruturada Gabaritos Glossário BNL	Protótipo Revisões <i>ad-hoc</i>	NDT-Tool	-

WebGen	Casos de uso Linguagem natural	-	-	-
AWDP	Casos de uso Linguagem natural	Protótipo	ProcessManager	
OOWS	ConcurTaskTrees Diagrama de atividades (com notação própria) Gabarito de dados	-	OOWS-Suite	-
WebRE	Casos de uso Linguagem natural semi- estruturada Diagrama de atividades estereotipado Gabaritos	-	-	-
HM ³	Casos de uso Linguagem natural Modelos conceituais	-	-	-
ADAMKO (2006)	Caso de uso Linguagem natural Diagrama de atividades	-	-	-
ADM	Linguagem natural Diagrama de estrutura Especificação de processos Catálogo de eventos	Protótipo	AriadneTool	-
WRM	Não define um artefato específico	-	<i>Plug-in no Eclipse</i>	-
GARRIGÓS <i>et al.</i> (2009)	Modelo de dependência estratégica Modelo de raciocínio estratégico	-	-	-
CASALÂNGUIDA <i>et al.</i> (2009)	ídem método WebRE	-	-	-
OGATA <i>et al.</i> (2009)	Caso de uso Diagrama de atividades	Protótipo	UI Prototype Generation Tool	-
DE SILVA <i>et al.</i> (2009)	Caso de uso Linguagem natural	Protótipo	Requirements Specification Tool	-
WebTDD	Diagramas em WebSpec	Protótipo	<i>Plug-in no Eclipse</i>	Geração de scripts

3.4 Contribuições da Revisão

A análise dos 24 métodos apresentados na seção 3.2.3 possibilitou uma visão mais ampla do estado-da-arte em termos de tratamento de requisitos por parte dos métodos Web. Apesar dessa revisão ter sido executada ao longo desses últimos anos, o cenário inicial não sofreu alterações significativas, podendo ser resumido nos seguintes tópicos:

- Casos de uso têm sido propostos em combinação com o uso de linguagem natural, o que sugere que o comportamento do sistema definido nesses artefatos é traduzido manualmente para os modelos de projeto, apesar de muitos métodos advogarem que geram esses modelos de forma automática;

- De um modo geral, não foram observadas preocupações em definir atividades de garantia da qualidade dos requisitos. Os métodos que procuram tratar essa questão, o fazem de forma *ad-hoc*;
- Não foi possível identificar sugestões em relação à estruturação do documento de requisitos de forma que os diversos modelos de especificação propostos possam estar integrados;
- O uso de protótipos é a técnica mais recorrente para validação dos requisitos. Entretanto, a maior parte desses protótipos representa apenas a visão estrutural relacionada à navegação e apresentação, negligenciando o comportamento relacionado às funcionalidades da aplicação;
- O apoio do diagrama de atividades da UML (OMG, 2010a) à descrição de casos de uso sem o uso de recursos adicionais não traz nenhum valor agregado ao entendimento do problema, pois não acrescenta nenhuma semântica a mais nessas descrições além daquelas que já existem quando é usado o formato textual;
- Parece haver uma cobertura inconsistente das atividades e técnicas propostas pelos métodos Web pelas ferramentas concebidas para apoiar essas atividades e técnicas. Isso parece ser contraproducente, já que grande parte desses métodos exploram modelos e transformações entre esses artefatos que parecem não fazer sentido sem o apoio ferramental adequado. Nesse contexto, a própria avaliação desses métodos, seja com um estudo de caso ou com um estudo controlado, só parece fazer sentido se o apoio ferramental adequado fizer parte do pacote onde o método está incluído, e;
- A falta de apoio aos testes de sistema, dentro da filosofia do modelo V, também pode ser observada. Esse cenário indica que os métodos Web não tem se preocupado em definir uma política integrada para garantia da qualidade do produto final, através do apoio ao planejamento/geração de testes de sistema.

3.5 Conclusão

Este capítulo teve como objetivo descrever os resultados de uma revisão da literatura técnica com o propósito de caracterizar os métodos de desenvolvimento que tratam requisitos de aplicações Web. A análise desses métodos forneceu indícios da existência de diversas oportunidades de pesquisa relacionadas a esse tema. Mais especificamente com relação à estruturação dos requisitos, é possível perceber que,

apesar da proposição de vários modelos para capturar as características específicas das aplicações Web (estado-da-arte), grande parte dos métodos explora mecanismos comumente encontrados na indústria para a especificação dos requisitos (estado-da-prática). Esse desalinhamento representa um risco para os projetos, pois a distância entre a forma como os requisitos são especificados e como os conceitos são capturados pelos modelos tende a dificultar a elaboração destes últimos, com efeitos negativos sobre a qualidade do produto final.

Assim, baseado no que pôde ser depreendido dos estudos de observação apresentados no capítulo 2, na análise dos métodos Web propostos na literatura técnica apresentada no capítulo 3, e nas observações realizadas no processo de desenvolvimento do SiGIC, mais precisamente sobre fatores que impactam a qualidade do produto, foram identificadas lacunas no que diz respeito ao tratamento de requisitos, principalmente com relação à estruturação desses requisitos em um arcabouço que possa servir de ponto de partida para exploração dos modelos preconizados nos métodos Web contemporâneos.

4 Abordagem para Especificação de Requisitos

Neste capítulo uma visão geral da abordagem proposta e seu escopo de atuação são apresentados, juntamente com o processo previsto para especificação dos requisitos e os diversos elementos que visam representar a especificação funcional sob a ótica das perspectivas de projeto Web e das visões estrutural e comportamental.

4.1 Introdução

A partir dos resultados dos estudos preliminares visando observar questões relacionadas ao uso das perspectivas Web (capítulo 2), da análise dos métodos de desenvolvimento Web propostos na literatura técnica (capítulo 3) e das experiências no contexto do desenvolvimento do SiGIC, foram identificadas oportunidades de pesquisa relacionadas ao desenvolvimento de aplicações Web contemporâneas, mais especificamente no que tange à estruturação dos requisitos funcionais e garantia da qualidade da aplicação. Desta forma, este capítulo apresenta uma abordagem para especificação de requisitos funcionais que propõe uma estrutura para análise, classificação e especificação desses requisitos alinhada aos conceitos explorados pelos métodos Web e que explora essa mesma estrutura para garantia da qualidade da especificação e do produto final.

A abordagem apresentada neste trabalho, e objeto desta tese, propõe uma especificação de requisitos funcionais apoiada por modelos e organizada de forma que seus elementos sejam descritos à luz das perspectivas de projeto Web (conceituação, navegação e apresentação) e das visões de modelagem (estrutural e comportamental) permitindo, assim, que esses requisitos sejam explorados de forma consistente desde a fase inicial do projeto, ou seja, a especificação. A preocupação com essa consistência no tratamento dos requisitos se justifica porque os métodos de desenvolvimento Web contemporâneos, propostos na literatura técnica, exploram modelos que se propõem a representar essas mesmas dimensões.

Esta abordagem propõe, também, a exploração de modelos conceituais para representação das entidades relacionadas ao domínio do problema, juntamente com sua estrutura e comportamento, e modelos de casos de uso para especificação do comportamento esperado do sistema do ponto de vista dos seus usuários. Para

modelagem dos casos de uso a abordagem adota um metamodelo chamado UCMModel, definido no escopo deste trabalho. O objetivo desse metamodelo é permitir a estruturação da especificação dos casos de uso segundo um conjunto de critérios e restrições bem definido e oferecer um arcabouço sintático e semântico para apoiar o controle da qualidade da especificação e do produto final.

Assim, a estrutura que apóia a análise, classificação e especificação dos requisitos funcionais oferece um arcabouço acoplado conceitualmente ao arcabouço de modelagem utilizado pelos métodos Web contemporâneos, e que pode ser explorado em conjunto com tais métodos, suprimindo-os com um tratamento consistente dos requisitos e com o controle de qualidade da especificação e do produto final.

É possível argumentar que, nesse contexto, a abordagem proposta nesta tese teria um espectro mais amplo, podendo também ser utilizada na especificação de aplicações convencionais ou não Web. Certamente, várias características da abordagem de especificação aqui apresentada transcendem os limites tecnológicos e/ou arquiteturais impostos pelas aplicações Web porque tratam de questões que lidam com um nível mais alto de abstração. Entretanto, é importante ressaltar que os elementos que compõem a abordagem sugerem que os requisitos sejam tratados de acordo com as mesmas perspectivas de projeto (conceituação, navegação e apresentação) propostas pelos métodos de desenvolvimento Web. Esse tratamento tem dois objetivos nesta pesquisa: 1) permitir que os requisitos funcionais sejam tratados sob diferentes perspectivas, visando aprofundar o seu entendimento e detalhamento, e; 2) possibilitar a exploração de formas mais adequadas para especificação dos requisitos funcionais com o objetivo de apoiar a elaboração dos modelos de projeto preconizados pelos métodos de desenvolvimento Web. Embora, essas mesmas perspectivas possam ser relevantes em outros cenários de desenvolvimento, conforme já foi destacado no capítulo 1, a inspiração para a exploração das perspectivas de projeto Web e todo o contexto da pesquisa, desde a revisão sistemática de CONTE *et al.* (2005), passando pelos estudos de observação preliminares (capítulo 2) e culminando com a revisão da literatura técnica (capítulo 3), está baseado no desenvolvimento de aplicações Web.

A seção 4.2 apresenta uma visão geral da abordagem e seu escopo de atuação. A seção 4.3 apresenta o processo adotado para engenharia dos requisitos. As seções 4.4 a 4.7 detalham a abordagem, focando desde a classificação dos requisitos até a geração de casos e procedimentos de testes funcionais. Finalmente, a seção 4.8 apresenta a conclusão, resumindo o diferencial da abordagem de especificação proposta em relação a outros trabalhos na mesma linha.

4.2 Visão Geral da Abordagem

Modelos têm sido usados em diversas áreas da Engenharia como uma forma de documentação das decisões de projeto, comunicação entre os membros da equipe e compartilhamento de informações para tomadas de decisão, entre outras (SWINGER e KOCH, 2003). A utilização de modelos nos permite abstrair e simplificar a representação de um problema ou solução através da utilização de elementos que capturem características particulares, criando, assim, um foco bem definido para o modelo. Esse foco é essencial, pois permite diminuir a complexidade do modelo e fornecer uma perspectiva de leitura e compreensão para o mesmo. Vários problemas da engenharia clássica demandam a representação da solução a partir de um conjunto de modelos que, por sua vez, visam representar as várias facetas da solução proposta (OGREN, 2000).

Na Engenharia de Software, mais especificamente, a multiplicidade de perspectivas vem sendo explorada há bastante tempo. Um exemplo clássico dessa multiplicidade é o enfoque na estrutura dos dados, capturado pelo modelo de Entidades e Relacionamentos (CHEN, 1976), e o enfoque comportamental, capturado pelo Diagrama de Fluxo de Dados (CONSTANTINE e YOURDON, 1979). Apesar de ortogonais, a exploração dessas duas perspectivas se mostrou adequada, tanto para a representação do domínio do problema quanto para representação da solução computacional. Além disso, a junção dessas duas visões (estrutural e comportamental) em um único conceito compõe a base da abordagem orientada a objetos, tão difundida e explorada nos dias atuais (SCHWINGER e KOCH, 2006).

As características particulares das aplicações Web no que diz respeito à exploração não linear da informação e integração de diferentes meios para apresentação das informações, levou os pesquisadores da Engenharia Web a propor a separação entre esses conceitos. A Figura 4-1 sintetiza essa separação e apresenta em seus três eixos as perspectivas de projeto Web (conceituação, apresentação e navegação), as visões de modelagem (comportamentais e estruturais) e as fases que representam o processo de desenvolvimento (especificação, projeto, dentre outras, independente de um modelo de processo específico). O que essa figura se propõe a representar é que os conceitos relacionados ao domínio do problema, navegação e apresentação (perspectivas de projeto Web) devem ser capturados tanto por modelos estruturais quanto comportamentais (visões da modelagem), criando, assim, um foco bem definido para leitura e interpretação desses modelos (SCHWINGER e KOCH, 2006). Adicionalmente, esses modelos evoluem durante o processo de desenvolvimento e apóiam a geração de outros artefatos ao longo do ciclo de vida do

sistema, caracterizando um cenário de Desenvolvimento Dirigido por Modelos (FRANCE e RUMPE, 2007).

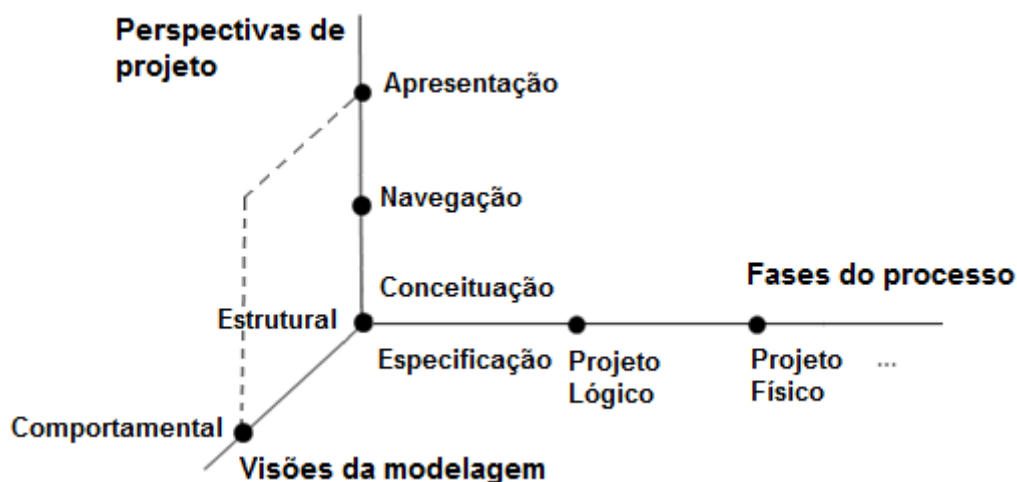


Figura 4-1 - Perspectivas de projeto x características dos modelos x fases de desenvolvimento (adaptada de SCHWINGER e KOCH (2006))

SCHWINGER e KOCH (2006) sugerem que tanto as perspectivas de projeto Web quanto as visões de modelagem sejam exploradas desde as fases iniciais do desenvolvimento. É possível incluir os requisitos funcionais nessa visão, buscando garantir consistência entre o tratamento dado aos requisitos e os diversos artefatos propostos pelos métodos Web gerados ao longo do processo de desenvolvimento.

Nas seções 4.2.1, 4.2.2. e 4.2.3, a seguir, serão apresentadas as definições das perspectivas de projeto Web e os artefatos definidos para representação da visão estrutural e comportamental em cada uma dessas perspectiva, segundo a abordagem proposta nesta pesquisa. Cada um desses artefatos e as informações que estes se propõem a representar serão detalhados ao longo da seção 4.5.

4.2.1 Perspectiva de Conceituação

A perspectiva de Conceituação trata da representação dos elementos conceituais que compõem o domínio do problema, abrangendo tanto as entidades que povoam esse domínio, com seus atributos e associações, quanto os eventos ou processos que exploram ou afetam essas entidades.

A Tabela 4-1 define os artefatos a serem construídos para representar as visões estrutural e comportamental da perspectiva de conceituação. Os detalhes de cada artefato serão apresentados nas seções seguintes.

Tabela 4-1 - Artefatos para representação dos requisitos na perspectiva de conceituação

Estrutural
(1) Entidades do domínio: modelo de classes
Comportamental
(2) Funcionalidades do sistema na visão do usuário: casos de uso
(3) Detalhamento do comportamento sistêmico: diagrama de atividades
(4) Processos: diagrama de atividades
(5) Eventos: máquinas de estado

4.2.2 Perspectiva de Navegação

A perspectiva de Navegação procura representar o espaço de exploração não-linear das informações a partir dos elementos conceituais que compõem o domínio do problema. Assim, essa perspectiva define **quais** caminhos podem ser explorados e **quais** informações (originadas do modelo conceitual) estarão acessíveis aos usuários em determinado ponto, sob a perspectiva de uma exploração não-linear dessas informações. Esse conceito tem sido explorado por diversos pesquisadores da Engenharia Web, conforme pode ser observado na literatura técnica, (SWABE *et al.*, 1996; HENNICKER e KOCH, 2000; PASTOR *et al.*, 2001; CACHERO e KOCH, 2002; CACHERO *et al.*, 2002, ESCALONA *et al.*, 2007), com pequenas variações de um método para outro.

A Tabela 4-2 lista os artefatos organizados para representar os aspectos estrutural e comportamental dessa perspectiva. Os artefatos sugeridos nessa tabela devem ser usados em pares obedecendo sempre a mesma numeração. Por exemplo, se a opção de representação da visão estrutural for **(1a)**, a opção para representação da visão comportamental deve ser **(1b)**, e assim por diante.

Tabela 4-2 - Artefatos para representação dos requisitos na perspectiva de navegação

Estrutural
(1a) Detalhamento da interação ator-sistema na descrição do caso de uso, ou seja, em linguagem natural
(2a) Modelos navegacionais usados nos métodos Web. Por exemplo: UWE – Modelo do Espaço Navegacional OOWS – Diagrama de Contexto Navegacional e Diagrama de Navegação
(3a) Protótipos da interface ou <i>mockups</i>
Comportamental
(1b) Regras em linguagem natural estruturada associadas à descrição do caso de uso
(2b) Restrições associadas ao modelos navegacionais
(3b) Pode ser representado no próprio protótipo através de anotações ou usar regras em linguagem natural estruturada associadas à descrição do caso de uso

4.2.3 Perspectiva de Apresentação

A perspectiva de Apresentação se propõe a representar características relativas aos aspectos visuais e à diagramação da interface com o usuário. Ou seja, essa perspectiva define **como** as informações acessíveis pelos usuários serão efetivamente apresentadas e as variações relacionadas a essa apresentação. Assim, da mesma forma que a perspectiva de conceituação apóia a definição da perspectiva de navegação, esta apóia a definição da perspectiva de apresentação.

A Tabela 4-3 lista os artefatos definidos para representar a estrutura e o comportamento relacionados à perspectiva de apresentação.

Tabela 4-3 - Artefatos para representação dos requisitos na perspectiva de apresentação

Estrutural
(1) Protótipos da interface
Comportamental
(2) Regras em linguagem natural estruturada associadas à descrição do caso de uso
(3) Regras em linguagem natural estruturada associadas ao protótipo

4.2.4 Linguagem de Modelagem

Com relação aos modelos adotados na fase de especificação, a UML (OMG, 2010a) foi escolhida como linguagem para modelagem desses artefatos. Essa escolha se deve a um conjunto de fatores:

- É um padrão de fato na indústria de software;
- Oferece uma variedade de modelos para capturar aspectos estruturais e comportamentais;
- Possui mecanismos de extensão (perfis UML e metamodelagem) que permitem a definição e exploração de novos elementos, não concebidos originalmente na linguagem;
- Não impõe nenhum método ou processo de desenvolvimento específico;
- É apoiada por uma grande quantidade de ferramentas *case*, algumas delas *open-source*.

4.3 Processo de Engenharia de Requisitos

O SWEBOK (*Software Engineering Body of Knowledge*) (SAWYER e KOTONYA, 2004), publicado também como ISO/IEC TR 19759:2005, sugere que o processo de Engenharia de Requisitos seja dividido em quatro atividades:

- **Elicitação de requisitos:** abrange a identificação de necessidades e a coleta de informações relacionadas ao sistema. Essas informações podem

ser capturadas de diversas fontes (usuários, documentos, sistemas legados, modelos e outros mais) usando diversas técnicas (entrevistas, *brainstorming*, observação, dentre outras) (ESCALONA e KOCH, 2004; SAWYER e KOTONYA, 2004) e constituem os requisitos da aplicação.

- **Análise de requisitos:** processo de análise das informações obtidas durante a elicitacão a fim de obter os requisitos do software. Pode envolver a classificacão dos requisitos elicitados, a definicão das fronteiras do sistema e a elaboracão de modelos conceituais relacionados ao domínio do problema;
- **Especificacão de requisitos:** produçã de um ou mais documentos, segundo uma abordagem ou técnica, que podem ser sistematicamente revisados, avaliados e aprovados. Dependendo da complexidade do produto a ser desenvolvido esses documentos podem conter a definicão do sistema, os requisitos do sistema e os requisitos do software, e;
- **Validacão de requisitos:** preocupa-se com o exame dos documentos de requisitos para averiguar se estes estã consistentes entre si, em concordância com os requisitos elicitados e se contemplam as expectativas e necessidades dos usuáios.

É possível observar na literatura técnica algumas variações a partir da definicão dessas quatro atividades. SOMMERVILLE (2006) propõe estas mesmas quatro atividades para a Engenharia de requisitos, porém com um arranjo ligeiramente diferente: as fases de elicitacão e análise sã fundidas como uma só atividade, denominada elicitacão e análise de requisitos. LAMSWEERDE (2009) também propõe essas quatro atividades, porém a atividade de análise de requisitos é substituída pela atividade de avaliaçã dos requisitos (resoluçã de conflitos e riscos) dividindo as tarefas relacionadas à análise de requisitos entre a avaliaçã e a especificacão dos requisitos.

A abordagem proposta neste trabalho não engloba questões relacionadas à elicitacão dos requisitos. Entretanto, o uso dessa abordagem demanda a produçã de um conjunto de artefatos relacionados tanto com a Análise quanto com a Especificacão de requisitos, pois a geraçã desses artefatos se inicia com a análise das informações coletadas na elicitacão e se completa com a obtençã dos documentos de especificacão propriamente ditos. Em virtude disso, as atividades de Análise e Especificacão de requisitos, da forma como sã definidas pelo SWEBOK, sã tratadas como uma única atividade, denominada Especificacão. Adicionalmente, as preocupações relativas à qualidade do produto levaram à inclusã de uma atividade

relacionada ao planejamento de casos e procedimentos de testes funcionais ao final das atividades de engenharia de requisitos. Assim, o processo organizado neste trabalho adota os conceitos preconizados pelo modelo V, que promove o planejamento antecipado dos testes.

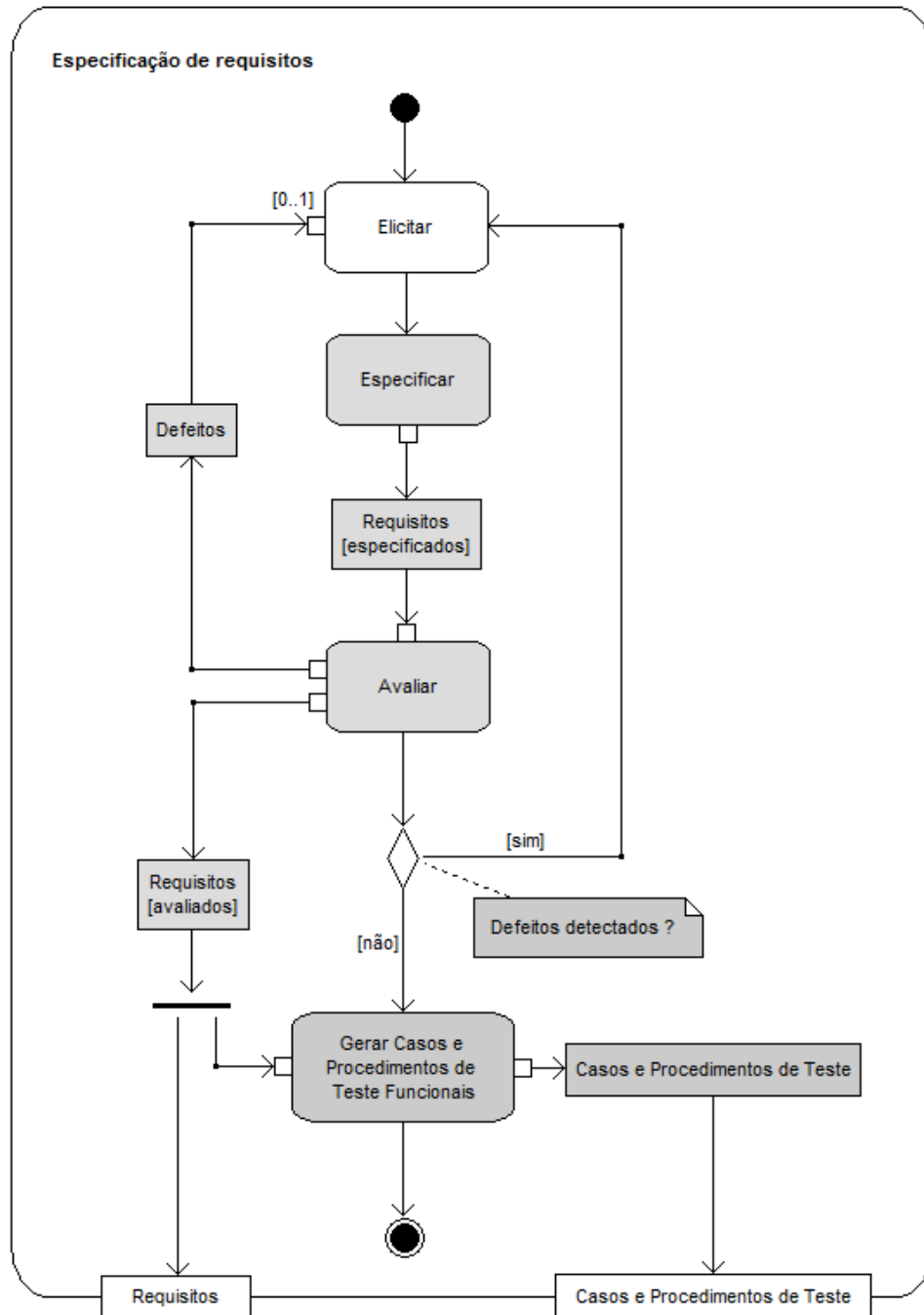


Figura 4-2 - Atividades previstas na abordagem proposta

O diagrama de atividades da Figura 4-2 apresenta uma visão geral do fluxo de atividades organizado para especificação dos requisitos, destacando, em cinza, as atividades e artefatos que fazem parte do escopo da abordagem proposta neste trabalho. Essas atividades podem ser definidas como:

- **Especificação de requisitos:** é a atividade que analisa, classifica, organiza e documenta os requisitos coletados. A técnica adotada para especificação baseia-se, principalmente, em modelos conceituais, para representar os aspectos estruturais e comportamentais relacionados ao domínio do problema, e em modelos de casos de uso para representar os aspectos comportamentais do sistema (JACOBSON *et al.*, 1992). A seção 4.5 discute com mais detalhes essa atividade e os seus artefatos;
- **Validação de requisitos:** é a atividade responsável por tentar garantir que não existem defeitos no documento de requisitos gerado na especificação. A seção 4.6 examina com mais detalhes essa atividade;
- **Geração de Casos e Procedimentos de Testes Funcionais:** é a atividade responsável por, a partir do documento de requisitos, gerar os casos e procedimentos de testes funcionais. Essa geração ocorre de forma semi-automática e a seção 4.7 apresenta mais detalhes acerca deste recurso.

4.4 Classificação de Requisitos

Tradicionalmente, os requisitos têm sido classificados como requisitos funcionais e não-funcionais. Enquanto os requisitos funcionais descrevem as funções que o software deve executar para solucionar determinado problema, os requisitos não-funcionais restringem o espaço da solução (SAWYER & KOTONYA, 2004; SOMMERVILLE, 2006).

A maioria dos métodos de desenvolvimento Web analisados no 3 propõe uma classificação para os requisitos. Embora não haja consenso sobre essa classificação e os termos empregados variem de um método para o outro, é possível observar uma classificação recorrente, principalmente nos métodos mais recentes ou que vem evoluindo ao longo do tempo, como OOHDM (SCHWABE *et al.*, 1996; VILAIN *et al.*, 2000; URBIETA *et al.*, 2007), WebML (CERI *et al.*, 2000; MORENO *et al.*, 2007), UWE (HENNICKER e KOCH, 2000; KOCH e KRAUS, 2002; KOCH *et al.*, 2006; PRECIADO *et al.*, 2008), W2000 (BARESI *et al.*, 2001; BARESI *et al.*, 2002; BARESI e MAINETTI, 2006) e NDT (ESCALONA *et al.*, 2003; ESCALONA *et al.*, 2007):

- Requisitos de dados: também chamados requisitos de armazenamento ou requisitos conceituais, determinam como a informação deverá ser estruturada.
- Requisitos de apresentação ou interação: determinam a estrutura e o comportamento da interface da aplicação durante a interação com o usuário.

- Requisitos de navegação: determinam as necessidades de exploração da informação.
- Requisitos de adaptação, personalização ou customização: descrevem como a aplicação deverá se adaptar dinamicamente ao usuário ou ao ambiente.
- Requisitos de transação: também chamados de requisitos de serviço, determinam como ocorrerá o processamento dos dados sem levar em conta os aspectos de interação com os usuários.

A abordagem proposta neste trabalho prevê a classificação dos requisitos elicitados conforme as perspectivas de projeto Web e as visões estrutural e comportamental da modelagem. Essa classificação visa definir um foco para o requisito e, a partir desse foco:

- Analisar e refatorar o requisito de acordo com esses conceitos;
- Permitir uma visão mais clara sobre quais modelos de projeto serão impactados pelo requisito;
- Fornecer um foco de leitura bem definido, minimizando ambigüidades na interpretação do requisito;
- Favorecer uma política de rastreabilidade desses requisitos pelos artefatos gerados, e;
- Determinar qual ou quais elementos da especificação são mais adequados para representação do requisito.

	Estrutural	Comportamental
Conceituação	Requisitos de dados	Requisitos de transação
Apresentação	Requisitos de apresentação	Requisitos de apresentação
Navegação	Requisitos de navegação	

Figura 4-3- Mapeamento entre a taxonomia de requisitos de ESCALONA e KOCH (2004) e as perspectivas de projeto Web e visões de modelagem

A Figura 4-3 apresenta o mapeamento entre a taxonomia usada por ESCALONA e KOCH (2004), as perspectivas de projeto Web e as visões estrutural/comportamental. Requisitos de dados são exclusivamente estruturais, enquanto requisitos de transação são exclusivamente comportamentais. Requisitos de navegação e apresentação podem ter visões estruturais ou comportamentais,

dependendo se estes tratam da estrutura ou de características dinâmicas dessas perspectivas. Os requisitos de adaptação, diferente dos demais, englobam a noção de variabilidade que pode estar associada a uma funcionalidade ou restrição de qualidade (QURESHI e PERINI, 2009). Assim, seu mapeamento pode ter desdobramentos nas diversas perspectivas e visões de modelagem.

Os requisitos não-funcionais não são tratados sob essa perspectiva porque eles podem impactar em mais de uma perspectiva de projeto ou visão da modelagem (da mesma forma que os requisitos de adaptação exemplificados anteriormente). Além disso, requisitos não-funcionais impactam diretamente em modelos arquiteturais que não fazem parte do escopo da abordagem proposta. Dessa forma, requisitos não-funcionais transcendem as perspectivas de projeto e as visões de modelagem, requerendo outras perspectivas para sua representação. Assim, apesar da sua importância para definição completa da aplicação a ser desenvolvida, é necessário um estudo mais detalhado sobre o impacto dos requisitos não-funcionais na abordagem proposta e quais alterações podem ser adotadas com o objetivo de adequar o arcabouço proposto às especificidades dos requisitos não-funcionais. Essa investigação representa um considerável esforço de pesquisa está fora do escopo deste trabalho.

4.5 Especificação de Requisitos

Por ser uma abordagem centrada no usuário serão adotados Casos de Uso (JACOBSON, 1992) para representar o comportamento do sistema no nível mais alto de abstração. Os elementos conceituais que permeiam o domínio do problema serão representados pelos Modelos Conceituais e, por fim, as políticas, procedimentos e restrições organizacionais serão capturados através de Regras de Negócio.

A abordagem apresentada neste trabalho defende que o documento de requisitos contenha um conjunto mínimo de informações que:

- Permita a descrição do conjunto de informações geradas pela abordagem de especificação proposta, e;
- Forme um conjunto coeso de informações acerca da especificação de requisitos.

O conjunto mínimo adotado representa, também, um subconjunto dos conceitos definidos tanto no padrão IEEE 830-1998 (IEEE, 1998) quanto no gabarito de especificação *Volere* (ROBERTSON e ROBERTSON, 2010). Assim, a organização adotada para o documento de requisitos possui as seguintes seções:

- **Propósito do projeto:** uma breve descrição da organização onde o projeto está inserido, da motivação que gerou a demanda pelo projeto e dos objetivos do projeto;
- **Glossário:** definição de termos, acrônimos e abreviações usadas na especificação dos requisitos;
- **Requisitos funcionais:** especificação de cada requisito funcional do usuário.
- **Requisitos não-funcionais:** especificação de cada requisito não-funcional, categorizados segundo a ISO/IEC 9126-1:2001 (ISO, 2001).
- **Modelos conceituais:** modelos criados para representar as diversas características dos conceitos relacionados ao domínio do problema.
- **Modelo de casos de uso:** modelos criados para capturar o comportamento esperado do sistema através de casos de uso, com a representação e detalhamento desses comportamentos através de diagramas de atividades.

Certamente outras seções poderão ser acrescentadas à estrutura desse documento para que seja possível especificar outras informações relevantes no contexto de um projeto específico. Todavia, este é o conjunto mínimo organizado para apoiar a especificação de requisitos funcionais de aplicações Web segundo a abordagem proposta. O detalhamento de cada um desses elementos é apresentado nas seções a seguir.

4.5.1 Modelos Conceituais

As informações coletadas na elicitação de requisitos podem ter várias fontes (usuários, documentos, sistemas legados e modelos, dentre outros) e estar representadas de diversas formas (textos em linguagem natural, planilhas eletrônicas, gravações de áudio de entrevistas, modelos de processos e estruturas, códigos-fonte ou outros artefatos). Os modelos conceituais na especificação de requisitos têm o objetivo de organizar e representar os conceitos contidos nessas fontes e que são tratados no escopo do sistema: entidades, atributos, associações, eventos, estados, processos, dentre outros (LAMSWEERDE, 2009). Esses conceitos são, em grande parte, entidades que compõem o domínio do problema, porém também devem ser consideradas as entidades relacionadas ao escopo do sistema, ou seja, entidades que não fazem parte do domínio do problema, mas são tratadas ou referenciadas nos requisitos do sistema.

A notação adotada para representar entidades conceituais e suas estruturas (atributos e associações) é o diagrama de classes da UML (OMG, 2010a), cujo enfoque estrutural se mostra adequado para representar esses conceitos.

Com relação aos eventos que podem ocorrer no escopo do sistema, estes devem ser representados em modelos comportamentais que capturem os efeitos da ocorrência desses eventos nos demais conceitos. Na UML, as máquinas de estado (OMG, 2010a) se apresentam como um mecanismo adequado para capturar essa dinâmica, pois representam de forma nativa os conceitos de eventos, transições e estados. Por outro lado, processos descritos nos requisitos podem ser representados através da seqüência de ações que os compõem, dos insumos e produtos dessas ações e dos responsáveis pelas mesmas. Nesse caso o diagrama de atividades da UML (OMG, 2010a) possui construtores capazes de representar os fluxos de controle e dados existentes nesses processos.

Por exemplo, o texto a seguir apresenta a dinâmica de um processo de promoção de funcionários no contexto de uma organização:

Para promover um funcionário o seu coordenador deve abrir um processo de solicitação de movimentação de pessoal junto ao departamento de RH. Essa solicitação será avaliada e pode ser indeferida, se não estiver de acordo com um conjunto de regras, ou enviada para apreciação da gerência sênior da respectiva área. Nessa segunda avaliação a solicitação pode ser reprovada ou aprovada em definitivo. Caso seja aprovada o setor de RH realiza as alterações necessárias no registro do funcionário e encerra a solicitação.

O comportamento descrito nesse exemplo pode ser representado sob a ótica do ciclo de vida da entidade tratada (solicitação de movimentação, nesse caso) e dos eventos que a afetam (Figura 4-4) ou sob a ótica das atividades e respectivos responsáveis pelo processo (Figura 4-5). A Figura 4-4 apresenta uma máquina de estado onde os possíveis estados da entidade são definidos e as transições representam os eventos e as condições necessárias para que haja uma mudança de estado. Por outro lado, a Figura 4-5 apresenta as atividades do processo modelado, com os respectivos responsáveis, e as informações que fluem por essas atividades com seus respectivos estados (destacados em cinza). Note que os processos e eventos capturados nesse momento podem transcender os limites de um caso de uso, ou seja, não existe necessariamente uma relação direta entre um processo descrito nesse nível de abstração e um caso de uso.

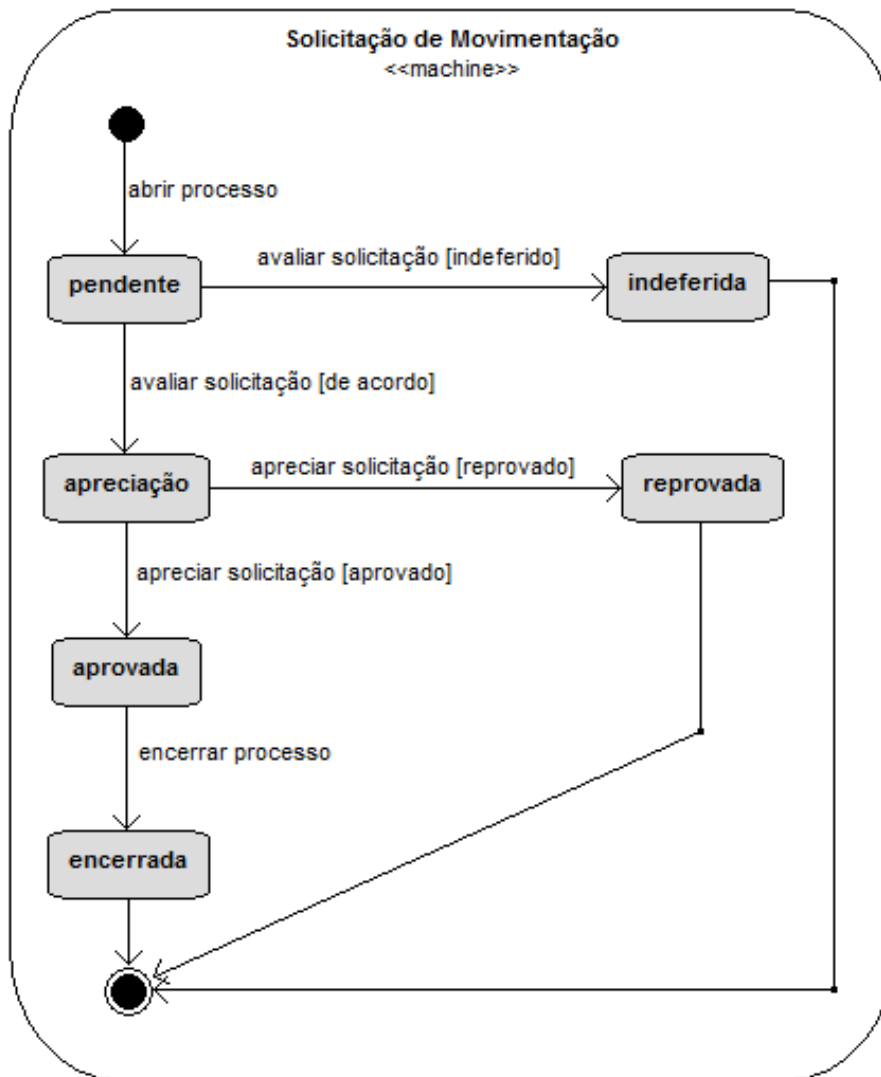


Figura 4-4 – Máquina de estado representando o ciclo de vida da entidade Solicitação de Movimentação

É importante ressaltar que os modelos conceituais devem:

- Capturar os conceitos independentemente das funcionalidades nos quais eles estejam envolvidos;
- Capturar os conceitos e suas características no mesmo nível de abstração dos requisitos;
- Limitar os conceitos modelados ao escopo do sistema, e;
- Ter uma estreita consistência entre si no que se refere aos conceitos representados e suas características.

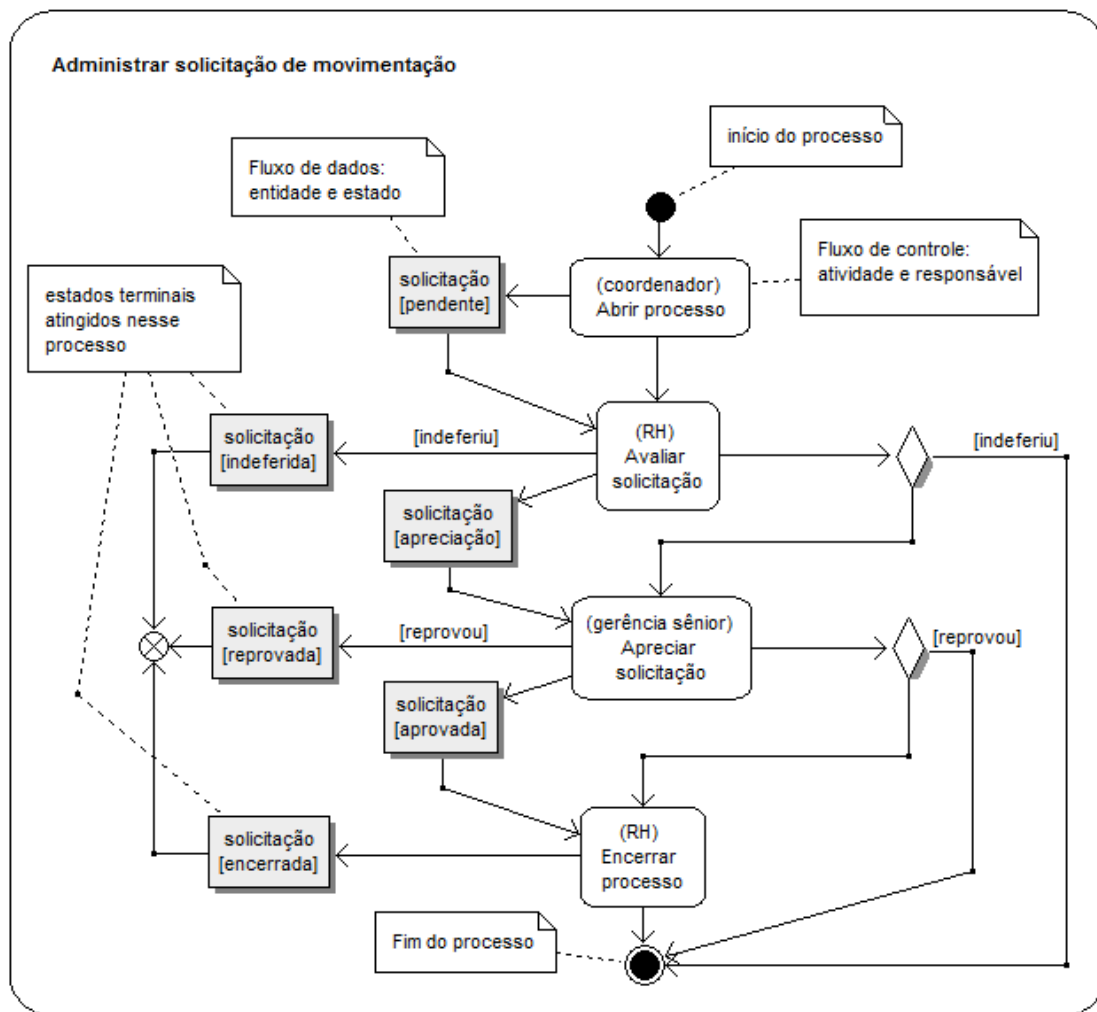


Figura 4-5 – Diagrama de atividades representando o processo de administração de solicitação de movimentação

4.5.2 Regras de Negócio

Regras de negócio são declarações que descrevem políticas, procedimentos ou restrições relacionadas a uma organização ou ao contexto onde essa organização atua e tem sido amplamente utilizadas na especificação de sistemas de informação (HALPIN, 2006; WANG *et al.*, 2010a). Entretanto, este limite pode ser estendido levando-se em consideração que não somente as organizações possuem um conjunto de regras específicas a serem observadas. Domínios de problema também possuem procedimentos ou restrições intrínsecos que precisam ser considerados.

O entendimento e a tradução corretos de regras relacionadas ao domínio ou negócio para o espaço da solução computacional é uma tarefa fundamental para produzir soluções que reflitam os procedimentos e restrições do ambiente nas quais estas estão inseridas. Assim, a especificação de tais regras em termos de elementos no espaço da solução computacional merece atenção especial no contexto geral da

especificação de requisitos funcionais. Com relação aos casos de uso, tais regras afetam diretamente o comportamento do sistema, restringindo-o ou detalhando-o.

Freqüentemente regras de negócio/domínio têm sido especificadas juntamente com requisitos funcionais de uma forma indistinta. Entretanto, algumas diferenças fundamentais separam esses dois conceitos, conforme exemplificado na Tabela 4-4:

Tabela 4-4 – Algumas diferenças entre Requisitos Funcionais e Regras de Negócio

Requisitos Funcionais	Regras de Negócio
Descrevem as funcionalidades ou capacidades do sistema	Descrevem políticas, procedimentos ou restrições organizacionais
Definem o comportamento do sistema	Definem o comportamento da organização
Estão limitados ao escopo do sistema	Transcendem o escopo do sistema
Devem ser garantidos pelo sistema	Podem ser garantidas sem a necessidade de sistema algum
Podem ser representados como casos de uso	Podem restringir o comportamento descrito nos casos de uso

A separação desses conceitos e a visualização clara do conjunto de regras de negócio envolvido podem trazer algumas vantagens:

- O trabalho de WANG *et al.* (2010b) apresenta evidências de que as demandas por manutenções estão, geralmente, relacionados à volatilidade das regras de negócio. Logo, o seu gerenciamento é essencial para rastrear os impactos das solicitações de alteração;
- As regras de negócio desempenham papel central na criação de testes funcionais, pois as alternativas do comportamento sistêmico podem estar associadas à políticas ou restrições definidas por tais regras, e;
- A reutilização das regras de negócio torna-se independente de um ou outro sistema.

Recentemente a OMG definiu uma nova especificação especialmente voltada à modelagem de regras de negócio chamada *Semantics of Business Vocabulary and Rule* (SBVR) (OMG, 2008) com o objetivo de prover uma base para a descrição detalhada das mesmas através de uma linguagem natural estruturada.

A SBVR distingue regras usando duas modalidades lógicas (HALPIN, 2006):

- Alética: regras que não deveriam, a princípio, ser violadas sob nenhuma hipótese, sendo portanto chamadas de regras estruturais, e;
- Deôntica: regras que expressam obrigações ou expectativas com relação ao comportamento, mas sem nenhuma garantia de que elas não serão violadas.

Baseado nessas modalidades, o gabarito de descrição de regras, definido no SBVR, defende que a descrição das regras siga um estilo prefixado, conforme apresentado na Tabela 4-5. O padrão prevê que, caso seja omitido, o estilo adotado é o “É necessário que...”.

Tabela 4-5 - Sugestão para prefixação de regras no SBVR (OMG, 2008; BOLLEM, 2008)

Modalidade	Estilo prefixado	Estilo coloquial
Estrutural		
Necessidade	É necessário que sempre ...
Impossibilidade	É impossível que nunca ...
Possibilidade restrita	É possível que às vezes ...
Comportamental		
Obrigatoriedade	É obrigatório que tem que ...
Proibição	É proibido que não deve ...
Permissão restrita	É permitido que pode ...

Segundo a SBVR, os princípios para descrição de regras são (LINEHAN, 2008; BOLLEM, 2008):

- Regras devem ser descritas, sempre que possível, usando lógica de primeira ordem;
- Regras são compostas de tipos de fatos (*fact types*);
- Tipos de fatos são compostos de termos, ou seja, definições conceituais, e;
- Uma regra é estrutural ou comportamental.

Seguindo esses princípios, regras devem ser organizadas de forma a referenciar de forma direta os conceitos do domínio/negócio no qual estão inseridas e devem expressar ou características/restrições estruturais ou comportamentos esperados. A abordagem proposta neste trabalho adota estes princípios e propõe que as regras de negócio/domínio também sejam analisadas e classificadas de acordo com as perspectivas de projeto Web, mantendo assim consistência com os conceitos da Figura 4-1. Essa classificação adicional tem por objetivo:

- Oferecer uma perspectiva adicional para escrita, leitura, compreensão e avaliação dessas regras, minimizando a sobrecarga semântica (cada regra está relacionada a uma e somente uma perspectiva de projeto);
- Possibilitar a especificação das regras usando o formalismo e vocabulário mais adequado de acordo com a perspectiva na qual ela está inserida, e;
- Evitar regras declarativas com comportamento complexo.

4.5.2.1 Regras x Perspectiva de Conceituação

Regras associadas à perspectiva de conceituação relatam proposições acerca dos conceitos relacionados ao escopo do sistema, ou seja, os mesmo conceitos que são capturados e representados pelos modelos conceituais.

A Tabela 4-6 exemplifica a aplicação dos princípios da SBVR para especificação de regras na perspectiva conceitual no contexto de um sistema de administração de unidades de ensino.

Tabela 4-6 - Exemplo de especificação de regras conceituais segundo a SBVR

Termos:	professor, departamento, coordenador, licença, solicitação de licença, disciplina, documentação adicional, em andamento, aprovada, reprovada
Tipos de fatos:	professor está vinculado a departamento departamento tem coordenador coordenador é um professor professor pode solicitar licença coordenador recebe gratificação professor leciona disciplina departamento oferece disciplina solicitação de licença requer documentação adicional solicitação de licença pode assumir os estados “em avaliação”, “aprovada” ou “reprovada”
Regras estruturais:	R1 – Um professor está vinculado a somente um departamento R2 – Um departamento tem somente um coordenador R3 – É necessário que um coordenador seja sempre um professor R4 – É impossível que um departamento tenha um coordenador que não seja um professor vinculado ao próprio departamento R5 – É possível que um departamento ofereça várias disciplinas R6 – Uma solicitação de licença deve estar associada a somente um estado
Regras comportamentais:	R7 – É proibido que um coordenador receba gratificação se estiver licenciado R8 – É proibido que um professor seja coordenador se ele lecionar mais de duas disciplinas R9 – É proibido que um professor solicite licença para outro professor R10 – É obrigatório que o estado de uma licença seja “em

avaliação” quando o professor solicitar a licença

R11 – É obrigatório que o estado da solicitação de uma licença seja “aprovada” quando a documentação adicional estiver de acordo com o conjunto de normas para concessão de licença

R12 – É obrigatório que o estado de uma licença seja “reprovada” quando a documentação adicional não estiver de acordo com o conjunto de normas para concessão de licença

As regras R1 a R6 da Tabela 4-6 estão associadas à visão estrutural, devendo, portanto, serem representadas através de um modelo de classes (Tabela 4-1, página 61). Por outro lado as regras R10 a R12 da Tabela 4-6 referem-se ao comportamento esperado para o tratamento da entidade “licença”, sendo indicado o uso de uma máquina de estados ou diagrama de atividades (OMG, 2010a) para representação dessas regras em alternativa à representação em linguagem natural estruturada (Tabela 4-1, página 61).

Regras comportamentais também podem estabelecer restrições ou detalhar procedimentos acerca de processos ou funcionalidades do sistema. Regras desse tipo devem estar associadas aos contextos em que atuam para que seja possível interpretá-las e tratá-las com risco mínimo de má interpretação. Para as regras R7 a R12 (Tabela 4-6) os possíveis contextos são exemplificados na Tabela 4-7:

Tabela 4-7 - Regras comportamentais e possíveis contextos de atuação

Regra	Possíveis contextos
R7 – É proibido que um coordenador receba gratificação se estiver licenciado	<ul style="list-style-type: none">• Cálculo dos vencimentos dos professores• Cálculo dos vencimentos dos funcionários
R8 – É proibido que um professor seja coordenador se ele lecionar mais de duas disciplinas	<ul style="list-style-type: none">• Avaliação de candidatos à coordenação
R9 – É proibido que um professor solicite licença para outro professor R10 – É obrigatório que o estado de uma licença seja em avaliação quando o professor solicitar a licença	<ul style="list-style-type: none">• Solicitação de licença
R11 – É obrigatório que o estado de uma licença seja aprovada quando a documentação adicional estiver de acordo com o conjunto de normas para concessão de licença R12 – É obrigatório que o estado de uma licença seja reprovada quando a documentação adicional não estiver de acordo com o conjunto de normas para concessão de licença	<ul style="list-style-type: none">• Avaliação da solicitação de licença

Os contextos de atuação desse tipo de regra normalmente representam funcionalidades previstas para o sistema ou procedimentos/ações dentro de uma funcionalidade prevista. Como a abordagem proposta prevê que o ponto de partida para especificação do comportamento do sistema são os casos de uso, também é previsto que as regras que impactam nesse comportamento estejam diretamente associadas ao caso de uso, mais precisamente às ações sistêmicas presentes nesses casos de uso. O detalhamento dessa associação será apresentado na seção 4.5.3.2.

4.5.2.2 Regras x Perspectiva de Navegação

Regras associadas à perspectiva de navegação definem **quais** informações serão disponibilizadas e possíveis restrições sobre essa disponibilização. A Tabela 4-8 exemplifica a aplicação dos princípios da SBVR para especificação de regras na perspectiva de navegação de um sistema de administração para a área de RH.

Tabela 4-8 - Exemplo de especificação de regras de navegação segundo a SBVR

Termos:	funcionário, nome, CPF, data de nascimento, estado civil, nome do cônjuge, data de casamento, formulário de cadastro de funcionário, casado, solteiro, perfil, usuário, rh, financeiro, gerência
Tipos de fatos:	<p>formulário de cadastramento de funcionário possui nome, CPF, nome, data de nascimento, estado civil, nome do cônjuge e data de casamento</p> <p>estado civil pode ser casado ou solteiro</p> <p>usuário tem perfil</p> <p>perfil pode ser rh, financeiro ou gerência</p> <p>usuário pode cadastrar funcionário</p>
Regras estruturais:	R1 – O formulário de cadastramento de funcionário requer CPF, nome, data de nascimento, estado civil, nome do cônjuge e data de casamento
Regras comportamentais:	<p>R2 – É obrigatório que nome do cônjuge e data de casamento sejam informados se o estado civil for casado</p> <p>R3 – É proibido que o usuário acesse a funcionalidade de cadastramento de funcionário se este não tiver perfil RH</p>

As regras estruturais representam quais informações estarão acessíveis aos usuários e possíveis restrições a esse acesso. Sendo assim, essa descrição pode ser feita, em linguagem natural estruturada, como um detalhamento da interação entre o ator e o sistema na descrição do caso de uso (Tabela 4-2, página 61). Ou seja, ao

especificar as interações entre o ator e o sistema essas regras podem estar associadas à essas interações no sentido de detalhar as informações acessíveis.

Uma representação alternativa são os modelos navegacionais propostos por métodos Web da literatura técnica (Tabela 4-2, página 61). Uma possível alternativa à essa representação seria a adoção de modelos navegacionais específicos de determinado método Web para essa tarefa, como por exemplo o Diagrama de Contexto Navegacional e o Diagrama de Navegação do método OOWS (PASTOR *et al.* 2001) ou o Modelo de Espaço Navegacional do método UWE (KOCH & KRAUS, 2002). A representação através desses modelos pode tornar-se uma escolha natural, caso o método Web seja adotado no restante do processo de desenvolvimento. Contudo, é preciso levar em consideração as eventuais dificuldades que serão impostas aos usuários ao tentarem validar esses modelos.

É possível também optar pelo uso de protótipos da interface ou maquetes (*mockups*) para representar as regras estruturais da perspectiva de navegação (Tabela 4-2). Cabe, nesse caso, uma análise de custo-benefício porque, se por um lado protótipos de interface apresentam vantagens como facilitar a comunicação com os usuários, por outro podem existir desvantagens como custos adicionais referentes à sua construção (SAWYER e KOTONYA, 2004). Entretanto, essa alternativa deve ser fortemente considerada se a prototipação da interface já estiver sendo usada no processo de desenvolvimento como técnica de validação dos requisitos junto aos usuários. É preciso ressaltar ainda que, mesmo com o uso de protótipos, as regras comportamentais ainda devem ser especificadas usando linguagem natural estruturada, pois mesmo que os protótipos simulem a dinâmica dessas regras estas devem ficar explicitamente especificadas a fim de serem consideradas nas fases posteriores do projeto.

Por fim, as regras relacionadas à perspectiva de navegação, sejam elas estruturais ou comportamentais, também devem estar associadas à casos de uso, pois elas definem quais informações estarão disponíveis durante a interação ator-sistema. Os detalhes dessa associação serão apresentados na seção 4.5.3.2.

4.5.2.3 Regras x Perspectiva de Apresentação

Regras associadas à perspectiva de apresentação definem **como** as informações disponibilizadas serão apresentadas e possíveis restrições sobre essa apresentação. A Tabela 4-9 exemplifica a aplicação dos princípios da SBVR para especificação de regras na perspectiva de apresentação.

Tabela 4-9 - Exemplo de especificação de regras de apresentação segundo a SBVR

Termos:	CPF, data de nascimento, formato de data, usuário, preferências
Tipos de fatos:	usuário tem preferências preferências define formato de data CPF pode ter formato
Regras estruturais:	R1 – O CPF deve ser apresentado no formato 999.999.999-99
Regras comportamentais:	R2 – É obrigatório que a data de nascimento seja apresentada no formato de data definido nas preferências do usuário

As regras estruturais e comportamentais da perspectiva de apresentação devem ser representadas em linguagem natural estruturada. Contudo, existem alguns tipos de regra onde essa representação não é recomendada, como por exemplo:

- Regras que definem diagramação;
- Regras que definem estilos relacionados ao *look-and-feel* da interface, e;
- Regras que definem os elementos da interface gráfica a serem utilizados.

Para esses casos sugere-se a adoção de um protótipo da interface onde essas questões podem ser representadas de forma mais adequada. Nesse caso, as mesmas observações sobre o uso de protótipos de interface na representação de regras de navegação são válidas para o uso de protótipo nas regras de apresentação.

Semelhante ao que ocorre com as regras da perspectiva de navegação, as regras da perspectiva de apresentação também devem estar associadas aos casos de uso, pois elas definem como as informações serão apresentadas durante a interação ator-sistema. A seção 4.5.3.2 discute esse tópico com maiores detalhes.

Com relação à transcrição das regras capturadas na elicitação de requisitos para o documento de requisitos, é importante ressaltar que nem todas as regras identificadas podem ser diretamente transcritas para o documento de especificação. A interpretação das regras originais segundo as perspectivas de projeto Web pode levar à refatoração destas e o seu conseqüente desdobramento de acordo com as perspectivas de projeto que elas impactam. Por fim, a abordagem aqui proposta prevê que as regras de negócio/domínio estejam diretamente associadas aos casos de uso descritos no documento de especificação, ou seja, a descrição dos casos de uso deve fazer referência direta às regras porque elas impactam no comportamento descrito no caso de uso, sendo, portanto, decisivas para definir e especificar de forma precisa o comportamento esperado para o mesmo.

4.5.3 Modelo de Casos de Uso

4.5.3.1 Casos de Uso

Requisitos funcionais podem ser especificados em diferentes níveis de detalhamento e formalização, usando abordagens que variam desde descrições em linguagem natural até especificações formais (ESCALONA e KOCH, 2004). A natureza interativa da grande maioria das aplicações Web favorece o uso de técnicas baseadas em cenários, tal como casos de uso (JACOBSON, 1992), para especificação dos aspectos comportamentais dessas aplicações. Além dos casos de uso representarem uma das abordagens mais usadas pela indústria de software, a ampla disseminação de conhecimento acerca dessa técnica a torna uma candidata natural para compor uma abordagem para especificação de requisitos de aplicações Web.

Casos de uso foram originalmente desenvolvidos por Ivar Jacobson como parte do Objectory, um processo de desenvolvimento de software orientado a objetos (JACOBSON, 1992). Desde então, casos de uso têm sido explorados em muitos outros processos (JACOBSON *et al.*, 1999; KRUCHTEN, 2004; ROSENBERG e STEPHENS, 2007) e, hoje em dia, eles não tem seu uso limitado às abordagens orientadas à objetos. Atualmente, casos de uso fazem parte da especificação da UML (OMG, 2010a), sendo definidos como “*a especificação de um conjunto de ações realizadas por um sistema que produz um resultado observável que é, normalmente, de valor para um ou mais atores ou outros stakeholders do sistema*”.

4.5.3.2 Descrição de Casos de Uso

A especificação da UML não define um padrão para descrição de casos de uso, embora sugira algumas opções:

1. Através de um tipo *Collaboration*, que representa a estrutura e o relacionamento de uma coleção de instâncias que realizam coletivamente uma funcionalidade desejada;
2. Através de pré e pós condições;
3. Através de linguagem natural;
4. Através de um tipo *Behavior*, o que permite a representação usando uma máquina de estados ou um diagrama de atividades, ou;
5. Através de uma combinação das opções anteriores.

O uso da linguagem natural para descrição de casos de uso é encorajado por profissionais da prática e por alguns métodos de desenvolvimento (JACOBSON *et al.*,

1999; COCKBURN, 2000; ARMOUR e MILLER, 2001; KRUCHTEN, 2004; ROSENBERG e STEPHENS, 2007). Entretanto, a especificação da UML não define se essa representação textual deve ser feita através de linguagem natural irrestrita ou estruturada, e, nesse último caso, em que formato. Nesse sentido, diversos trabalhos propõem gabaritos para a representação textual do caso de uso, como por exemplo:

- Trabalhos que tratam de descrições gerais de casos de uso frente a outras técnicas de Engenharia de Requisitos (LEFFINGWELL e WIDRIG, 2003; WIEGERS, 2003; ROBERTSON e ROBERTSON, 2006);
- Trabalhos que propõem métodos de desenvolvimento de software que empregam casos de uso (JACOBSON *et al.*, 1999; LARMAN, 2003; ARLOW e NEUSTADT, 2005; BOOCH *et al.*, 2007; ROSENBERG e STEPHENS, 2007);
- Trabalhos que detalham a técnicas de caso de uso (COCKBURN, 2000; ARMOUR e MILLER, 2001; SCHNEIDER e WINTERS, 2001; BITTNER e SPENCER, 2002; KULAK e GUINEY, 2003), ou;
- Trabalhos que propõem metamodelos de casos de uso (NAKATANI *et al.*, 2001; DURÁN *et al.*, 2004; SOMÉ, 2006; SOMÉ, 2008; GUTIÉRREZ, 2009).

Esses gabaritos visam definir um conjunto de elementos (mandatórios ou não) a partir dos quais os casos de uso devem ser descritos e a escolha desses elementos é influenciada pelo método empregado ou pelo ponto de vista que se deseja representar.

Apesar da grande variedade de gabaritos propostos na literatura técnica é possível observar algumas similaridades com relação aos elementos propostos, embora os nomes definidos para esses elementos difiram em muitos casos. Por exemplo, o conjunto de ações que formam o comportamento do caso de uso pode ser definido como: curso normal, caminho básico, fluxo principal, fluxo otimista, caminho alternativo ou fluxo alternativo, dentre outros. Em alguns casos a diferença está apenas no nome do elemento, mas em outros existe uma interpretação diferente para esses termos. Apesar dessa variação, é possível distinguir nos gabaritos adotados nesses trabalhos um conjunto de elementos comuns presentes em mais da metade dos trabalhos listados anteriormente: nome, descrição, ator, pré-condições, pós-condições, cenário principal, cenários secundários e requisitos especiais.

A abordagem proposta neste trabalho também adota um conjunto de elementos para descrição de casos de uso. Esse conjunto foi definido a partir da relevância desses elementos no contexto da abordagem proposta e está alinhado com o corpo de conhecimento existente na literatura técnica. A Tabela 4-10 apresenta os

elementos adotados para descrição de casos de uso e suas respectivas definições, comumente utilizados pelo grupo de Engenharia de Software Experimental da COPPE/UFRJ e que foram inspirados nos trabalhos de MUSA (1992), MEYER (2004) e JACOBSON (1992).

Tabela 4-10 - Gabarito de caso de uso adotado na abordagem proposta

Elemento	Descrição
ID	Identificador único do caso de uso
Nome	Nome do caso de uso
Descrição	Uma breve descrição sobre o objetivo do caso de uso
Ator(es)	Nome(s) do(s) ator(es) que participa(m) do caso de uso
Criticalidade	Um classificação, de acordo com uma escala ordinal, do quão crítico é esse caso de uso para a organização. Essa caracterização visa apoiar o gerenciamento das atividades relacionadas à garantia da qualidade, como priorização de inspeções ou aplicação de critérios de parada de inspeções e testes.
Frequência de uso	Um classificação, de acordo com uma escala ordinal, da frequência de utilização do caso de uso para a organização. Essa caracterização também visa apoiar o gerenciamento das atividades relacionadas à garantia da qualidade
Pré-condições	Uma descrição textual das condições que devem ser satisfeitas para que o caso de uso se inicie. Deve ser usada em casos de uso cuja execução não faz sentido em qualquer momento, mas somente quando o sistema está em determinado estado ou quando determinada condição for satisfeita.
Pós-condições	Uma descrição textual do estado que o sistema alcança ou do próprio resultado observável após o caso de uso ter sido executado com sucesso.
Trigger	Uma descrição textual dos eventos ou ações que iniciam o caso de uso.
Fluxo principal	Uma seqüência de passos numerados que descreve a seqüência de ações que serão executadas considerando que nada de errado acontecerá durante essa execução. Representa o caminho mais comum ou o cenário ideal para que o ator atinja o seu objetivo.
Fluxos alternativos	Uma seqüência de passos numerados que descreve o comportamento do sistema quando existe uma alternativa na execução do caso de uso ou quando determinado estado ou condição é alcançado. Representa um caminho alternativo no qual o ator pode ou não atingir o seu objetivo. Cada fluxo alternativo possui uma identificação única e deve estar associado à uma condição que define se este deve ou não ser executado.
Fluxos de exceção	Uma seqüência de passos numerados que descreve o comportamento do sistema quando algo inesperado ocorre. Representa um caminho onde o ator, normalmente, não atinge o seu objetivo. Cada fluxo de exceção também possui uma identificação única e deve estar associado a uma condição que define se este deve ou não ser executado.
Regras	São políticas, procedimentos ou restrições que devem ser levadas em consideração durante a execução do caso de uso. Devem estar sempre associadas aos passos dos fluxos onde devem ser observadas.
Passos	Representam ações a serem executadas no contexto do fluxo ao qual pertencem. Cada passo/ação pode estar associado a um ou mais fluxos alternativos, de exceção ou regras.

A Figura 4-6 exemplifica o uso do gabarito proposto pela abordagem. Esse exemplo, apesar de simples, ilustra o uso de todos os elementos existentes no gabarito.

Identificação:	UC001
Nome:	Autenticar usuário
Descrição:	Esse caso de uso permite que o usuário faça login na aplicação
Atores:	Internauta
Criticalidade:	Alta
Frequência de uso:	Média
Pré-condições:	O usuário não está autenticado
Pós-condições:	O menu da aplicação é apresentado de acordo com o perfil do usuário
Trigger:	Ator acessa qualquer página da aplicação
Fluxo Principal:	<ol style="list-style-type: none"> 1. Sistema solicita login e senha juntamente com uma opção “Esqueci a senha” 2. Internauta fornece login e senha [A1] 3. Sistema valida login e senha [R1, R2] 4. Sistema apresenta o menu da aplicação de acordo com o perfil do usuário [A2]
Fluxos Alternativos:	<p>[A1] Internauta seleciona “Esqueci a senha”</p> <ol style="list-style-type: none"> A1.1. Sistema solicita o login A1.2. Internauta fornece o login A1.3. Sistema valida o login [R1] A1.4. Sistema gera uma senha temporária [R3] A1.5. Sistema envia a senha temporária para o email do internauta [E1] A1.6. Sistema registra a nova senha A1.7. Sistema avisa para qual email a senha temporária foi enviada. <p>[A2] Login ou senha inválida</p> <ol style="list-style-type: none"> A2.1. Sistema avisa que o login ou a senha são inválidos A2.2. Vai para 1
Fluxos de Exceção:	<p>[E1] Erro no envio do email</p> <ol style="list-style-type: none"> E1.1. Sistema avisa que não foi capaz de enviar o email e pede que o usuário contacte o suporte.
Regras:	<p>R1 – login deve estar registrado no sistema como válido</p> <p>R2 – senha fornecida deve ser idêntica à senha associada ao login</p> <p>R3 – A senha temporária deve ser um número de 8 dígitos</p>

Figura 4-6 - Exemplo de uso do gabarito proposto pela abordagem

Através desse exemplo é possível observar que:

1. Não são previstas estruturas de controle do tipo **faça..enquanto** ou **se..então**. Por isso, o conjunto de ações a ser executado sob determinada condição deve ser definido como um fluxo alternativo ou de exceção
2. Cada um dos fluxos definidos no caso de uso (principal, alternativos e de exceção) são descritos separadamente e possuem sub-objetivos próprios, o

que oferece a possibilidade de serem tratados e detalhados de forma independente.

3. Fluxos alternativos e de exceção possuem uma identificação única e uma expressão de guarda que determina se estes devem ou não serem executados.
4. Para cada ação é possível definir os fluxos alternativos/exceção que podem ser ativados a partir da mesma. A ativação do fluxo em determinado ponto do caso de uso está associada à satisfação da expressão de guarda associada ao mesmo.
5. Regras definem restrições a serem observadas e estão sempre associadas às ações nas quais impactam.

A adoção de gabaritos para descrição dos casos de uso em um processo de desenvolvimento pode trazer alguns benefícios, como por exemplo:

- Padronização da especificação de casos de uso;
- Padronização da perspectiva de leitura e escrita da especificação de casos de uso através da definição dos elementos que o compõem;
- Possibilidade de verificar a aderência do documento ao gabarito proposto, e;
- Facilitar a comunicação entre membros da equipe e os demais interessados.

Entretanto, embora seja importante a definição de um padrão, a sua adoção não nos permite explorar cenários mais proeminentes, tais como:

- A organização de um apoio adequado à especificação dos casos de uso;
- A organização de um apoio (semi) automatizado à avaliação da qualidade dos casos de uso;
- Transformações modelo-modelo ou modelo-texto visando apoiar as fases subsequentes do desenvolvimento que usam as informações contidas no documento de especificação dos casos de uso.

O problema reside na pouca formalização da semântica dos elementos usados na descrição do caso de uso. A adoção de gabaritos para descrição de casos de uso, mesmo acompanhada de orientações para redação dos mesmos, não permite a organização de um apoio computacional que explore o processamento das informações existentes no caso de uso, limitando a sua utilização como simples veículos de comunicação entre os membros da equipe e os demais interessados (WILLIAMS *et al.*, 2005).

Visando aprimorar o grau de formalização das descrições dos casos de uso, um metamodelo para descrição de casos de uso, denominado UCMModel, foi definido no contexto deste trabalho. Este metamodelo estende o diagrama de atividades da UML (OMG, 2010a) e acrescenta novos elementos para descrição do comportamento do caso de uso. O capítulo 5 apresenta uma descrição completa do metamodelo UCMModel e de seus elementos.

4.5.3.3 Detalhamento do Comportamento

A especificação do comportamento de sistemas complexos, especialmente aqueles orientados a processos, requer que o desenvolvedor explore o uso de formalismos alternativos para sua representação, já que o nível de abstração oferecido pelos diálogos ator-sistema não é o mais adequado para o detalhamento de comportamentos sistêmicos. O que a abordagem apresentada neste trabalho propõe e o metamodelo UCMModel tenta garantir é que o nível de abstração da descrição dos casos de uso esteja mais próximo da visão do ator (o que o sistema deve fazer) do que da visão interna do sistema (como o sistema deve fazer). Entretanto, durante a especificação funcional, muitas vezes torna-se necessário o detalhamento de uma ação do sistema que apresenta-se atômica no nível de abstração do caso de uso, mas está associada a uma série de regras comportamentais que devem ser validadas pelos *stakeholders* e, posteriormente, observadas pelos desenvolvedores.

O diagrama de atividades da UML (OMG, 2010a) possui um recurso que apóia essa necessidade: a habilidade de detalhar uma ação com um diagrama de atividades, ou seja, o diagrama de atividades associado à ação têm o objetivo de detalhar o comportamento da mesma.

A Figura 4-7 apresenta a especificação do caso de uso “Concluir compra”, onde é possível observar um conjunto de regras associadas ao cálculo do pedido. Essas regras descrevem uma série de condições relacionadas ao desconto e cálculo do valor do frete que definem o comportamento esperado do sistema. Nesse caso, é possível optar por detalhar esse comportamento através de outro diagrama de atividades (Figura 4-8), buscando eliminar qualquer ambigüidade ou inconsistência em relação à descrição das regras em linguagem natural.

Na Figura 4-8 é possível observar as anotações indicando em quais pontos do diagrama de atividades as regras definidas no caso de uso são tratadas. Os fluxos de dados foram omitidos para facilitar a visualização e compreensão do modelo, porém é possível acompanhar esse fluxo através dos fluxos de controle e dos parâmetros de entrada e saída das ações. Cabe ressaltar também que, através do diagrama, é possível estabelecer de forma clara a ordem de aplicação das regras.

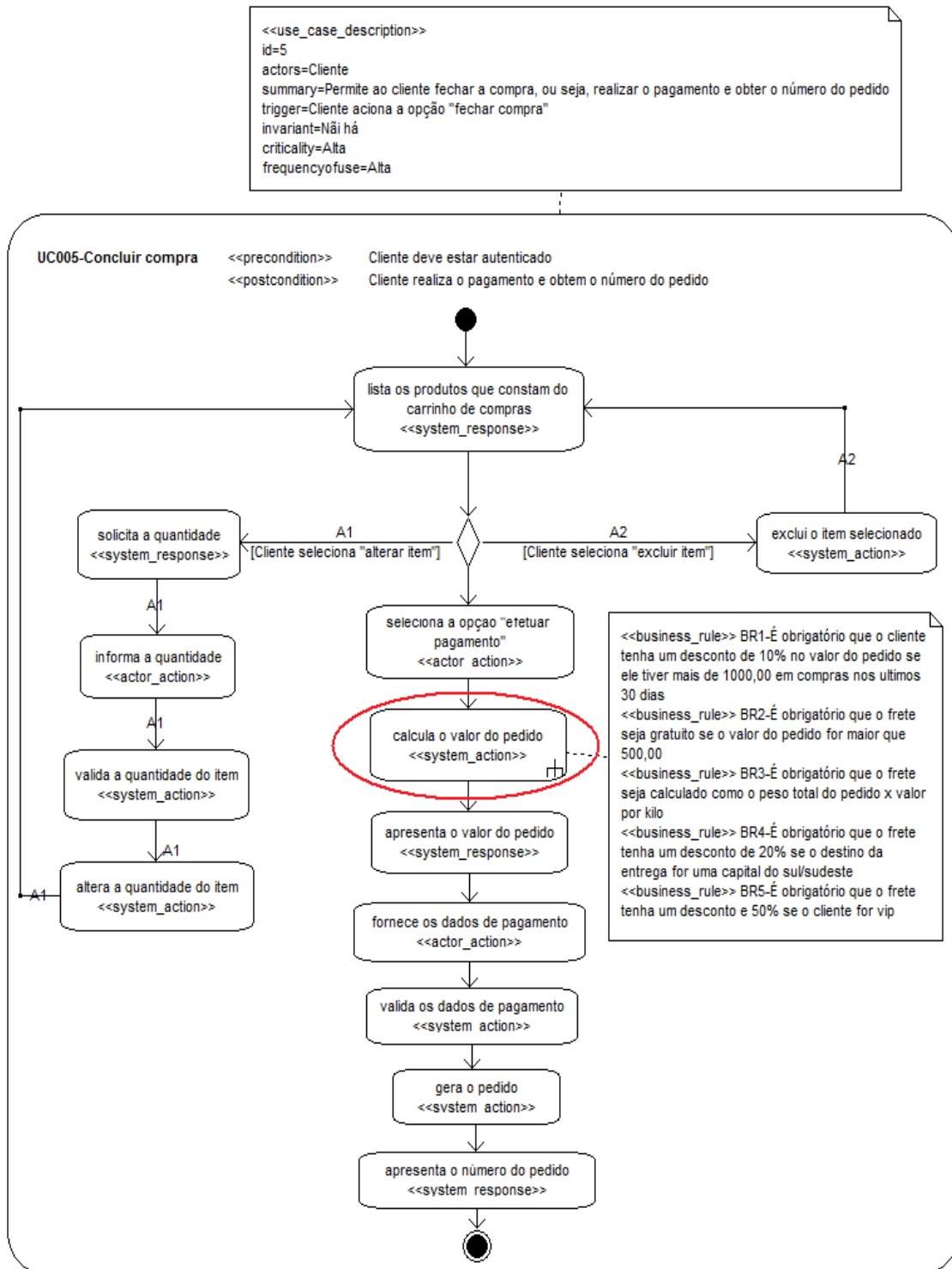


Figura 4-7 – Diagrama de atividades com a especificação do caso de uso “Concluir compra”

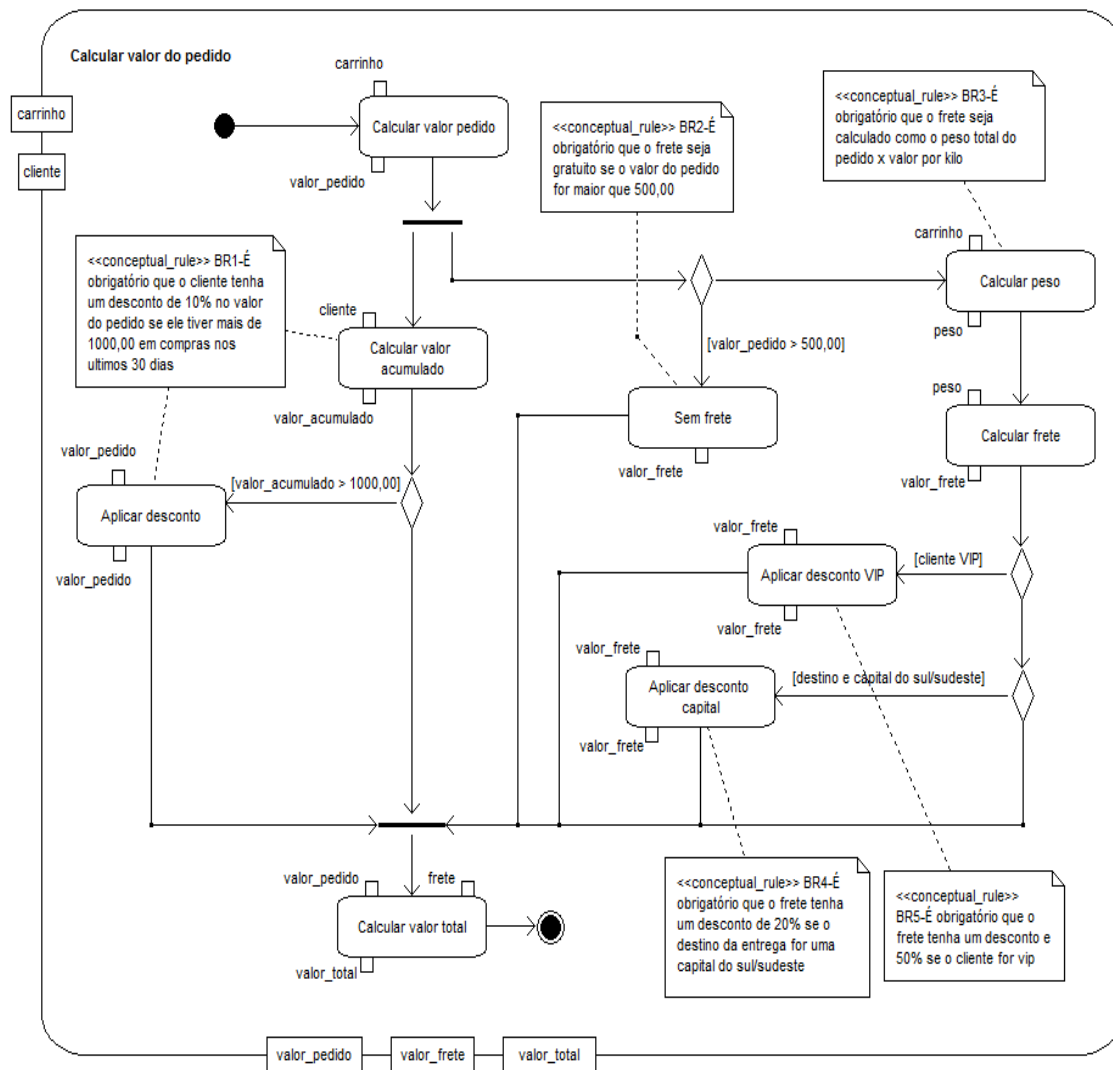


Figura 4-8 - Diagrama de atividades detalhando o comportamento da ação "Calcular valor do pedido" do caso de uso da Figura 4-7

Esse mecanismo possibilita a especificação dos comportamentos inerentes ao sistema em dois níveis de abstração:

- No primeiro nível (nível do caso de uso): ações do sistema descritas no nível de interação ator-sistema (o que o sistema faz), e;
- No segundo nível (nível do comportamento sistêmico): ações do sistema que descrevem o comportamento interno do mesmo (como o sistema faz).

É importante ressaltar que o segundo nível não se refere à implementação, mas sim ao refinamento do comportamento esperado que normalmente reflete algum procedimento relacionado ao domínio do problema.

Essa separação de abstrações reforça o nível em que deve ser descrito o caso de uso e oferece um mecanismo coerente para descrição do comportamento do

sistema em diversos níveis de abstração usando sempre o mesmo formalismo de representação (diagramas de atividades).

4.6 Inspeção de Requisitos

Uma forma de minimizar defeitos nas especificações de casos de uso é a aplicação de técnicas de avaliação estática para controlar a sua qualidade. Dentre essas técnicas, as de inspeção têm se mostrado um importante mecanismo de aprimoramento da qualidade do software (FAGAN, 1976; ACKERMAN *et al.*, 1989). Embora as técnicas de inspeção *ad-hoc* tenham o seu valor e sejam efetivas em muitos casos, aquelas com um foco bem definido (por exemplo, técnicas baseadas em *checklist* ou técnicas de leitura baseadas em perspectivas) tendem a produzir melhores resultados no que diz respeito ao número de defeitos detectados e/ou produtividade (defeitos detectados por unidade de tempo) (PORTER *et al.*, 1995).

Na literatura técnica é possível encontrar diferentes técnicas de inspeção que tratam documentos de requisitos. Algumas dessas técnicas são capazes de inspecionar o documento de requisitos independente da abordagem usada para a sua especificação (MAFRA e TRAVASSOS, 2006), enquanto outras foram especialmente projetadas para inspeção de casos de uso (ANDA *et al.*, 2001; COX *et al.*, 2004; BELGAMO e FABBRI, 2004; GREGOLIN e DEBONI, 2008) sendo a maioria dessas últimas baseadas em *checklist*. Mais especificamente, com respeito às questões apresentadas pelas técnicas baseadas em *checklist*, elas geralmente variam de questões puramente sintáticas (por exemplo, verificação da consistência da numeração dos passos do caso de uso ou verificação de definições ausentes) até questões semânticas de grande amplitude (por exemplo, verificação da consistência do nível de abstração do caso de uso ou verificação da clareza do objetivo do caso de uso).

A abordagem proposta neste trabalho sugere a adoção de uma combinação de questões a fim de maximizar a detecção de defeitos nos requisitos:

1. Questões gerais relacionadas ao gabarito adotado para descrição dos casos de uso;
2. Questões específicas relacionadas aos construtores usados na descrição dos casos de uso, e;
3. Detecção automática de defeitos sintáticos.

Para a cobertura dos itens 1 e 2, é possível aplicar a técnica apresentada por GREGOLIN e DEBONI (2008) Essa técnica foi definida a partir de um modelo de

qualidade com atributos de qualidade e regras para elaboração de diagramas e descrições de casos de uso. A partir desse modelo foram produzidos *checklists* para inspeção dos diagramas de caso de uso e das descrições de casos de uso. Uma das características dessa técnica é que as questões do *checklist* para inspeção das descrições de casos de uso estão alinhadas com o gabarito adotado neste trabalho, permitindo o uso da técnica sem nenhuma customização adicional. Além disso, essa técnica foi experimentalmente avaliada no contexto do SiGIC (SANTOS e TRAVASSOS, 2010), cuja especificação foi desenvolvida pelo grupo de Engenharia de Software Experimental da COPPE/UFRJ usando o gabarito de casos de uso aqui definido e adotado.

Com relação à questão 2, a técnica de inspeção ActCheck (MELLO *et al.*, 2010) foi desenvolvida para inspeção de diagramas de atividades no contexto da especificação de requisitos. Um diferencial dessa técnica é que ela possui, dentre as questões formuladas no *checklist*, um grupo de questões especialmente projetadas para detectar defeitos relacionados à especificação de casos de uso construídas a partir do metamodelo UCMoel. Essas questões foram elaboradas com base em um conjunto de orientações para descrição dos casos de uso, que foram definidas no escopo deste trabalho a partir da semântica dos elementos do UCMoel. A avaliação inicial da técnica ActCheck também foi realizada no contexto do SiGIC e os resultados desse estudo, bem como a evolução da técnica, estão reportados em (MELLO, 2011).

Por fim, com relação à questão 3, o capítulo 6 apresenta uma infra-estrutura de apoio que contém uma ferramenta especialmente projetada para apoiar a descrição dos casos de uso de acordo com o metamodelo UCMoel. No escopo dessa ferramenta, o metamodelo proposto e as regras que o regem são usadas como um oráculo para detecção de defeitos de não conformidade.

4.7 Geração de Casos e Procedimentos de Testes Funcionais

Levando-se em consideração que as atividades relacionadas aos testes do software estão entre as mais custosas do processo de desenvolvimento (JURISTO *et al.*, 2004), uma das condições necessárias para sucesso de projetos de software reside no planejamento cuidadoso dos testes. Esse planejamento é vital para que, dentre outros fatores, as funcionalidades identificadas nos documentos de requisitos do software sejam executadas sob todas as condições e regras definidas pelos requisitos, evitando-se assim parcialidade nos testes. Dentro do Plano de Testes, os Casos de Teste definem o conjunto de dados a serem aplicados e os resultados

esperados, enquanto os Procedimentos de Testes referenciam os Casos de Teste e definem os passos para a execução efetiva dos testes (MYERS, 2004).

No Teste Baseado em Modelos (MBT – Model Based Testing) (APFELBAUM e DOYLE, 1997), modelos são usados para descrever os vários elementos relacionados ao Teste de Software, dentre eles os casos e procedimentos de teste. Em relação aos testes funcionais, esses modelos são obtidos a partir das especificações funcionais do sistema.

Algumas abordagens na literatura exploram a geração dos casos e procedimentos de teste a partir de casos de uso e modelos de teste (HARTMANN *et al.*, 2005; GÓIS *et al.*, 2010). Nestes casos, porém, a criação dos modelos de teste fica sob inteira responsabilidade do Analista de Testes, pois estas abordagens não definem como construir estes modelos a partir da especificação dos requisitos.

A abordagem proposta neste trabalho usa como base a especificação dos casos de uso, estruturada de acordo com o UCMModel, e define um conjunto de regras de transformação que apóia a geração automática dos modelos preliminares de testes funcionais diretamente a partir dessa especificação.

Para construção do modelo de testes foi adotado, por oportunidade e conveniência, o metamodelo TDE (HARTMANN *et al.*, 2005). Esse metamodelo foi selecionado em virtude do conhecimento prévio de membros do grupo de Engenharia de Software Experimental (ESE) da UFRJ sobre esse modelo e da colaboração existente entre o grupo ESE e a Siemens Corporation Research/USA, onde esse metamodelo foi desenvolvido e é atualmente explorado em projetos reais de larga escala com o apoio de uma ferramenta proprietária denominada TDE/UML[®].

O processo de geração dos casos e procedimentos de testes funcionais a partir das especificações de casos de uso é realizado em 3 etapas distintas:

- 1) **Derivação do modelo preliminar de testes:** ocorre após a especificação e validação da especificação e produz um modelo preliminar de testes representado como um diagrama de atividades da UML2 (OMG, 2010a) e descrito de acordo com o metamodelo TDE. Este modelo é chamado de preliminar porque, neste momento, existem lacunas no modelo que correspondem aquelas informações relevantes no contexto do teste funcional, mas que não puderam ser extraídas diretamente da especificação de casos de uso devido a diferenças conceituais entre esses dois modelos.
- 2) **Edição do modelo de testes:** nesta atividade os modelos gerados na atividade anterior são editados e complementados com um conjunto de informações que são relevantes apenas no contexto de teste, como por

exemplo, a definição de classes de valores a serem aplicados durante a execução dos testes. A complementação do modelo de testes é necessária porque o modelo de casos de uso trata de conceitos abstratos, enquanto o modelo de testes lida com valores concretos associados a esses conceitos para que o comportamento do sistema possa ser avaliado a partir destes valores. Para auxiliar o analista de testes nessa tarefa, o modelo preliminar de testes é incrementado com uma série de comentários que procuram indicar quais regras são relevantes em determinado ponto do diagrama.

- 3) **Geração dos casos e procedimentos de teste:** a partir dos modelos de teste devidamente complementados, essa atividade se limita a percorrer o modelo de testes e gerar os caminhos possíveis de execução no diagrama, aplicando os valores definidos em cada ponto de decisão.

Os passos 1 e 3 desse processo são apoiados por uma ferramenta chamada ModelIT² (ALBUQUERQUE *et al.*, 2010), definida e construída no escopo deste trabalho, que será apresentada em detalhes no capítulo 6.

4.8 Conclusão

Neste capítulo foi apresentada uma visão detalhada da abordagem proposta e dos conceitos que a compõem, além dos documentos e artefatos envolvidos na produção da especificação de requisitos funcionais. Fazem parte do escopo dessa abordagem:

- 1) Um conjunto de atividades de especificação e inspeção de requisitos funcionais, alinhadas com o processo de engenharia de requisitos previsto no SEWBOK (SAWYER e KOTONYA, 2004), além de atividades para geração de testes funcionais a partir de descrições de casos de uso;
- 2) Um gabarito para o documento de requisitos, alinhado com o padrão IEEE 830-1998 (IEEE, 1998) e com o gabarito de especificação *Volere* (ROBERTSON e ROBERTSON, 2010) e, organizado para apoiar a especificação de requisitos funcionais de aplicações Web segundo a abordagem proposta;
- 3) Um gabarito para especificação de casos de uso, organizado a partir do corpo de conhecimento existente na literatura técnica, e que define os elementos para descrição do comportamento do sistema;
- 4) Estruturação dos requisitos funcionais a partir de:

- Um conjunto de modelos conceituais que capturam os aspectos estruturais e comportamentais relacionados ao domínio do problema;
 - Um conjunto de diagramas de atividades que formalizam o comportamento do sistema tanto no nível de abstração do caso de uso quanto no detalhamento das ações do sistema, e;
 - Um conjunto de regras que define e restringe o comportamento do sistema, classificadas de acordo com as perspectivas de projeto Web nas quais impactam.
- 5) O metamodelo UCModel, definido com o objetivo de estruturar a especificação dos casos de uso em torno de um conjunto de regras e restrições bem definido e prover um arcabouço sintático e semântico a partir do qual é possível explorar questões relacionadas à qualidade da especificação, e;
 - 6) Um conjunto de regras de transformação modelo-modelo que possibilitam a geração automática de um modelo preliminar de testes funcionais a partir da especificação de casos de uso, proporcionando ao analista de testes um ponto de partida consistente com a especificação a partir do qual os modelos definitivos de teste podem ser obtidos.

Assim, a abordagem apresentada como objeto desta tese, procura contemplar o espectro que engloba desde a análise e classificação dos requisitos funcionais, de forma alinhada à visão dos métodos Web contemporâneos, passando pela estruturação da especificação em torno de um conjunto de modelos que representam esses requisitos e formalizam o comportamento do sistema com o apoio de um metamodelo, chegando até o controle de qualidade da especificação (técnicas de inspeção) e do produto final (geração de casos e procedimentos de testes funcionais) através da exploração do arcabouço sintático e semântico que apóia a própria especificação dos requisitos. O tratamento dos requisitos a partir de uma visão horizontal, representada pela cobertura no tratamento dos requisitos desde a sua análise e classificação até o seu desdobramento em termos de garantia da qualidade do produto, aliada à uma visão vertical, representada pelo detalhamento e formalização em termos de modelos das características comportamentais do sistema, representam diferenciais desta abordagem em relação a outros trabalhos que também investigaram questões relacionadas à especificação de requisitos para aplicações de software (INSFRAN *et al.*, 2002; ESCALONA *et al.*, 2003; SOMÉ, 2006; ESCALONA *et al.*, 2007; GUTIÉRREZ *et al.*, 2008; SOMÉ, 2009).

No capítulo 5 será apresentado em detalhes o metamodelo UCModel que foi definido no contexto deste trabalho com o objetivo de estruturar a especificação de casos de uso através de um conjunto de critérios e restrições, que visam formalizar a descrição dos casos de uso através de um arcabouço com estrutura sintática e semântica bem definidas.

O capítulo 6 apresenta a infra-estrutura computacional organizada com o propósito de apoiar a abordagem de especificação e garantia da qualidade proposta nesta tese, além de detalhar as ferramentas UseCaseAgent e ModelT² que compõem essa infra-estrutura.

No capítulo 7 são apresentados dois estudos que foram planejados e conduzidos com o objetivo de avaliar a abordagem proposta neste trabalho, juntamente com o metamodelo UCModel e a ferramenta UseCaseAgent que apóia a atividade de especificação.

5 Metamodelo para Especificação de Casos de Uso

Neste capítulo é apresentado o metamodelo UCMModel, criado com o objetivo de apoiar a descrição de casos de uso segundo um conjunto de critérios e restrições bem definido, e que compõe a abordagem proposta nesta tese. São apresentados os princípios que nortearam a criação desse metamodelo, os elementos e as restrições que o compõem e os desdobramentos do seu uso em relação ao controle de qualidade da especificação

5.1 Introdução

A organização e adoção de um gabarito para descrição dos casos de uso representa o primeiro passo no sentido de padronizar essas descrições, pois ele oferece uma estrutura a partir da qual é possível capturar um conjunto de características relevantes do caso de uso, sob determinada perspectiva. Essa padronização oferece, ainda, um foco de leitura para os casos de uso baseado na definição dos componentes do gabarito, possibilitando que os membros da equipe e demais interessados no projeto tenham um entendimento comum sobre esses artefatos (COCKBURN, 2000; ARMOUR e MILLER, 2001).

Por outro lado, a natureza livre das descrições textuais dos elementos que compõem o caso de uso limita sua utilização à comunicação entre os *stakeholders*. Apesar dessa comunicação ser relevante para o sucesso de projetos de software, o frágil apoio semântico proporcionado pelas descrições textuais não permite a organização de uma infra-estrutura computacional capaz de (WILLIAMS *et al.*, 2005):

- Oferecer apoio adequado à especificação dos casos de uso, principalmente dos comportamentos que o compõem (fluxos principal, alternativos e de exceção);
- Apoiar um procedimento (semi) automatizado de avaliação da qualidade dos casos de uso, e;
- Viabilizar transformações modelo-modelo ou modelo-texto a partir das especificações dos casos de uso com o objetivo de apoiar as fases subsequentes do desenvolvimento.

Nesse sentido, neste capítulo é apresentado um metamodelo, denominado UCMModel, cujo objetivo é permitir a organização das especificações de casos de uso segundo um conjunto de critérios e restrições bem definido, e oferecer uma estrutura sintática e semântica a partir da qual seja possível:

- Avaliar as especificações dos casos de uso sintática (revisão com apoio ferramental) e semanticamente (inspeções), e;
- Apoiar a garantia da qualidade do produto final, através da geração semi-automática de casos e procedimentos de testes funcionais.

O metamodelo UCMModel foi definido como uma extensão do metamodelo da UML (OMG, 2010a), especializando parte dos conceitos relacionados ao diagrama de atividades para capturar tanto as informações básicas acerca dos casos de uso quanto os fluxos das ações que descrevem o seu comportamento.





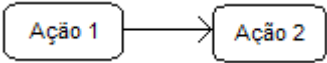

Este capítulo apresenta o metamodelo UCMModel e os princípios que orientaram a sua criação. São detalhados os elementos que compõem o UCMModel, as extensões do metamodelo da UML usadas para defini-los e as restrições que possibilitam a estruturação das descrições dos casos de uso segundo os critérios e princípios desse metamodelo.

5.2 Diagrama de Atividades e Casos de Uso

O Diagrama de Atividades é um dos diagramas comportamentais propostos pela UML 2 (OMG, 2010a). Diagramas de atividades descrevem comportamentos a partir de uma seqüência de ações, chamada de fluxo de controle. Os comportamentos representados podem estar em diversos níveis de abstração, sendo possível modelar desde processos de negócio e procedimentos relacionados à regras de negócio até algoritmos e operações que devem ser efetivamente implementadas em um sistema. A partir da especificação da UML 2.0, os diagramas de atividades passaram a trabalhar, também, com fluxos de dados, permitindo a modelagem dos objetos consumidos ou produzidos pelas ações do processo que está sendo modelado.

A construção de diagramas de atividades é realizada a partir de um conjunto de elementos que fazem parte do metamodelo da UML. Entretanto, os diagramas de atividades que compõem a abordagem proposta neste trabalho utilizam apenas uma parte desses elementos, mais precisamente os sete elementos definidos na Tabela 5-1:

Tabela 5-1 – Subconjunto de elementos do diagrama de atividades da UML explorados no contexto deste trabalho

Nome	Descrição	Símbolo
Nó inicial	Representa o ponto de partida para execução do diagrama	
Atividade	É o processo ou comportamento que está sendo modelado	
Nó de ação ou Ação	É um ação que compõe a seqüência de ações do processo ou comportamento que está sendo modelado	
Nó de decisão	Define um ponto a partir do qual existem diferentes alternativas de execução	
Fluxo de controle	É uma conexão direcionada entre duas ações que denota a seqüência de execução do diagrama	
Condição de guarda	É uma expressão associada a um fluxo de controle que define se este pode ou não executado	[expressão]
Nó final	Representa o ponto de parada da execução do diagrama	

Por outro lado, casos de uso podem ser definidos como especificações do comportamento esperado do sistema sob a ótica de quem interage com ele (ROSENBERG e STEPHENS, 2007), sendo tal comportamento descrito a partir de uma seqüência de ações que definem como essa interação ocorre. Assim, a flexibilidade para descrever processos do diagrama de atividades em diversos níveis de abstração e o alinhamento conceitual entre a seqüência de ações do diagrama de atividades e a seqüência de ações do caso de uso fizeram surgir algumas propostas de exploração deste modelo para especificação do comportamento de casos de uso (NAKATANI *et al.*, 2001; ALMENDROS-JIMÉNEZ e IRIBARNE, 2005; KOCH *et al.*, 2006; GUTIÉRREZ *et al.*, 2008). Entretanto, essas propostas se limitam a explorar o diagrama de atividades para descrever o comportamento do caso de uso utilizando somente os elementos originalmente propostos na UML, ou seja, sem estender a semântica dos elementos que compõem os diagramas de atividades. Esse cenário nos leva às seguintes restrições:

- A representação do caso de uso através de um diagrama de atividades é semanticamente equivalente à sua descrição textual, pois não há especialização de nenhum elemento do diagrama de atividades com o objetivo de alterar a semântica original;
- Os diagramas representam somente a descrição do comportamento do caso de uso, ou seja, a sua seqüência de ações. Os demais elementos que

compõem o caso de uso (descrição, atores, criticalidade, frequência de uso, *trigger*, pré e pós condições e regras) são ignorados, e;

- Mesmo com o foco voltado para a descrição do comportamento não é possível distinguir, no diagrama de atividades, o fluxo principal dos fluxos alternativos, pois o caso de uso é descrito como uma seqüência de ações e decisões sem contrapartida com os conceitos do caso de uso.

Com o objetivo de tratar essas questões, o metamodelo UCMModel propõe um conjunto de elementos e restrições que visam mapear todos os conceitos do caso de uso explorados pela abordagem de especificação proposta neste trabalho (Tabela 4-10 - página 81), de forma que o caso de uso possa ser completamente descrito usando o diagrama de atividades. Em complemento, o UCMModel procura explorar o conceito de transação, preconizado por JACOBSON (1992), com o objetivo de criar um conjunto especializado de ações a partir das quais os diversos fluxos do caso de uso possam ser descritos.

5.3 O Metamodelo UCMModel

De acordo com BÉZIVIN (2006), um metamodelo “descreve os diversos tipos de elementos contidos em um modelo e a forma como eles são arranjados, relacionados e restringidos”, ou seja, um metamodelo define a linguagem a ser usada por modelos que estejam em conformidade com ele. Entretanto, antes de definir concretamente quais elementos farão parte do metamodelo, é necessário definir quais informações serão capturadas e suas respectivas definições. No caso do metamodelo UCMModel o conjunto de informações acerca dos casos de uso a ser capturado se refere aquelas organizadas no gabarito de casos de uso apresentado na Tabela 4-10 (página 81).

A partir da definição do conjunto de informações a ser capturado foram analisados quais elementos do metamodelo da UML seriam candidatos à especialização a fim de apoiarem a representação dessas informações. Essa análise não foi realizada levando-se em conta todos os elementos do metamodelo da UML, mas somente aqueles usados na definição dos diagramas de atividades. A Figura 5-1 apresenta uma visão parcial do metamodelo da UML 2 contendo esses elementos destacando, em cinza, as metaclasses concretas.

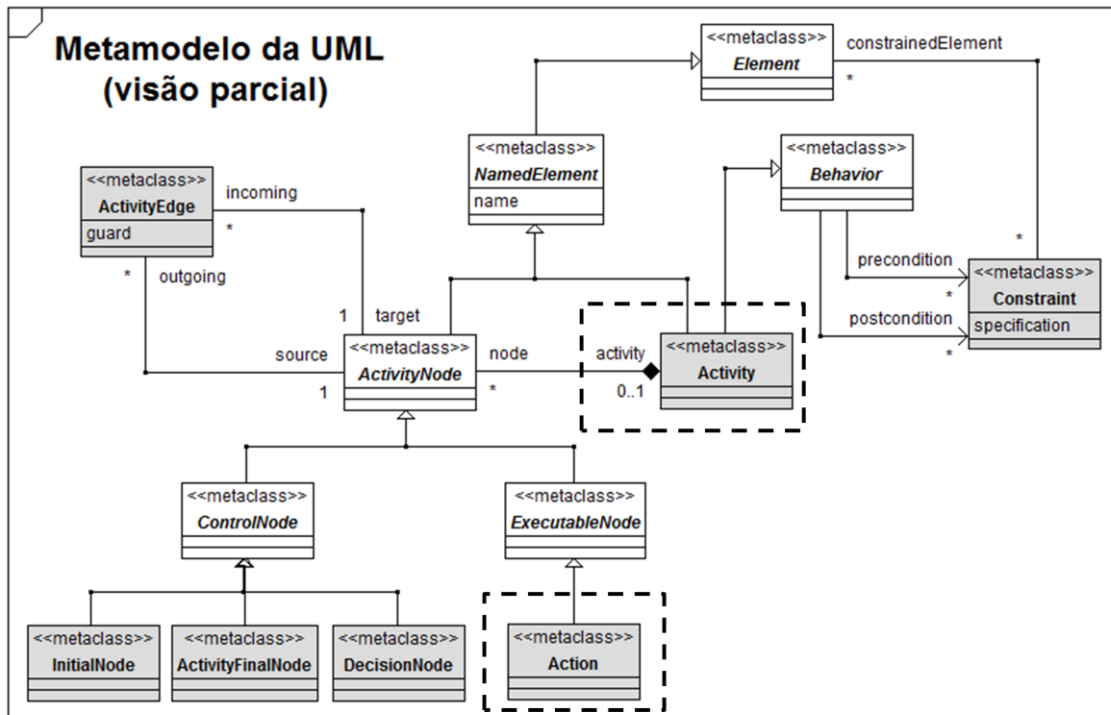


Figura 5-1- Visão parcial do metamodelo da UML 2 com os elementos do diagrama de atividades usados na definição do metamodelo UCMModel

Analisando os elementos da Figura 5-1 é possível traçar um paralelo entre os elementos **Activity/Action** (destacados por retângulos tracejados) e **Casos de Uso/Ações** do caso de uso. Assim como o **Caso de Uso** representa, em linhas gerais, um comportamento composto por uma seqüência de **Ações**, o elemento **Activity** representa um processo especificado por uma seqüência de elementos **Action**.

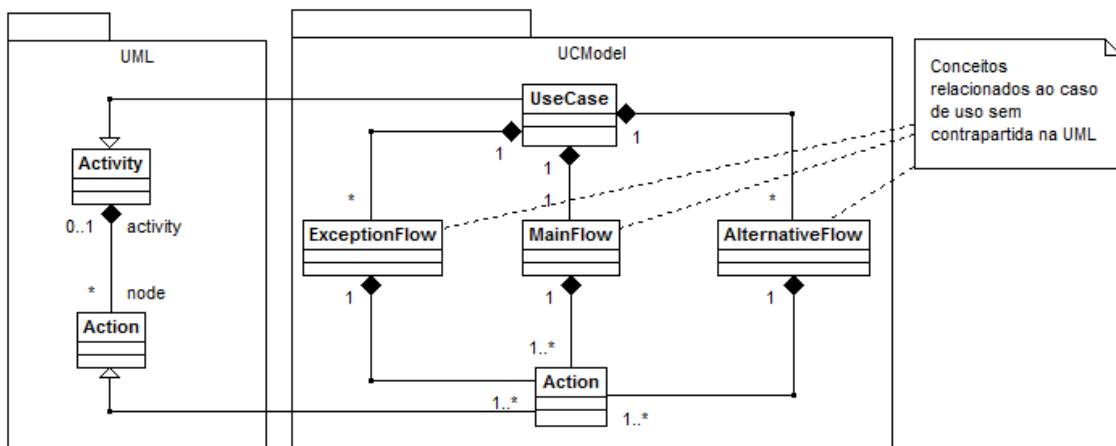


Figura 5-2 - Exemplo da análise realizada para definir quais elementos da UML seriam especializados para criação dos elementos da UCMModel

Entretanto, essa equiparação não é totalmente verdadeira porque as ações do caso de uso não estão diretamente relacionados a ele, mas sim aos fluxos principal, alternativos e de exceção que, por sua vez, pertencem ao caso de uso, conforme

apresentado na Figura 5-2. Nesse caso, houve a necessidade de acrescentar elementos ao metamodelo UCMModel que representassem esses três conceitos (fluxos principal, alternativos e de exceção) sem paralelo no metamodelo da UML.

Essa mesma análise foi realizada para cada um dos elementos que compõem o conjunto de informações do caso de uso, conforme o gabarito de casos de uso apresentado na Tabela 4-10 (página 81). O resultado final pode ser observado na Figura 5-3, que apresenta os elementos do metamodelo UCMModel e como estes se relacionam com o metamodelo da UML.

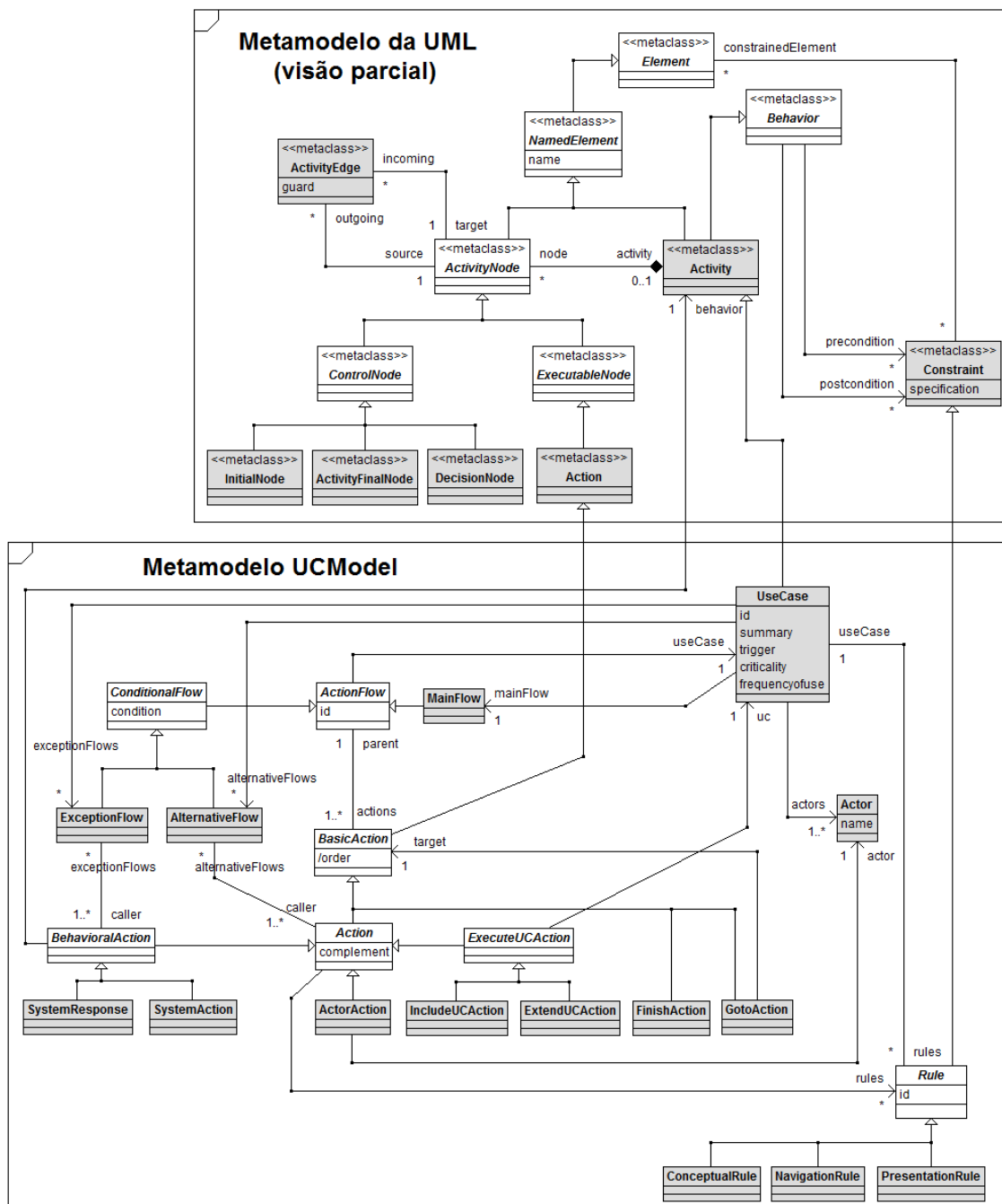


Figura 5-3 - Metamodelo UCMModel e seu relacionamento com o metamodelo da UML

As seções 5.3.1 a 0 detalham cada elemento do metamodelo UCMModel e apresentam os estereótipos e *tagged-values* definidos para construção de diagramas de atividades aderentes ao metamodelo proposto, além de exemplos de como tais elementos são representados nesses diagramas.

5.3.1 Caso de Uso e seus Elementos Básicos

A Tabela 5-2 apresenta o caso de uso e seus elementos básicos (conforme organizado na Tabela 4-10 - página 81) e como esses elementos são capturados pelo metamodelo UCMModel. A Figura 5-4 apresenta uma parte do metamodelo UCMModel (retirado da Figura 5-3) que destaca as metaclasses utilizadas para capturar o caso de uso e seus elementos básicos.

A primeira coluna da Tabela 5-2 define a informação a ser capturada, enquanto a segunda coluna indica o elemento do metamodelo UCMModel usado para representar essa informação. A terceira coluna da Tabela 5-2 indica em que parte do metamodelo UCMModel (Figura 5-4) está o elemento referenciado.

Tabela 5-2 - Mapeamento entre as informações básicas do caso de uso e os elementos do metamodelo UCMModel

Informação	Elemento do UCMModel	Figura 5-4
Caso de Uso	<i>UseCase</i>	1
ID	<i>UseCase.id</i>	1
Nome	<i>UseCase.name</i>	3
Descrição	<i>UseCase.summary</i>	1
Criticalidade	<i>UseCase.criticality</i>	1
Frequência de uso	<i>UseCase.frequencyofuse</i>	1
Ator(es)	<i>UseCase.actors</i>	2
<i>Trigger</i>	<i>UseCase.trigger</i>	1
Pré-condição	<i>UseCase.precondition</i>	4
Pós-condição	<i>UseCase.postcondition</i>	4

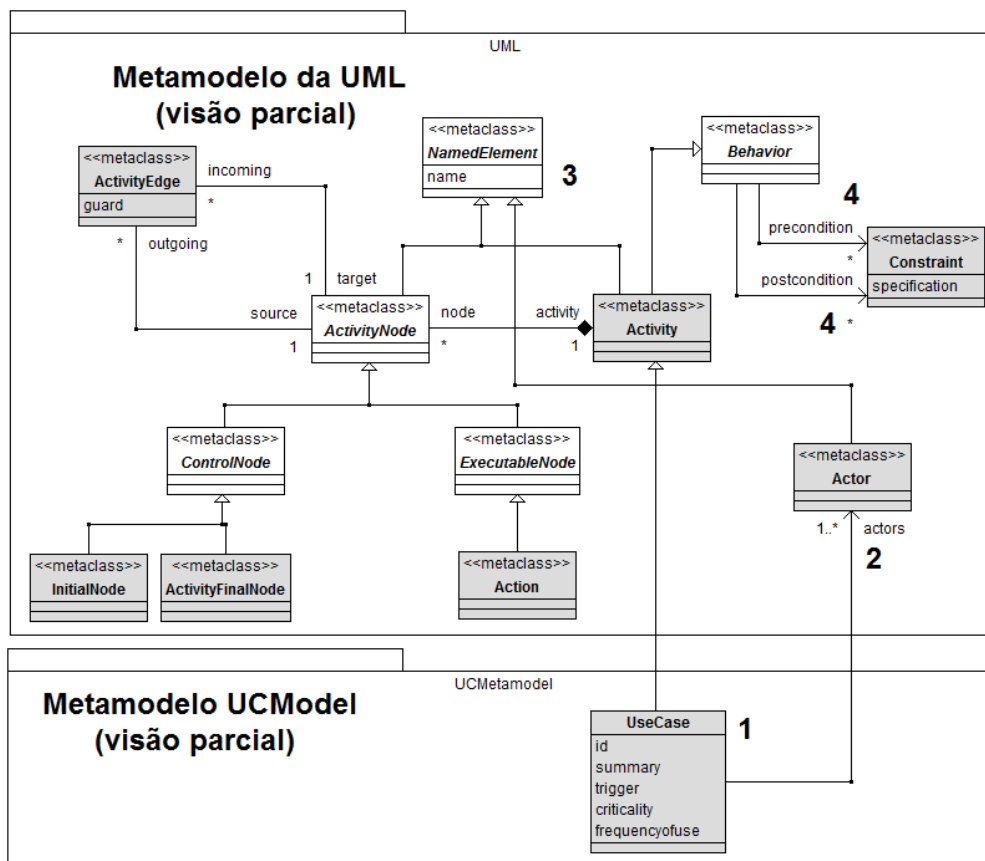


Figura 5-4 – Visão parcial do metamodelo UCMetamodel referente ao caso de uso e seus elementos básicos

Representação no Diagrama de Atividades

Com base no mapeamento da Tabela 5-2 foram definidos estereótipos e *tagged-values* para representar, no diagrama de atividades, o caso de uso e suas informações básicas. Neste caso, houve a necessidade de criar somente um estereótipo para estender a metaclasses *Activity* da UML na metaclasses *UseCase* do UCMetamodel (Tabela 5-3). Os atributos *precondition* e *postcondition* da metaclasses *UseCase* não precisam ser representados como *tagged-values* porque são mapeados diretamente para os mesmos atributos na classe *Activity* da UML.

Tabela 5-3 - Estereótipos usados na definição do caso de uso e seus elementos básicos

Estereótipo	Tipo-base	Definição
<<use_case_description>>	<i>Activity</i>	Indica que a atividade é uma descrição de caso de uso
	Tagged-values	
	<i>id</i>	Número único que identifica o caso de uso
	<i>summary</i>	Breve descrição do objetivo do caso de uso
	<i>actors</i>	Nomes dos atores separados por vírgula

	<i>trigger</i>	Expressão que indica o evento ou condição que dispara a execução do caso de uso
	<i>criticality</i>	Indica o nível de criticalidade do caso de uso (escala ordinal)
	<i>frequencyofuse</i>	Indica a frequência, em escala ordinal, com que o caso de uso é executado pelos atores

A Figura 5-5 apresenta um diagrama de atividades que descreve um caso de uso e seus elementos básicos. Os elementos que são mapeados diretamente para a UML são representados nativamente no diagrama de atividades (elementos 3 e 4 da Figura 5-4). Os demais elementos, por não terem contrapartida no metamodelo da UML, devem ser descritos como estereótipos ou *tagged-values*, que são apresentados no diagrama como comentários associados ao elemento redefinido pelo estereótipo. Assim, o comentário associado à atividade indica que esta é, na realidade, uma descrição de caso de uso (estereótipo `<<use_case_description>>`) e define os valores para as *tagged-values* *id*, *summary*, *actors*, *criticality* e *frequencyofuse*. Para facilitar a identificação do caso de uso para o leitor do modelo, o *id* do caso de uso também é representado no nome do mesmo no formato “UCxxx” onde “xxx” é o *id* do caso de uso. Além dos dez elementos apresentados na Tabela 5-2, o diagrama da Figura 5-5 apresenta também os nós inicial e final que marcam o início e o fim, respectivamente, da execução do caso de uso.

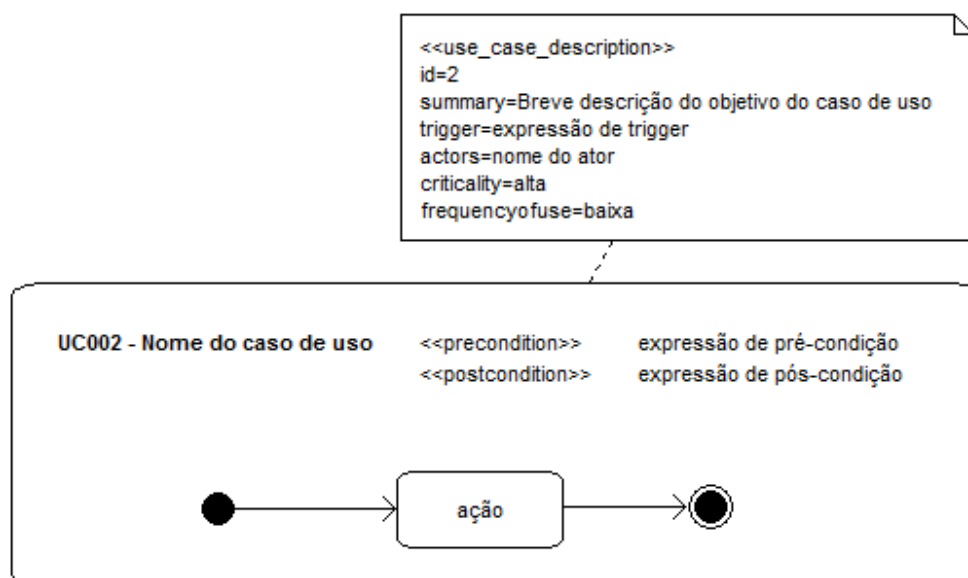


Figura 5-5 – Padrão de representação de um caso de uso e seus elementos básicos usando o diagrama de atividades da UML 2

5.3.2 Regras

Regras definem políticas, procedimentos ou restrições relacionadas a uma organização, ao contexto onde essa organização atua ou a um domínio de problema. Do ponto de vista da sua natureza, regras podem expressar restrições que não podem ser violadas sob nenhuma circunstância ou expressar expectativas com relação a um comportamento (HALPIN, 2006). Essa distinção leva à exploração de mecanismos diferentes para representar um ou outro tipo de regra, como por exemplo: linguagem natural estruturada, modelos de domínio que definem relações estruturais, diagramas de atividades que especificam processos, máquinas de estado que representam ciclos de vida com eventos e reações a esses eventos, dentre outros, conforme detalhado na seção 4.5.2 (página 71).

Na especificação do caso de uso estamos interessados em representar as regras que impactam no comportamento esperado para o mesmo ou na interface usada no diálogo ator-sistema. Com o objetivo de definir regras auto-contidas do ponto de vista da perspectiva em que atuam, a Tabela 5-4 descreve os três tipos de regras definidas no metamodelo UCMoel. A classificação dessas regras, inspirada nas perspectivas de projeto Web, tem o objetivo de oferecer um foco bem definido para tratamento dessas regras. É importante ressaltar que, segundo o UCMoel, regras pertencem ao caso de uso, mas devem estar sempre associadas às ações que afetam. Essa localização é importante porque delimita o escopo do impacto daquela regra no contexto do caso de uso.

Tabela 5-4 – Classificação da regras de negócio/domínio de acordo com as perspectivas de projeto Web

Tipo de Regra	Descrição	Perspectiva Web
<i>ConceptualRule</i>	Restringe as ações do sistema, definindo como esta deve ser realizada ou estabelecendo condições para sua execução	Conceituação
<i>NavigationRule</i>	Restrições sobre quais caminhos podem ser explorados e sobre quais informações serão apresentadas/solicitadas ao ator	Navegação
<i>PresentationRule</i>	Restrições sobre como as informações serão apresentadas/solicitadas ao ator	Apresentação

A Tabela 5-5 apresenta como as regras de negócio/domínio e suas associações são capturadas no metamodelo UCMoel. A Figura 5-6 apresenta uma parte do metamodelo UCMoel (retirado da Figura 5-3) destacando as metaclasses utilizadas para capturar as regras associadas ao caso de uso. Como elemento base para criação das regras no metamodelo UCMoel foi selecionada a metaclasses

Constraint do metamodelo da UML, por esta representar restrições ou condições que afetam o elemento ao qual estão associadas.

Tabela 5-5 - Mapeamento entre as regras e os elementos do metamodelo UCModel

Informação	Elemento do UCModel	Figura 5-6
Regras	<i>ConceptualRule</i> <i>NaviagtionRule</i> <i>PresentationRule</i>	1
Associação de regras com casos de uso	<i>UseCase.rules</i>	2
Associação de regras com ações	<i>Action.Rules</i>	3

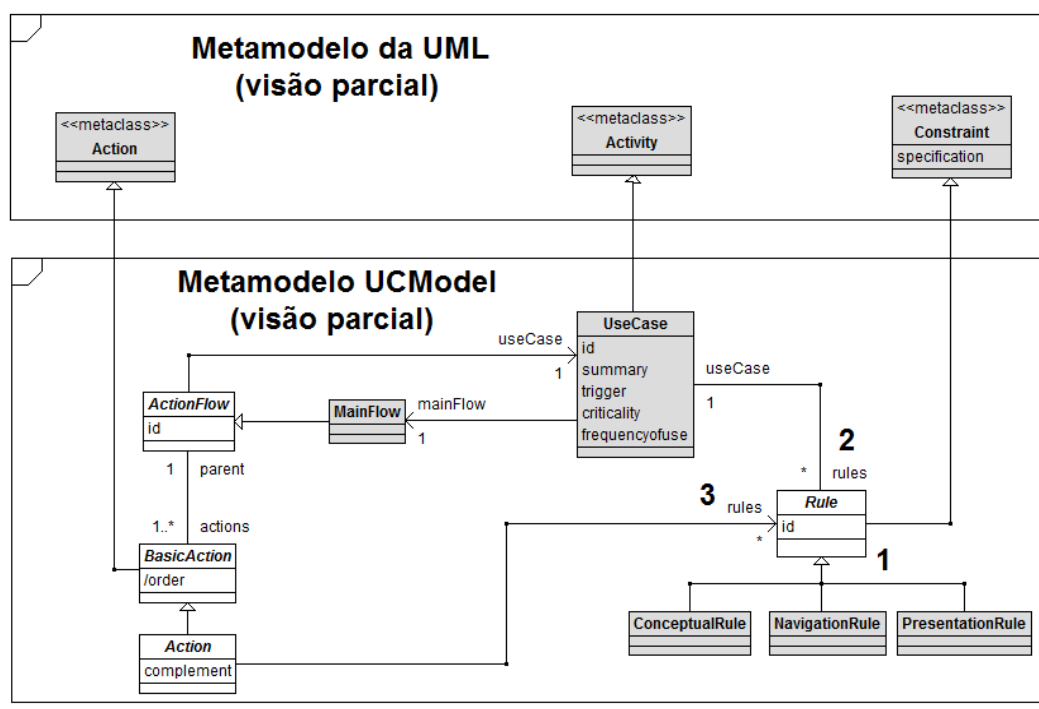


Figura 5-6 – Visão parcial do metamodelo UCModel referente às regras de negócio/domínio e sua relação com o metamodelo da UML

Representação no Diagrama de Atividades

Para representar os três tipos de regras de negócio/domínio (*ConceptualRule*, *NavigationRule* e *PresentationRule*) da Tabela 5-5 no diagrama de atividades foram definidos três estereótipos, conforme detalhado na Tabela 5-6:

Tabela 5-6 - Estereótipos usados na definição de regras de negócio/domínio

Estereótipo	Tipo-base	Definição
<code><<conceptual_rule>></code>	<i>Constraint</i>	Indica que a restrição é uma regra associada à perspectiva de conceituação
<code><<navigation_rule>></code>	<i>Constraint</i>	Indica que a restrição é uma regra associada à perspectiva de navegação
<code><<presentation_rule>></code>	<i>Constraint</i>	Indica que a restrição é uma regra associada à perspectiva de apresentação

A Figura 5-7 apresenta um exemplo de um diagrama de atividades com um caso de uso e as regras de negócio/domínio definidas para o mesmo. Como as regras no metamodelo UCMModel são criadas a partir do conceito de *Constraint* da UML, estas são representadas usando comentários no diagrama de atividades. Cada comentário contendo uma ou mais regras está associado à ação na qual esta(s) regra(s) impacta(m). Cada regra possui um identificador único dentro do caso de uso e é classificada com um único tipo.

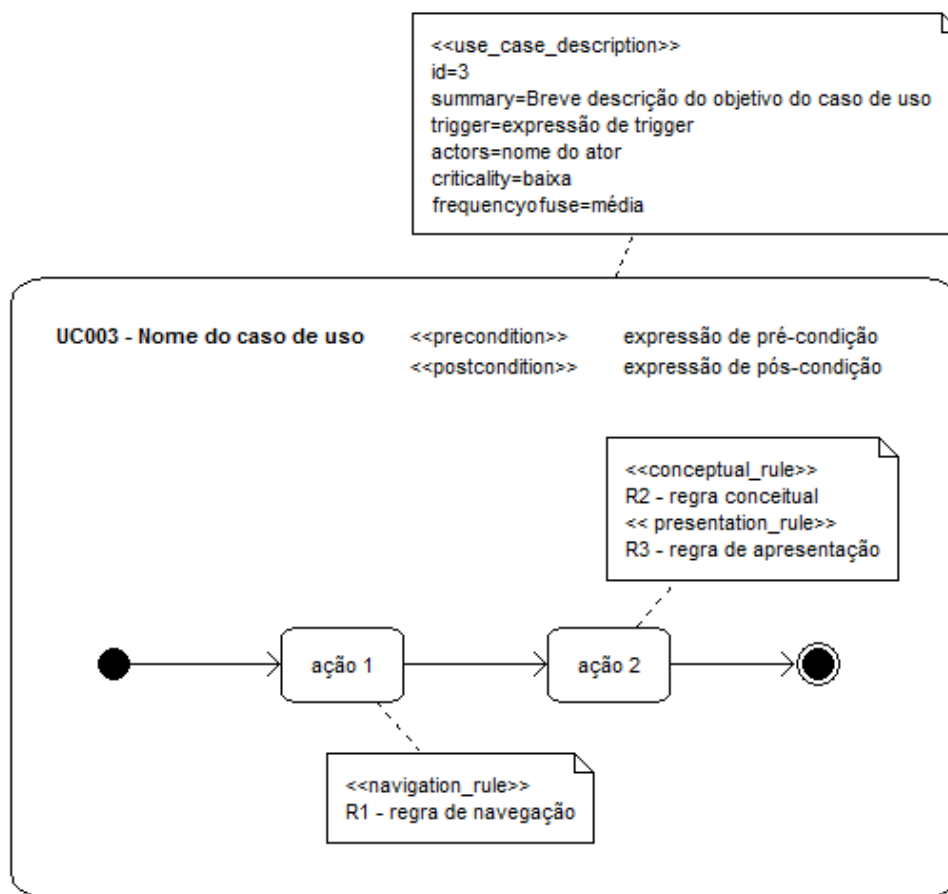


Figura 5-7 – Padrão de representação de um caso de uso e suas regras de negócio/domínio usando o diagrama de atividades da UML 2

5.3.3 Fluxos do Caso de Uso

A Tabela 5-7 apresenta os diferentes fluxos de execução em um caso de uso e como esses elementos são capturados pelo metamodelo UCMModel. A Figura 5-8 apresenta um extrato do metamodelo UCMModel (extrato da Figura 5-3) que destaca as metaclasses utilizadas para capturar os fluxos do caso de uso. Cada fluxo possui uma seqüência de passos (ações) que especificam o comportamento do mesmo e cada

fluxo alternativo e de exceção está associado a uma condição usada para avaliar se ele deve ou não ser executado.

Tabela 5-7 - Mapeamento entre os fluxos do caso de uso e os elementos do metamodelo UCMModel

Informação	Elemento do Modelo	Figura 5-8
Fluxo principal	<i>UseCase.mainFlow</i>	1
Ações do fluxo principal	<i>UseCase.mainFlow.actions</i>	5
Fluxo alternativo	<i>AlternativeFlow</i>	2
Ações do fluxo alternativo	<i>AlternativeFlow.actions</i>	5
Fluxo de exceção	<i>ExceptionFlow</i>	3
Ações do fluxo de exceção	<i>ExceptionFlow.actions</i>	5
Identificador do fluxo	<i>AlternativeFlow.id</i> <i>ExceptionFlow.id</i>	6
Condição de guarda dos fluxos	<i>AlternativeFlow.condition</i> <i>ExceptionFlow.condition</i>	4

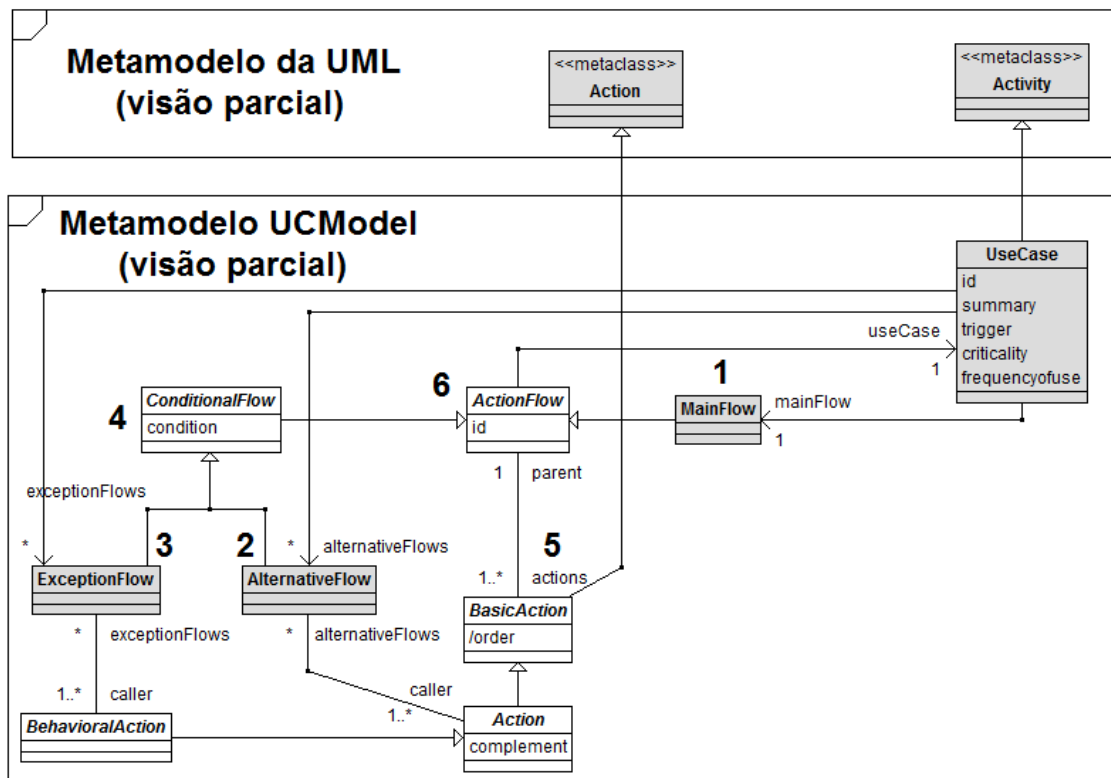



Figura 5-8 – Visão parcial do metamodelo UCMModel referente aos fluxos principal, alternativos e de exceção e sua relação com o metamodelo da UML

Representação no Diagrama de Atividades

A representação dos fluxos principal, alternativos e de exceção é feita de forma distinta no diagrama de atividades, conforme apresentado na Figura 5-9:

- **Fluxo principal:** seqüência de ações que representa o caminho ideal ou mais comum para que o ator atinja o objetivo desejado. É representado

pelos fluxos de controle que ligam o nó inicial ao nó final sem nenhuma condição de guarda associada a nenhum dos fluxos. Na Figura 5-9 o fluxo principal é formado pela seqüência de ações “ação 1”, “ação 2” e “ação 3”.

- **Fluxo alternativo:** seqüência de ações que indica uma escolha alternativa do ator ou um caminho alternativo para execução do caso de uso quando determinado estado ou condição é alcançado. É representada por um fluxo que se inicia sempre a partir de um nó de decisão. O nó de decisão indica que, neste ponto do diagrama, existe um caminho alternativo que pode ser executado dependendo da avaliação da condição associada ao fluxo alternativo. Os fluxos de controle que pertencem ao fluxo alternativo são nomeados com o mesmo identificador do fluxo alternativo, a fim de tornar claro para o leitor do diagrama que este caminho pertence aquele fluxo. Na Figura 5-9 os fluxos alternativos são os fluxos de controle nomeados “A1” e “A2”.
- **Fluxo de exceção:** seqüência de ações que interrompe a execução de uma outra ação indicando que uma situação extraordinária ocorreu e deve ser tratada. É representada por um fluxo do tipo `<<interrupt>>` (marcado pelo ícone ) que sai diretamente da ação que gerou a exceção, ou seja, não usa nó de decisão na sua representação. Na Figura 5-9 o fluxo de exceção é representado pelo fluxos de controle nomeados “E1”.

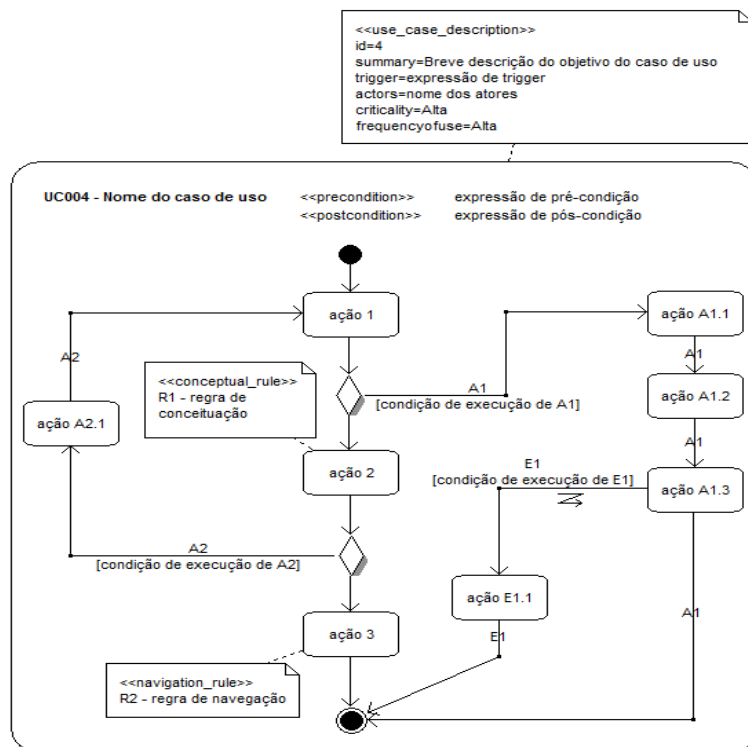


Figura 5-9 – Padrão de representação de um caso de uso e seus fluxos principal, alternativos e de exceção usando diagrama de atividades da UML

Como cada fluxo é representado de forma distinta e inequívoca no diagrama de atividades, não foi necessária a definição de estereótipos específicos para capturar esses elementos.

5.3.4 Ações do Caso de Uso

O metamodelo da UML prevê unicamente a metaclassa *Action* para descrição das ações em um diagrama de atividades. Embora esse conceito genérico possibilite o uso desse diagrama na representação de comportamentos sob diferentes perspectivas, o aumento no rigor da descrição de casos de uso que usam diagramas de atividades passa pela especialização das ações que compõem esse diagrama. A definição dessa especialização, por sua vez, demanda uma análise cuidadosa sobre a natureza das ações usadas na descrição de casos de uso. Neste trabalho, essa análise foi feita a partir do conceito de transação de JACOBSON (1992).

5.3.4.1 Transação e Casos de Uso

Os conceitos de caso de uso e transação, preconizados por JACOBSON (1992), são definidos como:

- “Um caso de uso é uma seqüência de transações executadas por um sistema, que produz um resultado de valor observável para um determinado ator”.
- “uma transação consiste em um conjunto de ações executadas por um sistema. Uma transação é invocada por um estímulo enviado do ator para o sistema ou por um gatilho de tempo dentro do sistema”.

De acordo com essas duas definições, uma transação pode ser detalhada da seguinte forma:

1. O ator envia ao sistema uma requisição e os dados;
2. O sistema valida a requisição e os dados, se necessário;
3. O sistema processa a requisição e altera seu estado interno ou produz um resultado a partir desta, e;
4. O sistema retorna o resultado do processamento da requisição ao ator.

Analisando a natureza das ações que detalham a transação do caso de uso é possível afirmar que:

- 1) O primeiro passo descreve uma ação que exprime o comportamento esperado para o ator. Essa ação é executada fora do escopo do sistema;

- 2) Os passos dois e três descrevem ações que mudam o estado interno do sistema ou produzem algum resultado. Essas ações normalmente consomem a requisição e os dados enviados pelo ator, e;
- 3) O quarto passo descreve uma ação do sistema que retorna para o ator algum resultado observável. Ela normalmente consulta o estado interno do sistema ou usa o resultado produzido anteriormente para gerar uma resposta para o ator.

Os conceitos contidos nesses três itens foram usados na elaboração de um conjunto de três ações que formam a base para descrição de casos de uso segundo a abordagem aqui proposta.

5.3.4.2 Especialização das Ações

A partir da análise do conceito de transação de caso de uso (JACOBSON, 1992) a metaclassa *Action* da UML foi especializada em três sub-classes: *ActorAction*, *SystemAction* e *SystemResponse* (Tabela 5-8). Adicionalmente, o foco de cada uma dessas ações torna possível outra correspondência relevante no contexto do desenvolvimento de aplicações Web: cada uma delas normalmente envolve o uso ou a referência a elementos de uma ou mais perspectivas de projeto Web (Tabela 5-8)

Tabela 5-8 - Principais tipos de ações previstos pelo metamodelo UCMModel

Tipo da Ação	Descrição	Perspectiva Web
<i>ActorAction</i>	Representa uma interação do ator com o sistema na qual este faz uma requisição ao sistema informando os dados necessários.	Apresentação e Navegação
<i>SystemAction</i>	Representa uma ação do sistema cujos resultados gerados não são diretamente observados pelo ator. É usada para explicitar o tratamento dado à requisição do ator. Esta ação está, normalmente, associada a recuperação de informações, alteração do estado interno do sistema e geração de resultados que serão posteriormente apresentados.	Conceituação
<i>SystemResponse</i>	Representa uma ação do sistema cujos resultados são direta ou indiretamente observados pelo ator. Essa ação pode apresentar resultados gerados anteriormente ou solicitar outros dados ao ator.	Apresentação e Navegação

Alguns trabalhos da literatura técnica tratam a ação do sistema e a resposta do sistema como um só elemento, que é normalmente chamado de ação do sistema (KOSTERS *et al.*, 2001; DURÁN *et al.*, 2002; GUTIÉRREZ *et al.*, 2008; LU e SONG,

2008). Todavia, as perspectivas associadas a cada uma dessas ações revelam diferenças sutis, porém importantes: enquanto a *SystemAction* lida com conceitos do domínio do problema que compõem a estrutura interna do sistema, a *SystemResponse* lida com elementos da interface com o mundo externo, seja essa uma interface gráfica padrão ou uma interface não convencional representada por algum tipo de dispositivo.

Apesar de serem a base para especificação de casos de uso, as três ações definidas na Tabela 5-8 não são suficientes. Para atender aos conceitos de inclusão (<<include>>) e extensão (<<extend>>) previstos na especificação da UML (OMG, 2010a) foram acrescentadas mais duas especializações da metaclassa *Action* da UML, definidas na Tabela 5-9:

Tabela 5-9 - Tipos de ações para inclusão e extensão de casos de uso

Tipo da Ação	Descrição
<i>IncludeUCAction</i>	Define a inclusão de outro caso de uso no ponto onde esta ação é definida.
<i>ExtendUCAction</i>	Define a inserção de outro caso de uso no ponto onde esta ação é definida.

5.3.4.3 Ações Especializadas no Metamodelo

A Tabela 5-10 apresenta as várias ações definidas para descrição de casos de uso e como esses elementos são capturados pelo metamodelo UCMModel. A Figura 5-10 apresenta um extrato do metamodelo UCMModel (extrato da Figura 5-3) que destaca as metaclasses utilizadas para capturar as ações especializadas.

Tabela 5-10 - Mapeamento entre as ações do caso de uso e os elementos do metamodelo UCMModel

Informação	Elemento do Modelo	Figura 5-10
Ação do ator	<i>ActorAction</i>	1
Ator da ação	<i>ActorAction.actor</i>	2
Ação do sistema	<i>AlternativeFlow</i>	3
Resposta do sistema	<i>AlternativeFlow.actions</i>	4
Ordem da ação	<i>BasicAction.order</i>	5
Fluxo ao qual a ação pertence	<i>BasicAction.parent</i>	6
Descrição da ação	<i>ActorAction.name</i> <i>SystemAction.name</i> <i>SystemResponse.name</i>	7
Complemento da ação	<i>ActorAction.complement</i> <i>SystemAction.complement</i> <i>SystemResponse.complement</i> <i>IncludeUCAction.complement</i> <i>ExtendUCAction.complement</i>	8
Fluxo alternativos associados à ação	<i>Action.alternativeFlows</i>	9

Fluxo de exceção associados à ação	<i>BehavioralAction.exceptionFlows</i>	10
Detalhamento do comportamento da ação	<i>BehavioralAction.behavior</i>	11
Inclusão de caso de uso	<i>IncludeUCAction</i>	12
Extensão de caso de uso	<i>ExtendUCAction</i>	13
Caso de uso incluído	<i>IncludeUCAction.uc</i>	14
Caso de uso que estende o atual	<i>ExtendUCAction.uc</i>	14

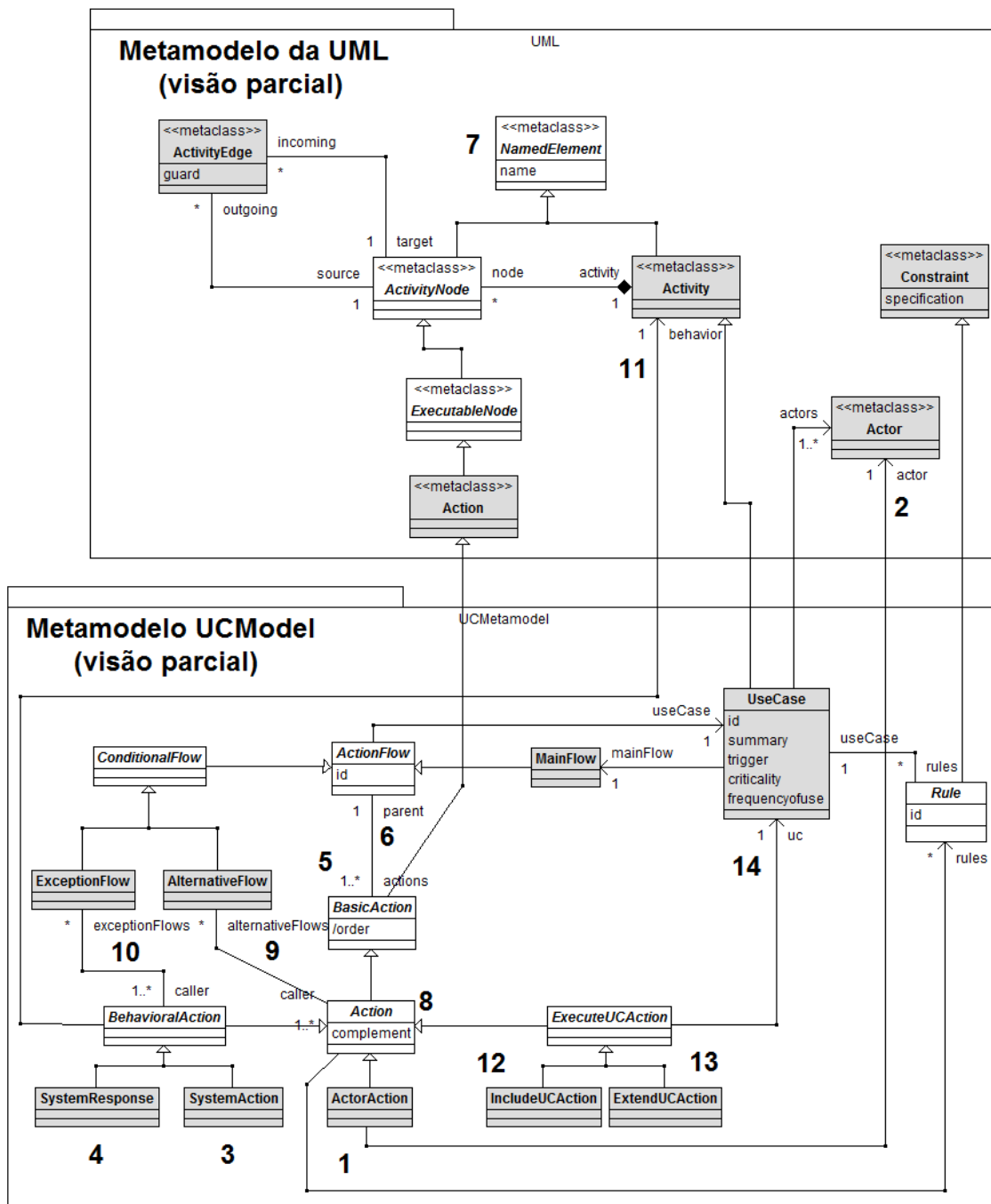


Figura 5-10 – Visão parcial do metamodelo UCMetamodel referente às ações que compõem os fluxos do caso de uso e a sua relação com o metamodelo da UML

Dois elementos do metamodelo UCMModel, apresentado na Figura 5-10, merecem explicações adicionais:

- Detalhamento do comportamento da ação (destaque 11): conforme definido no UCMModel é possível detalhar o comportamento de uma *SystemAction* ou *SystemResponse* usando um outro diagrama de atividades. Esse segundo diagrama de atividades não deverá ser criado usando os mesmos conceitos do metamodelo UCMModel, já que ele deverá conter somente ações que dizem respeito ao comportamento do sistema. Esse expediente pode ser usado para detalhar e formalizar um procedimento do sistema descrito como uma ação atômica no nível do caso de uso ou regras de negócio/domínio que devem ser observadas durante determinada ação ou resposta do sistema, ao invés de explorar a descrição dessas regras em linguagem natural.
- Complemento da ação (destaque 8): O atributo *complement*, que é parte integrante das ações *ActorAction*, *SystemAction*, *SystemResponse*, *IncludeUcAction* e *ExtendUcAction*, foi definido para que o desenvolvedor forneça informações complementares sobre a ação, permitindo que a descrição da mesma seja curta e objetiva. Esse recurso será explorado na seção 5.4 de orientações para descrição das ações.

Representação no Diagrama de Atividades

Para representar as ações do metamodelo UCMModel no diagrama de atividades foram definidos os estereótipos da Tabela 5-11.

Tabela 5-11 - Estereótipos usados na definição das ações

Estereótipo	Tipo-base	Definição
<<actor_action>>	<i>Action</i>	Define uma ação do ator (<i>ActorAction</i>)
	Tagged-value	
	<i>Actor</i>	Nome do ator que executa a ação
	<i>complement</i>	Informações complementares sobre a ação
Estereótipo	Tipo-base	Definição
<<system_action>>	<i>Action</i>	Define uma ação do sistema (<i>SystemAction</i>)
	Tagged-value	
	<i>complement</i>	Informações complementares sobre a ação
Estereótipo	Tipo-base	Definição
<<system_response>>	<i>Action</i>	Define uma resposta do sistema (<i>SystemResponse</i>)
	Tagged-value	

	<i>complement</i>	Informações complementares sobre a ação
Estereótipo	Tipo-base	Definição
<< <i>include_action</i> >>	<i>Action</i>	Define um ponto de inclusão
	Tagged-value	
	<i>uc</i>	Identificador do caso de uso a ser incluído
	<i>complement</i>	Informações complementares sobre a ação
Estereótipo	Tipo-base	Definição
<< <i>extend_action</i> >>	<i>Action</i>	Define um ponto de extensão
	Tagged-value	Definição
	<i>uc</i>	Identificador do caso de uso que estende o caso de uso corrente
	<i>complement</i>	Informações complementares sobre a ação

5.3.5 Combinando Ações na Descrição do Comportamento

A definição de cada ação proposta pelo metamodelo UCMModel e o próprio conceito de transação sugerem que nem todas as combinações entre essas ações fazem sentido. Por exemplo: não faz sentido especificar uma ação do ator (*ActorAction*) logo após uma ação do sistema (*SystemAction*), já que o ator, nesse ponto, ainda não obteve nenhuma resposta informando o que aconteceu durante o processamento da requisição. Da mesma forma não faz sentido especificar uma resposta do sistema (*SystemResponse*) logo após uma ação do ator (*ActorAction*), pois é necessário processar a requisição para que o resultado a ser apresentado seja gerado.

As quatro seções a seguir discutem as combinações válidas entre ação do ator (*ActorAction*), ação do sistema (*SystemAction*) e resposta do sistema (*SystemResponse*) com a possibilidade de definição de fluxos alternativos entre eles.

5.3.5.1 Transação Básica sem Fluxos Alternativos

A Figura 5-11 apresenta o fluxo básico das transições permitidas pelo metamodelo UCMModel envolvendo as ações *ActorAction*, *SystemAction* e *SystemResponse* sem levar em conta a existência de fluxos alternativo.

```
basic_flow:(system_action* system_response)?
          (actor_action (system_action+ system_response?))+ ;
```

Figura 5-11 - Transições básicas permitidas entre as ações *ActorAction*, *SystemAction* e *SystemResponse*

A Figura 5-11 mostra que:

1. O sucessor do nó inicial pode ser qualquer uma das três ações, ou seja, a descrição do fluxo de ações pode ser iniciada com qualquer uma das três ações;
2. O antecessor do nó final deve ser uma *SystemAction* ou uma *SystemResponse*, ou seja, a última ação de um fluxo deve ser uma dessas duas ações;
3. O sucessor de uma *ActorAction* deve ser uma *SystemAction*;
4. O sucessor de uma *SystemAction* deve ser: o nó final, outra *SystemAction* ou uma *SystemResponse*; e;
5. O sucessor de uma *SystemResponse* deve ser o nó final ou uma *ActorAction*.

O cenário apresentado pela Figura 5-11 torna-se mais complexo levando-se em consideração a possibilidade da existência de fluxos alternativos após cada uma dessas ações. Conforme definido no metamodelo da UML (Figura 5-1 – página 97), é possível ter nós de decisão, que representam fluxos alternativos no caso de uso, após cada ação da atividade. Como o metamodelo UCMModel se baseia no metamodelo da UML, é possível ter fluxos alternativos associados às ações *ActorAction*, *SystemAction* e *SystemResponse*.

5.3.5.2 Ação do Ator com Fluxo Alternativo

A Figura 5-12 apresenta as transições válidas caso exista um fluxo alternativo (*DecisionNode*) associado a uma ação do ator (*ActorAction*). De acordo com essa figura, a partir do nó de decisão é possível continuar a execução pelo fluxo principal ou iniciar um fluxo alternativo. Em ambos os casos, porém, a seqüência de ações é a mesma, ou seja, tanto no fluxo principal quanto no fluxo alternativo a primeira ação do fluxo deve ser uma ação do sistema. Esse arranjo reforça a idéia de que uma requisição do ator deve sempre ser tratada, mesmo que esse tratamento leve à conclusão de que a requisição não pode ser atendida.

```
aa_flow : actor_action decision_node
        (principal system_action+ system_response? |
         alternativo (system_action+ system_response?)) ;
```

Figura 5-12 - Transições permitidas a partir de uma *ActorAction* com fluxo alternativo

5.3.5.3 Ação do Sistema com Fluxo Alternativo

A Figura 5-13 apresenta as transições válidas caso exista um fluxo alternativo (*DecisionNode*) associado a uma ação do sistema (*SystemAction*). Esta figura define que, caso exista um fluxo alternativo associado à uma ação do sistema, este fluxo deve iniciar, obrigatoriamente, com outra ação do sistema (*SystemAction*) ou uma resposta do sistema (*SystemResponse*). Esse arranjo possibilita que o fluxo alternativo represente possibilidades alternativas de resposta do sistema ou de processamento da informação em função do resultado obtido em determinada ação do sistema.

```
sa_flow : (system_action+ ( | decision_node ))+
         system_response? ;
```

Figura 5-13 - Transições permitidas a partir de uma *SystemAction* com fluxo alternativo

5.3.5.4 Resposta do Sistema com Fluxo Alternativo

A Figura 5-14 apresenta as transições válidas caso exista um fluxo alternativo (*DecisionNode*) associado a uma resposta do sistema (*SystemResponse*). Esta figura define que, se houver um fluxo alternativo após uma resposta do sistema, este fluxo deve iniciar, obrigatoriamente, com outra resposta do sistema (*SystemResponse*) ou ação do sistema (*SystemAction*). Nesse caso, como a resposta do sistema apresenta o resultado de um processamento e/ou a solicitação de uma nova informação, então o fluxo alternativo é interpretado como uma atitude do ator diferente daquela esperada/solicitada pelo sistema, o que leva ao início de um novo diálogo com o sistema.

```
sr_flow : system_response decision_node
         (principal actor_action |
          alternativo (system_response | system_action )) ;
```

Figura 5-14 Transições permitidas a partir de uma *SystemResponse* com fluxo alternativo

5.3.6 Restrições do Metamodelo UCMModel

Os diagramas apresentados nas Figuras 5-11 a 5-14 não esgotam todas as restrições previstas no metamodelo UCMModel. Outras restrições foram definidas com o objetivo de dar consistência ao metamodelo e se baseiam na semântica capturada por cada elemento ou grupo de elementos.

As seções a seguir apresentam uma lista de restrições para cada elemento do metamodelo e a sua respectiva especificação em OCL (*Object Constraint Language*) (OMG, 2010b).

5.3.6.1 Metaclassa *UseCase*

- 1) ID é obrigatório
- 2) Nome é obrigatório
- 3) Resumo é obrigatório
- 4) Criticalidade é obrigatória
- 5) Frequência de uso é obrigatória
- 6) Pré-condição é obrigatória
- 7) Pós-condição é obrigatória
- 8) Deve ter pelo menos um ator associado

Formalização em OCL:

```
context: UseCase
inv: self.id->notEmpty()
inv: self.name->notEmpty()
inv: self.description->notEmpty()
inv: self.criticality->notEmpty()
inv: self.frequencyofuse->notEmpty()
inv: self.precondition.specification->notEmpty()
inv: self.postcondition.specification->notEmpty()
inv: self.actors->size() >= 1
```

5.3.6.2 Metaclassa *Actor*

- 1) Nome é obrigatório

Formalização em OCL:

```
context: Actor
inv: self.name->notEmpty()
```

5.3.6.3 Metaclassa *MainFlow*

- 1) Deve possuir pelo menos uma transação

Formalização em OCL:

```
context: MainFlow
inv: (self.actions->size() >= 2
    and
    self.actions->asSequence()
        ->at(0).oclIsTypeOf(ActorAction)
    and
    self.actions->asSequence()
        ->at(1).oclIsTypeOf(SystemAction))
or
```

```

(self.actions->size() >= 2 and
 self.actions->asSequence()
  ->at(0).oclIsTypeOf(SystemAction)
 and
 self.actions->asSequence()
  ->at(1).oclIsTypeOf(SystemResponse))
or
(self.actions->size() >= 3 and
 self.actions->asSequence()
  ->at(0).oclIsTypeOf(SystemResponse)
 and
 self.actions->asSequence()
  ->at(1).oclIsTypeOf(ActorAction)
 and
 self.actions->asSequence()
  ->at(2).oclIsTypeOf(SystemAction))

```

5.3.6.4 Metaclass *AlternativeFlow*

- 1) Condição de guarda é obrigatória
- 2) Deve ter pelo menos uma ação
- 3) Deve ser referenciado por pelo menos uma ação do *UseCase*
- 4) Deve sempre ser disparado por ações do mesmo tipo
- 5) Se for disparado por uma *ActorAction*, então sua primeira ação tem que ser uma *SystemAction*
- 6) Se for disparado por uma *SystemAction*, então sua primeira ação tem que ser uma *SystemResponse* ou uma *UML::ActivityFinalNode*
- 7) Se for disparado por uma *SystemResponse*, então sua primeira ação tem que ser uma *SystemResponse*, uma *SystemAction* ou uma *UML::ActivityFinalNode*

Formalização em OCL:

```

context: AlternativeFlow
inv: self.condition->notEmpty()
inv: self.actions->size() >= 1
inv: self.caller->size() >= 1
inv: self.caller->size() =
  self.caller
  ->select(incoming.source.oclIsTypeOf(ActorAction))
  ->size()
or
self.caller->size() =
self.caller
  ->select(incoming.source.oclIsTypeOf(SystemAction))
  ->size()
or
self.caller->size() =
self.caller

```

```

->select (incoming.source.oclIsTypeOf (SystemResponse)
->size ()
inv: self.caller->asSequence ()
->first ().incoming.source.oclIsTypeOf (ActorAction)
implies
self.actions->asSequence ()
->first ().oclIsTypeOf (SystemAction)
inv: self.caller->asSequence ()
->first ().incoming.source.oclIsTypeOf (SystemAction)
implies
(self.actions->asSequence ()
->first ().oclIsTypeOf (SystemResponse)
or
self.actions->asSequence ()
->first ().oclIsTypeOf (ActivityFinalNode))
inv: self.caller->asSequence ()
->first ().incoming.source.oclIsTypeOf (SystemResponse)
implies
(self.actions->asSequence ()
->first ().oclIsTypeOf (SystemResponse)
or
self.actions->asSequence ()
->first ().oclIsTypeOf (SystemAction)
or
self.actions->asSequence ()
->first ().oclIsTypeOf (ActivityFinalNode))

```

5.3.6.5 Metaclass *ExceptionFlow*

- 1) Condição de guarda é obrigatória
- 2) Deve ter pelo menos uma ação
- 3) Deve ser referenciado por pelo menos uma ação do *UseCase*
- 4) Deve estar sempre associado a ações do mesmo tipo, ou seja, sempre é disparado por ações do mesmo tipo
- 5) Se for disparado por uma *SystemAction*, então sua primeira ação tem que ser uma *SystemResponse* ou outra *SystemAction*
- 6) Se for disparado por uma *SystemResponse*, então sua primeira ação tem que ser uma *SystemResponse*, uma *SystemAction* ou uma *UML::ActivityFinalNode*

Formalização em OCL:

```

context: ExceptionFlow
inv: self.condition->notEmpty ()
inv: self.actions->size () >= 1
inv: self.caller->size () >= 1
inv: self.caller->size () =
self.caller
->select (oclIsTypeOf (SystemAction) )->size ()

```

```

    or
    self.caller->size() =
    self.caller
    ->select(oclIsTypeOf(SystemResponse))->size()
inv: self.caller->asSequence()
    ->first().oclIsTypeOf(SystemAction)
    implies
    (self.actions->asSequence()
    ->first().oclIsTypeOf(SystemResponse)
    or
    self.actions->asSequence()
    ->first().oclIsTypeOf(ActivityFinalNode))
inv: self.caller->asSequence()
    ->first().oclIsTypeOf(SystemResponse)
    implies
    (self.actions->asSequence()
    ->first().oclIsTypeOf(SystemResponse)
    or
    self.actions->asSequence()
    ->first().oclIsTypeOf(SystemAction)
    or
    self.actions->asSequence()
    ->first().oclIsTypeOf(ActivityFinalNode))

```

5.3.6.6 Metaclass *ActorAction*

- 1) Descrição da ação é obrigatória
- 2) Deve ter um ator associado
- 3) Não pode ser a última ação de um fluxo
- 4) Só pode ser sucedida por uma única ação
- 5) Deve ser sucedida por uma *SystemAction*, *IncludeUCAction* ou *ExtendUCAction*

Formalização em OCL:

```

context: ActorAction
inv: self.name->notEmpty()
inv: self.actor->notEmpty()
inv: self.order < self.parent->size()
inv: self.outgoing->size() = 1
inv: self.outgoing.target.oclIsTypeOf(SystemAction)
    or
    self.outgoing.target.oclIsTypeOf(IncludeUCAction)
    or
    self.outgoing.target.oclIsTypeOf(ExtendUCAction)

```

5.3.6.7 Metaclass *SystemAction*

- 1) Descrição é obrigatória

- 2) Se estiver no fluxo principal deve ser sucedida por no máximo uma ação do tipo *SystemAction*, *SystemResponse*, *IncludeUAction* ou *ExtendUAction*
- 3) Se estiver no fluxo alternativo ou de exceção deve ser sucedida por uma única ação do tipo *SystemAction*, *SystemResponse*, *UML::ActivityFinalNode*, *IncludeUAction* ou *ExtendUAction*

Formalização em OCL:

```

context: SystemAction
inv: self.name->notEmpty()
inv: self.parent.oclIsTypeOf(MainFlow)
    implies
        self.outgoing->size() <= 1
        and
        (self.outgoing.target.oclIsTypeOf(SystemAction)
        or
        self.outgoing.target.oclIsTypeOf(SystemResponse)
        or
        self.outgoing.target.oclIsTypeOf(IncludeUAction)
        or
        self.outgoing.target.oclIsTypeOf(ExtendUAction)
inv: self.parent.oclIsTypeOf(AlternativeFlow)
    or
    self.parent.oclIsTypeOf(ExceptionFlow)
    implies
        self.outgoing->size() = 1
        and
        (self.outgoing.target.oclIsTypeOf(SystemAction)
        or
        self.outgoing.target.oclIsTypeOf(SystemResponse)
        or
        self.outgoing.target.oclIsTypeOf(ActivityFinalNode)
        or
        self.outgoing.target.oclIsTypeOf(IncludeUAction)
        or
        self.outgoing.target.oclIsTypeOf(ExtendUAction)

```

5.3.6.8 Metaclassa *SystemResponse*

- 1) Descrição é obrigatória
- 2) Se estiver no fluxo principal deve ser sucedida por no máximo uma ação do tipo Deve ser sucedida por uma *ActorAction*, *IncludeUAction* ou *ExtendUAction*
- 3) Se estiver no fluxo alternativo ou de exceção deve ser sucedida por uma única ação do tipo *ActorAction*, *SystemResponse*, *UML::ActivityFinalNode*, *IncludeUAction* ou *ExtendUAction*

Formalização em OCL:

```
context: SystemResponse
inv: self.name->notEmpty()
inv: self.parent.oclIsTypeOf(MainFlow)
    implies
        self.outgoing->size() <= 1
        and
        (self.outgoing.target.oclIsTypeOf(ActorAction)
        or
        self.outgoing.target.oclIsTypeOf(IncludeUAction)
        or
        self.outgoing.target.oclIsTypeOf(ExtendUAction)
inv: self.parent.oclIsTypeOf(AlternativeFlow)
    or
    self.parent.oclIsTypeOf(ExceptionFlow)
    implies
        self.outgoing->size() = 1
        and
        (self.outgoing.target.oclIsTypeOf(ActorAction)
        or
        self.outgoing.target.oclIsTypeOf(SystemResponse)
        or
        self.outgoing.target.oclIsTypeOf(ActivityFinalNode)
        or
        self.outgoing.target.oclIsTypeOf(IncludeUAction)
        or
        self.outgoing.target.oclIsTypeOf(ExtendUAction)
```

5.3.6.9 Metaclasses *IncludeUAction* e *ExtendUAction*

- 1) Deve referenciar um UseCase

Formalização em OCL:

```
context: ExecuteUAction
inv: self.uc->notEmpty()
```

5.3.6.10 Metaclasses *ConceptualRule*, *NavigationRule* e *PresentationRule*

- 1) Descrição é obrigatória
- 2) Deve ser referenciada por pelo menos uma ação do *UseCase*

Formalização em OCL:

```
context: Rule
inv: self.specification->notEmpty()
inv: self.useCase.mainFlow.actions
    ->any(rules->any(id = self.id))
    or
```

```

self.useCase.alternativeFlows->forAll(af |
  af.actions->any(rules->any(id = self.id)))
or
self.useCase.exceptionFlows->forAll(ef |
  ef.actions->any(rules->any(id = self.id)))

```

A estrutura do metamodelo apresentado na figura complementado pelo conjunto de restrições apresentadas nas seções anteriores formam a base teórica para organização de um apoio ferramental para criação e validação de descrições de casos de uso, segundo o metamodelo UCMModel definido no escopo deste trabalho.

5.3.7 Perfil UML

Todos os estereótipos definidos nas seções anteriores foram reunidos em um perfil UML, chamado ProfileUCModel, organizado com o objetivo de permitir a criação de diagramas de atividades, segundo os conceitos preconizados pelo metamodelo UCMModel, com o apoio de ferramentas UML capazes de tratar perfis UML. A Figura 5-15 apresenta os estereótipos e *tagged-values* definidos no ProfileUCModel.

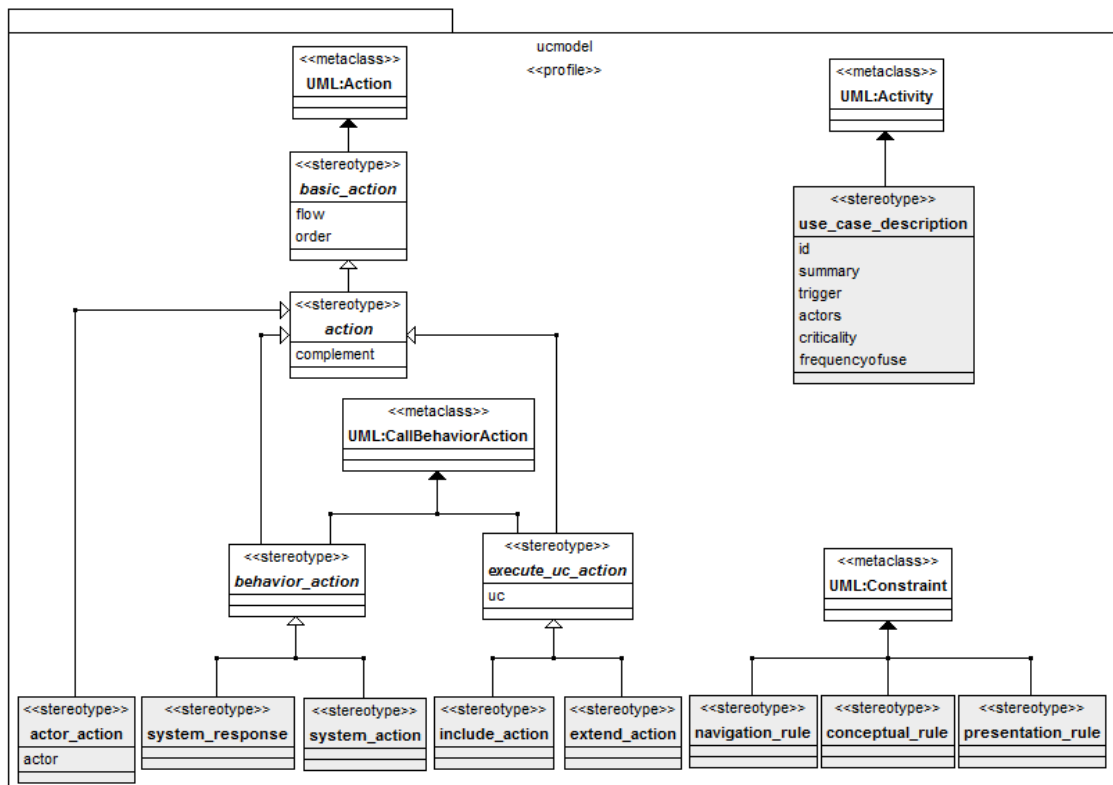


Figura 5-15 – Estereótipos e *tagged-values* para definição de diagramas de atividades segundo o UCMModel

5.4 Orientações para Especificação de Casos de Uso no UCMModel

A definição semântica das ações que compõem a especificação do caso de uso torna possível, além da validação da estrutura da descrição do caso de uso (validação sintática), a definição de um conjunto de orientações voltadas para a redação da descrição das ações que o compõem. A organização dos itens também incluiu algumas orientações gerais presentes na literatura técnica (COX *et al.*, 2001; ROSENBERG e STEPHENS, 2007) e que estão diretamente alinhadas com as orientações propostas neste trabalho.

5.4.1 Orientações Gerais

- Na descrição das ações use sempre frases curtas e objetivas;
- Indique sempre quem realiza a ação: “Sistema...” ou “Ator...”;
- Use sempre verbos no presente e na voz ativa;
- Não acrescente detalhes sobre a ação na sua descrição. Caso seja necessário fornecer alguma informação complementar use o complemento da ação para essa tarefa;
- Seja coerente na adoção dos termos usados na descrição. Em caso de existência de sinônimos adote um termo e utilize-o de forma consistente, e;
- Não use termos que direcionem para uma solução ou tecnologia específica, e;
- Não junte ações do sistema e resposta do sistema em uma única ação. Essas ações têm naturezas diferentes e podem estar associadas à alternativas, exceções ou regras diferentes.

5.4.2 Orientações relacionadas à Ação do Ator

- Indique objetivamente a requisição enviada pelo ator com seus respectivos dados;
- Use termos relacionados à perspectiva de apresentação ou navegação;
- Referencia regras relacionadas às perspectivas de apresentação ou navegação, e;
- As alternativas disparadas por uma ação do ator devem fazer sentido no contexto desse tipo de ação. Normalmente essas alternativas estão associadas à requisição enviada pelo ator e/ou estado do sistema.

5.4.3 Orientações relacionadas à Ação do sistema

- Use termos relacionados ao domínio do problema (perspectiva de conceituação), ou seja, os mesmo termos explorados nos modelos conceituais;
- Deixe explícito, através do complemento da ação ou de regras, os efeitos colaterais ou pós-condições da execução da regra em relação ao estado do sistema;
- Referencia regras relacionadas à perspectiva de conceituação;
- As alternativas disparadas por uma ação do sistema devem fazer sentido no contexto desse tipo de ação. Normalmente essas alternativas estão associadas ao resultado da execução da ação e/ou estado do sistema;
- As exceções associadas à ação do sistema devem representar situações extraordinárias nesse contexto, como por exemplo, situações onde há impossibilidade de processar requisição do ator, e;
- Caso exista uma estrutura de ações do sistema associadas à alternativas que levam a outras ações do sistema, verifique se o nível de abstração dessas ações é compatível com o caso de uso. Caso não seja, refatore essa estrutura como um diagrama de atividades à parte e substitua-a na descrição do caso de uso por uma ação que indique, do ponto de vista do ator, a ação esperada do sistema.

5.4.4 Orientações relacionadas à Resposta do Sistema

- Indique objetivamente os resultados que o sistema irá retornar ao ator;
- Use termos relacionados à perspectiva de apresentação ou navegação;
- Referencia regras relacionadas às perspectivas de apresentação ou navegação;
- As alternativas disparadas por uma resposta do sistema devem fazer sentido no contexto desse tipo de ação. Normalmente essas alternativas estão associadas a caminhos alternativos que o ator seleciona, e;
- As exceções associadas à uma resposta do sistema devem representar situações extraordinárias nesse contexto, como por exemplo, situações onde há impossibilidade de apresentar essa resposta ao ator.

É importante ressaltar que esses itens foram explorados na elaboração do *checklist* da técnica ActCheck (MELLO, 2011), criada para inspeção de diagramas de


atividades e que pode ser customizada para inspecionar diagramas de atividades que descrevem casos de uso segundo o metamodelo UCMModel.

5.5 Exemplo de Caso de Uso especificado com o UCMModel

Essa seção apresenta um exemplo completo do caso de uso “Autenticar usuário”, descrito de acordo com o metamodelo UCMModel. A Figura 5-16 apresenta a especificação desse caso de uso usando um diagrama de atividades que explora o perfil ProfileUCModel para definir a semântica dos vários elementos que fazem parte da especificação. A Figura 5-17 apresenta a descrição textual desse mesmo caso de uso usando o gabarito organizado na seção 4.5.3.2 e apresentado na Tabela 4-10 – página 81.

É importante frisar que, em termos semânticos, as especificações de caso de uso apresentadas no diagrama de atividades (Figura 5-16) e na descrição textual (Figura 5-17) são idênticas. Na prática, a descrição textual apresentada na Figura 5-17 foi gerada automaticamente a partir do diagrama de atividades da Figura 5-16, com o apoio ferramental que será detalhado no capítulo 6.

Observando a especificação do caso de uso “Autenticar usuário”, representada como um diagrama de atividades na Figura 5-16, é possível visualizar os elementos do metamodelo UCMModel definidos nas seções anteriores. Os itens listados a seguir estão destacados no diagrama (Figura 5-16) com o mesmo número do item.

- 1) O caso de uso e seus elementos básicos são representados pela atividade, estereotipada como `<<use_case_description>>`, e seus *tagged-values*. No diagrama tanto o estereótipo quanto as *tagged-values* são representados como um comentário associado à atividade;
- 2) Todas as ações do diagrama são estereotipadas para indicar o seu tipo (`<<actor_action>>`, `<<system_action>>` ou `<<system_response>>`);
- 3) Os pontos de ativação dos fluxos alternativos são marcados pelos nós de decisão do diagrama;
- 4) O ponto de ativação do fluxo de exceção é marcado pelo fluxo de interrupção partindo direto da ação onde essa exceção é gerada (estereótipo `<<interrupt>>`, também representado pelo ícone ).
- 5) As condições que restringem a execução dos fluxos alternativos e de exceção são definidas como condições de guarda associadas ao início desses fluxos;
- 6) As seqüências de ações que compõem os fluxos alternativos e de exceção são etiquetadas com os identificadores dos fluxos;

- 7) As regras são representadas como *Constraints* e estão estereotipadas (classificadas) de acordo com o seu tipo, e;
- 8) As regras estão sempre associadas às ações que restringem.

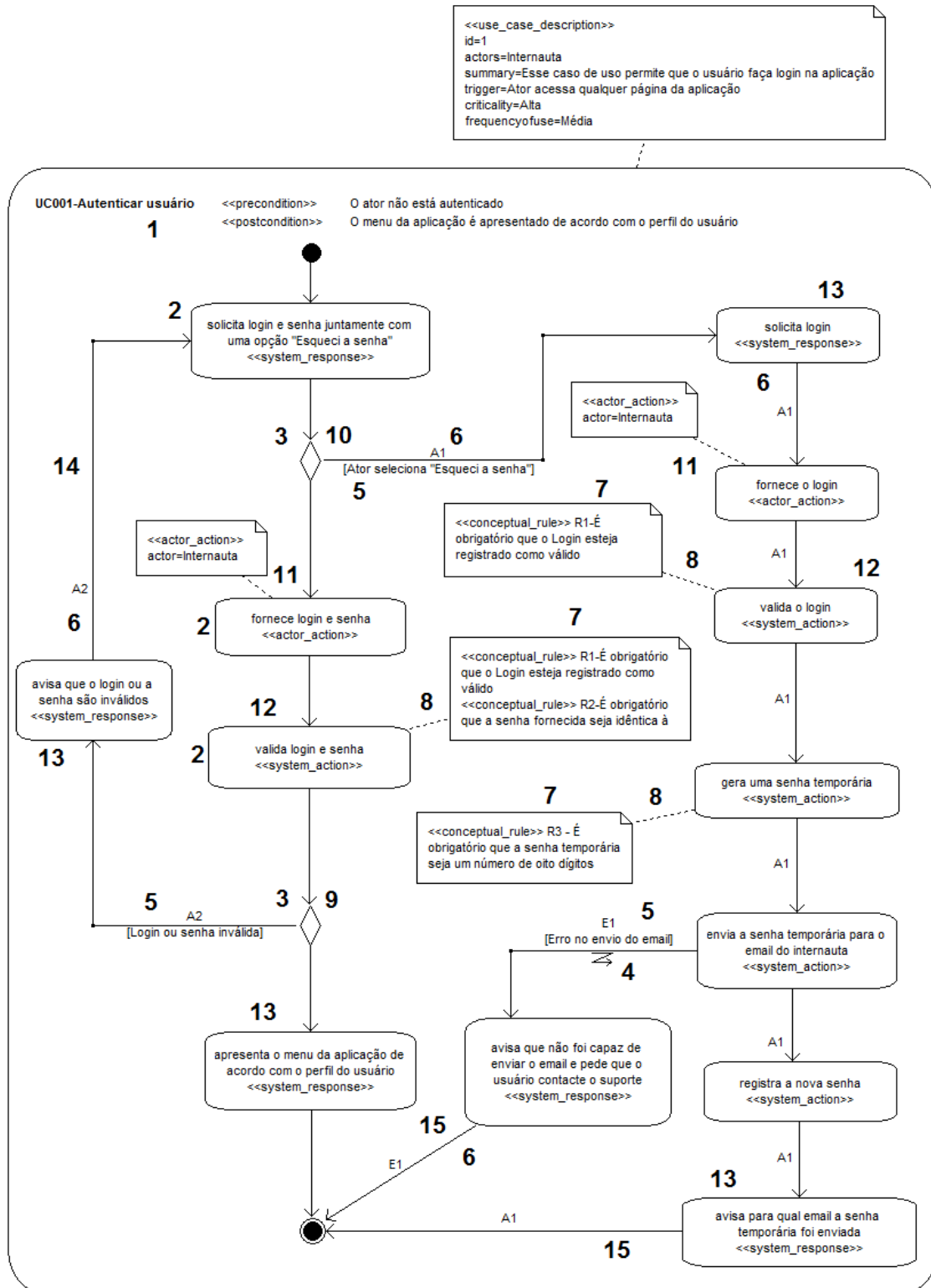


Figura 5-16 - Diagrama de atividades com a descrição do caso de uso da Figura 5-17

Identificação:	UC001
Nome:	Autenticar usuário
Descrição:	Esse caso de uso permite que o usuário faça login na aplicação
Ator(es):	Internauta
Criticalidade:	Alta
Frequência de uso:	Média
Pré-condições:	O ator não está autenticado
Pós-condições:	O menu da aplicação é apresentado de acordo com o perfil do usuário
Trigger:	Ator acessa qualquer página da aplicação
Fluxo Principal:	<ol style="list-style-type: none"> 1. Sistema solicita login e senha juntamente com uma opção “Esqueci a senha” 2. Internauta fornece login e senha [A1] 3. Sistema valida login e senha [R1, R2] 4. Sistema apresenta o menu da aplicação de acordo com o perfil do usuário [A2]
Fluxos Alternativos:	<p>[A1] Ator seleciona “Esqueci a senha”</p> <ol style="list-style-type: none"> A1.1. Sistema solicita o login A1.2. Internauta fornece o login A1.3. Sistema valida o login [R1] A1.4. Sistema gera uma senha temporária [R3] A1.5. Sistema envia a senha temporária para o email do internauta [E1] A1.6. Sistema registra a nova senha A1.7. Sistema avisa para qual email a senha temporária foi enviada. A1.8. Caso de uso é encerrado <p>[A2] Login ou senha inválida</p> <ol style="list-style-type: none"> A2.1. Sistema avisa que o login ou a senha são inválidos A2.2. Vai para 1
Fluxos de Exceção:	<p>[E1] Erro no envio do email</p> <ol style="list-style-type: none"> E1.1. Sistema avisa que não foi capaz de enviar o email e pede que o usuário contacte o suporte. E1.2. Caso de uso é encerrado
Regras:	<p>R1 – É obrigatório que o login esteja registrado como válido</p> <p>R2 – É obrigatório que a senha fornecida seja idêntica à senha associada ao login</p> <p>R3 – É obrigatório que a senha temporária seja um número de 8 dígitos</p>

Figura 5-17 – Descrição textual do caso de uso “Autenticar usuário” de acordo com o gabarito apresentado na Tabela 4-10 – página 81

No diagrama de atividades da Figura 5-16 é possível, também, observar a aderência desse modelo às restrições estabelecidas no escopo do metamodelo UCMModel. A Tabela 5-12 lista as regras aplicadas ao exemplo da Figura 5-16, destacando os itens do diagrama onde a aderência à regra pode ser observada.

Tabela 5-12- Restrições aplicadas ao caso de uso "Autenticar usuário" da Figura 5-16

Metaclasse	Restrições	Item
<i>UseCase</i>	<ol style="list-style-type: none"> 1) ID é obrigatório 2) Nome é obrigatório 3) Resumo é obrigatório 4) Criticalidade é obrigatória 5) Frequência de uso é obrigatória 6) Pré-condição é obrigatória 7) Pós-condição é obrigatória 8) Deve ter pelo menos um ator associado 	1
<i>MainFlow</i>	<ol style="list-style-type: none"> 1) Deve possuir pelo menos uma transação 	2
<i>AlternativeFlow</i>	<ol style="list-style-type: none"> 1) Condição de guarda é obrigatória 	5
	<ol style="list-style-type: none"> 2) Deve ter pelo menos uma ação 	6

	3) Deve ser referenciado por pelo menos uma ação do <i>UseCase</i>	3
	4) Deve estar sempre associado a ações do mesmo tipo, ou seja, sempre é disparado por ações do mesmo tipo	3
	6) Se for disparado por uma <i>SystemAction</i> , então sua primeira ação tem que ser uma <i>SystemResponse</i> ou uma <i>UML::ActivityFinalNode</i>	9
	7) Se for disparado por uma <i>SystemResponse</i> , então sua primeira ação tem que ser uma <i>SystemResponse</i> , uma <i>SystemAction</i> ou uma <i>UML::ActivityFinalNode</i>	10
<i>ExceptionFlow</i>	1) Condição de guarda é obrigatória	5
	2) Deve ter pelo menos uma ação	6
	3) Deve ser referenciado por pelo menos uma ação do <i>UseCase</i>	4
	4) Deve estar sempre associado a ações do mesmo tipo, ou seja, sempre é disparado por ações do mesmo tipo	4
	5) Se for disparado por uma <i>SystemAction</i> , então sua primeira ação tem que ser uma <i>SystemResponse</i> ou outra <i>SystemAction</i>	4
<i>ActorAction</i>	1) Descrição da ação é obrigatória 2) Deve ter um ator associado 3) Não pode ser a última ação de um fluxo 4) Deve ser sucedida por uma <i>SystemAction</i> , <i>IncludeUCAction</i> ou <i>ExtendUCAction</i>	11
<i>SystemAction</i>	1) Descrição é obrigatória 2) Deve ser sucedida por outra <i>SystemAction</i> , <i>SystemResponse</i> , <i>UML::ActivityFinalNode</i> , <i>IncludeUCAction</i> ou <i>ExtendUCAction</i>	12
<i>SystemResponse</i>	1) Descrição é obrigatória 2) Deve ser sucedida por uma <i>ActorAction</i> , <i>UML::ActivityFinalNode</i> , <i>IncludeUCAction</i> ou <i>ExtendUCAction</i>	13
<i>ConceptualRule</i>	1) Descrição é obrigatória 2) Deve ser referenciada por pelo menos uma ação do <i>UseCase</i>	7 e 8

5.6 Conclusão

Este capítulo apresentou o metamodelo UCModel, definido no escopo deste trabalho com o objetivo de oferecer um arcabouço sintático e semântico a partir do qual é possível estruturar a especificação do comportamento do sistema através de um conjunto de critérios e restrições que apóiam a especificação de casos de uso.

Dois princípios nortearam a definição do metamodelo UCModel:

- O conceito de transação de casos de uso (JACOBSON, 1992), a partir do qual foi elaborado um conjunto de ações especializadas para especificação das seqüências de ações do caso de uso, e;
- As perspectivas de projeto Web, que reforçaram a especialização das ações do sistema a partir da definição das perspectivas onde essas ações atuam.

A formalização sintática e semântica da especificação dos casos de uso, proporcionada pelo UCMModel, é um dos elementos chave da abordagem proposta nesta tese, pois ela traz a possibilidade de explorar cenários relacionados à garantia da qualidade da especificação e do produto final, como:

- Definição de um apoio computacional para verificação automática de defeitos sintáticos nas especificações de casos de uso, ou seja violações das regras do metamodelo UCMModel;
- Elaboração de um conjunto de orientações para descrição de casos de uso, baseado na semântica dos elementos do UCMModel;
- Elaboração de regras de transformação modelo-modelo com o propósito de derivar modelos de testes funcionais a partir dos modelos de especificação.

Com relação ao primeiro cenário, uma ferramenta denominada UseCaseAgent foi definida e implementada com o objetivo de apoiar a construção e verificação sintática de especificações de casos de uso, segundo os conceitos preconizados pelo UCMModel, além de permitir a geração da descrição textual do caso de uso a partir do seu diagrama de atividades.

No segundo cenário, o conjunto de orientações definido a partir da semântica do UCMModel apoiou a elaboração de um conjunto de questões de uma técnica customizável de inspeção de diagramas de atividades baseada em *checklist*, chamada ActCheck (MELLO, 2011). Adicionalmente, a descrição textual do caso de uso, obtida a partir do diagrama de atividades, oferece a possibilidade do uso de técnicas de inspeção de casos de uso não especificamente projetadas a partir dos conceitos do metamodelo UCMModel, mas alinhadas conceitualmente com a estrutura de especificação preconizada por esse metamodelo. Como exemplo, duas técnicas desenvolvidas por pesquisadores brasileiros podem ser aplicadas nesse contexto: a técnica GUCCRA (BELGAMO e FABRI, 2004) e a técnica desenvolvida por GREGOLIN e DEBONI (2008), ambas avaliadas experimentalmente (BELGAMO et al., 2005; SANTOS e TRAVASSOS, 2010).

Para o terceiro cenário, uma outra ferramenta denominada ModelT² (ALBUQUERQUE *et al.*, 2010) também foi definida e implementada com o objetivo de derivar modelos de testes funcionais a partir dos modelos de especificação.

Dessa forma, o metamodelo UCMModel contribui para o objetivo geral deste trabalho, oferecendo um arcabouço conceitual a partir do qual técnicas e ferramentas de apoio à especificação e garantia da qualidade podem ser definidas e implementadas.

O capítulo 6 apresenta o apoio computacional elaborado com o objetivo de apoiar a abordagem de especificação e garantia da qualidade proposta neste trabalho. Detalhes sobre a utilização das ferramentas UseCaseAgent e ModelT² também são apresentados nesse capítulo.

6 Infra-estrutura de Apoio à Especificação de Casos de Uso e Garantia da Qualidade

Neste capítulo é apresentada infra-estrutura computacional organizada para apoiar a abordagem de especificação e garantia da qualidade proposta nesta pesquisa e detalha as duas ferramentas desenvolvidas no escopo deste trabalho que compõem essa infra-estrutura.

6.1 Introdução

A infra-estrutura computacional organizada para apoiar a abordagem proposta neste trabalho é composta de quatro ferramentas que atuam em momentos distintos do processo de desenvolvimento. Dessas quatro ferramentas duas foram desenvolvidas no escopo desta pesquisa:

- **Ferramenta UseCaseAgent:** tem o objetivo de apoiar a construção da especificação dos casos de uso, garantindo que essas especificações sigam o metamodelo UCModel e as restrições nele definidas. Como insumo, a ferramenta recebe o resultado da análise das informações coletadas durante a elicitacão dos requisitos. Em conseqüência, ela gera um conjunto de diagramas de atividades que representam o modelo de casos de uso, criado de acordo com o metamodelo UCModel.
- **Ferramenta ModelT²:** é usada após a conclusão das atividades de engenharia de requisitos e tem por objetivo gerar um modelo preliminar de testes funcionais a partir do modelo de casos de uso. Como insumo a ferramenta recebe um diagrama de atividades que representa a especificação de um caso de uso baseada no UCModel e gera, como resultado, um outro diagrama de atividades que representa o caso de uso do ponto de vista do teste funcional.

As outra duas ferramentas que completam a infra-estrutura são:

- **Ferramenta BOUML (PAGÉS, 2011):** é uma ferramenta UML usada na construção dos diagramas UML. Foi selecionada porque é uma ferramenta open-source, implementa quase toda a especificação UML 2 (OMG, 2010a),

roda em diversas plataformas (Windows©, MacOS X© e distribuições Linux) e permite a construção de *plug-ins* para estender as suas funcionalidades.

- **Ferramenta TDE/UML®**: pertence à Siemens Corporation Research/USA e é usada para geração dos casos e procedimentos de teste a partir dos modelos de teste. Sua escolha se deu por oportunidade e conveniência, devido a facilidade de acesso à ferramenta em virtude da colaboração existente entre o grupo de Engenharia de Software Experimental e a Siemens Corporation Research/USA. É importante frisar que tanto a ferramenta TDE/UML® quanto a abordagem de modelagem preconizada por ela são efetivamente usadas em projetos reais dessa organização.

A Figura 6-1 apresenta o ciclo de uso completo das ferramentas previstas na abordagem proposta nesta tese, relacionando esse uso com as atividades do processo de especificação no qual a abordagem está inserida. O uso das ferramentas acontece em quatro etapas:

- 1) **Etapa 1**: nesta etapa o desenvolvedor usa a ferramenta UseCaseAgent para criar as especificação dos casos de uso, através de diagramas de atividades aderentes ao metamodelo UCModel. Também nesta etapa as especificações são validadas segundo as restrições do UCModel;
- 2) **Etapa 2**: nessa etapa o analista de testes gera os modelos preliminares de testes, a partir dos modelos de caso de uso criados na etapa anterior, usando a ferramenta ModelT². Esses modelos são considerados preliminares porque ainda carecem de um conjunto de informações que possibilite a geração dos casos e procedimentos de teste;
- 3) **Etapa 3**: nesta etapa o analista de testes edita os modelos de teste com a ferramenta BOUML, analisa-os e insere as informações faltantes ou altera informações existentes para que o modelo esteja completo o suficiente para permitir a geração dos casos e procedimentos de testes, e;
- 4) **Etapa 4**: aqui o analista de testes usa a ferramenta TDE/UML® para gerar, automaticamente, os casos e procedimentos de teste a partir dos modelos de teste.

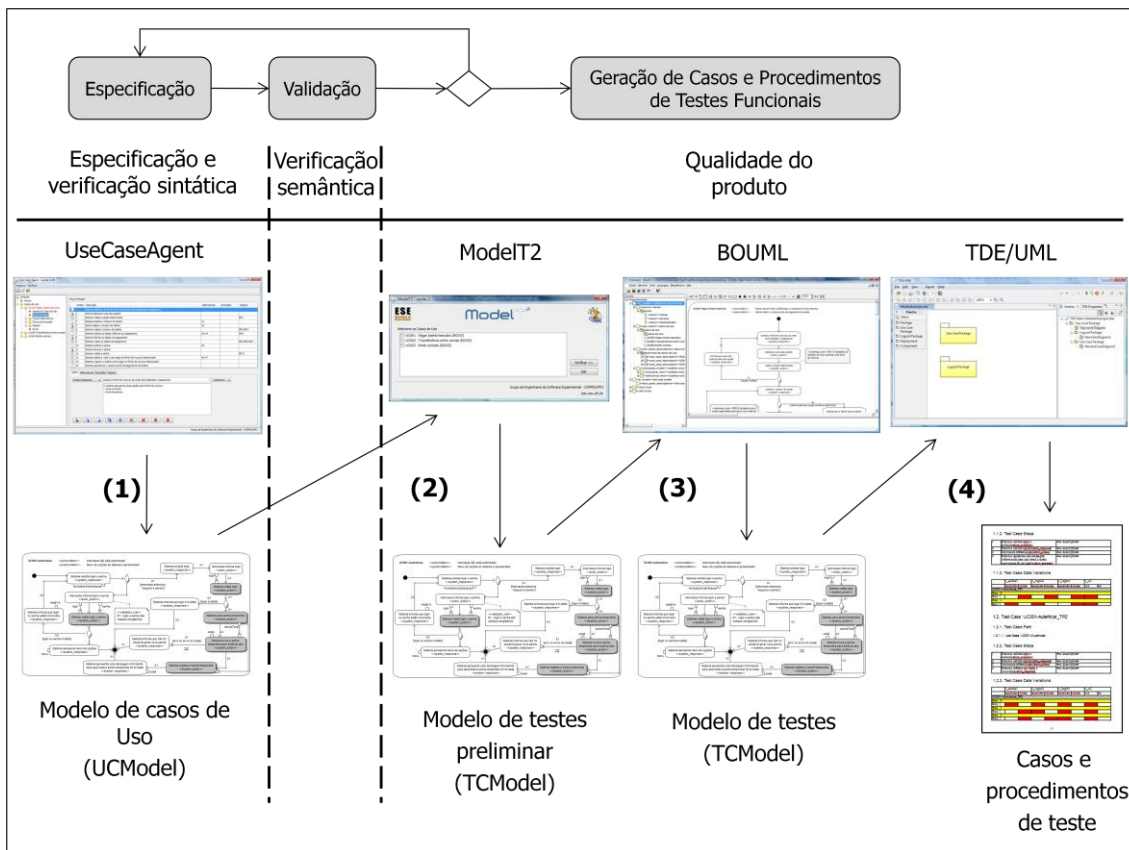


Figura 6-1 – Ciclo de uso das ferramentas previstas na abordagem com insumos e produtos

A seção 6.2, a seguir, descreve em detalhes a ferramenta UseCaseAgent, usada na etapa 1 e a seção 6.3 apresenta a ferramenta ModelT² e discute as atividades relativas às etapas 2, 3 e 4.

6.2 Ferramenta UseCaseAgent

A especificação de casos de uso de acordo com o metamodelo UCMoel demanda o uso de uma ferramenta UML capaz de:

- Criar diagramas de atividades de acordo com a especificação UML 2, e;
- Trabalhar com perfis UML, onde estão definidos os estereótipos e *tagged-values* do metamodelo UCMoel.

Entretanto, a criação de diagramas de atividades nesse contexto não contemplaria a validação da sua aderência ao metamodelo UCMoel, pois a ferramenta UML simplesmente aplicaria os estereótipos aos elementos do diagrama, mas não teria condições de validar a sua estrutura frente às restrições estabelecidas pelo UCMoel. Assim, foi identificada a necessidade de um apoio ferramental para suprir essa lacuna.

Em um primeiro momento, a funcionalidade básica do UseCaseAgent seria somente a de permitir a verificação da aderência de diagramas de atividades criados

com o perfil ProfileUCModel ao metamodelo UCMModel, ou seja, o desenvolvedor usaria o editor da ferramenta UML para criar o digrama de atividades com o perfil ProfileUCModel e, posteriormente, usaria a ferramenta UseCaseAgent para verificar a aderência de tais diagramas ao metamodelo UCMModel. Como clara desvantagem desse arranjo está a possibilidade de criar diagramas não aderentes ao UCMModel, pois ficaria a cargo do desenvolvedor acionar ou não a verificação dos modelos gerados. Em virtude disso, ao invés de implementar somente a funcionalidade de verificação do diagrama, optou-se por incluir uma funcionalidade para criação/edição do caso de uso em formato texto integrando a verificação de aderência ao UCMModel e a geração automática do diagrama de atividades a partir dessa descrição textual.

Assim, as principais funcionalidades implementadas na ferramenta UseCaseAgent são:

1. Criação das especificações dos casos de uso de acordo com o metamodelo UCMModel usando uma abordagem textual;
2. Verificação automática das restrições previstas no metamodelo UCMModel, ou seja, verificação sintática da especificação do caso de uso;
3. Geração automática do diagrama de atividades correspondente ao caso de uso especificado;
4. Edição do diagrama de atividades, ou seja, da especificação do caso de uso, usando uma abordagem textual, e;
5. Geração da especificação do caso de uso em formato texto (padrão RTF - *Rich Text Format*), conforme os elementos previstos no gabarito de casos de uso apresentado na Tabela 4-1 (página 61).

As funcionalidades 1, 2 e 4 tem o objetivo de dar mais flexibilidade ao desenvolvedor, pois coloca à sua disposição duas abordagens distintas para especificação dos casos de uso:

1. Gráfica: através de um editor de diagrama de atividades de uma ferramenta case UML com o apoio do perfil ProfileUCModel.
2. Textual: através da ferramenta UseCaseAgent, com a geração automática do diagrama de atividades.

Na primeira opção o desenvolvedor cria o elemento desejado diretamente no diagrama de atividades, aplica um dos estereótipos disponíveis no perfil UML ProfileUCModel e define o valor de cada *tagged-value* associada ao estereótipo.

Na segunda opção o desenvolvedor usa o editor de casos de uso da ferramenta UseCaseAgent, que oferece flexibilidade semelhante a um editor de textos

comum, porém direcionando a criação da especificação de acordo com os elementos do UCMModel.

Para atender às funcionalidades especificadas e à necessidade de integração da UseCaseAgent com a ferramenta UML, foi selecionada a ferramenta BOUML (PAGÉS, 2011) como ferramenta UML. A escolha da BOUML está baseada nas seguintes características e recursos disponíveis:

- É uma ferramenta *open-source*;
- Executável em várias plataformas (Windows®, MacOS X®, distribuições Linux);
- Implementa a especificação da UML 2;
- Trabalha com perfis UML, e;
- É extensível através de *plug-ins* escritos em C++ ou Java.

A ferramenta UseCaseAgent pode ser classificada como um editor orientado a formulários e foi implementada na linguagem Java como um *plug-in* para a ferramenta BOUML. A arquitetura oferecida pela BOUML permite que seus *plug-ins* leiam, criem, alterem ou excluam modelos ou elementos associados a esses modelos programaticamente. Dessa forma, a UseCaseAgent é capaz de manipular os diagramas de atividades da BOUML, criando ou editando esses diagramas segundo o metamodelo UCMModel e suas restrições.

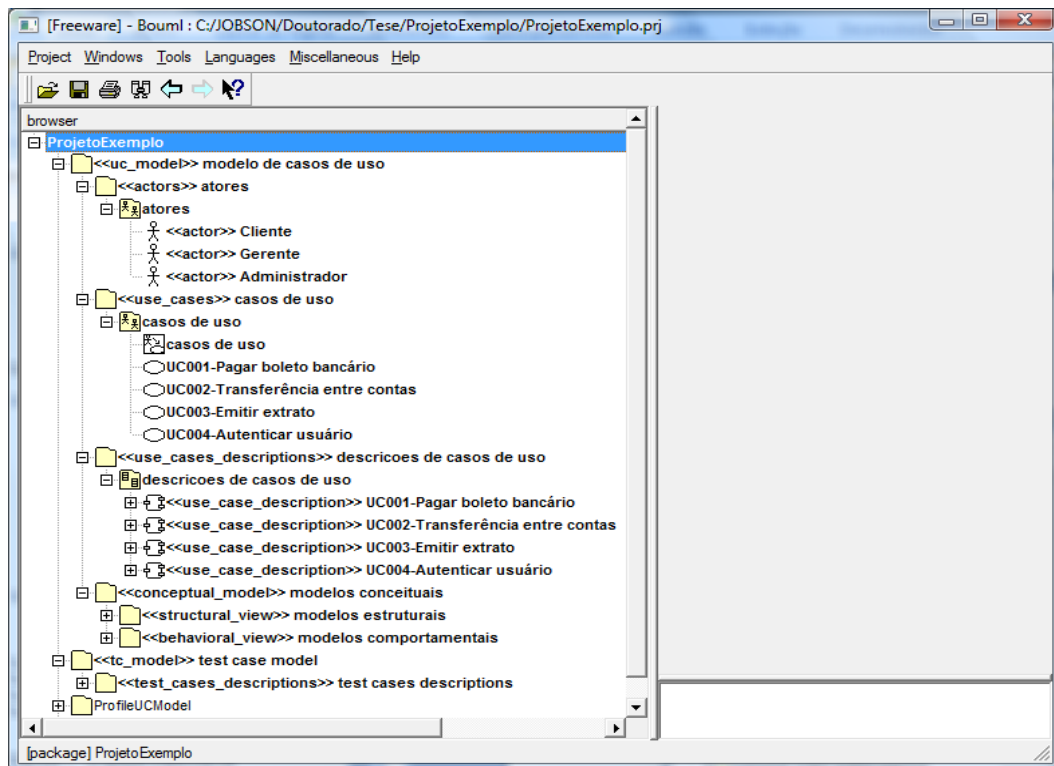


Figura 6-2 - Tela principal da ferramenta BOUML com a estrutura de pacotes

Para que as especificações de casos de uso criadas com a UseCaseAgent pudessem ser disponibilizadas na BOUML como diagramas de atividades, foi organizada uma estrutura de pacotes para armazenamento dessas informações (atores, casos de uso e diagramas de atividades com as especificações dos casos de uso). A Figura 6-2 apresenta a tela principal da ferramenta BOUML destacando os pacotes que organizam os atores, casos de uso e diagramas de atividades.

A seção a seguir irá descrever um cenário de especificação de um caso de uso onde as cinco funcionalidades da UseCaseAgent serão exploradas.

6.2.1 Cenários de uso da UseCaseAgent

O cenário de exemplo retrata a funcionalidade de pagamento de boleto bancário usando um sistema de *home banking*. Será criado um caso de uso chamado “Pagar boleto bancário”

6.2.1.1 Criação do Caso de Uso

Para criar o caso de uso basta selecionar o pacote-raiz estereotipado `<<uc_model>>` e, em seguida, selecionar a opção *Tools->Create Use Case*. O UseCaseAgent será executado e será apresentada a tela da Figura 6-3 para entrada das informações básicas sobre o caso de uso. Ao clicar nos itens do painel, à esquerda da tela, é possível navegar pelos atores do caso de uso, fluxo principal, fluxos alternativos e de exceção, regras que devem ser observadas e erros de verificação do caso de uso. Nessa tela destacam-se também as opções de salvar o caso de uso (isto é, gerar o diagrama de atividades a partir da descrição) e verificar o caso de uso.

A Figura 6-4 apresenta a tela para definição dos atores associados ao caso de uso. É possível selecionar um ou mais atores, que ficarão disponíveis no momento de criar os fluxos dos casos de uso. Neste exemplo, somente o ator “Cliente” foi selecionado.

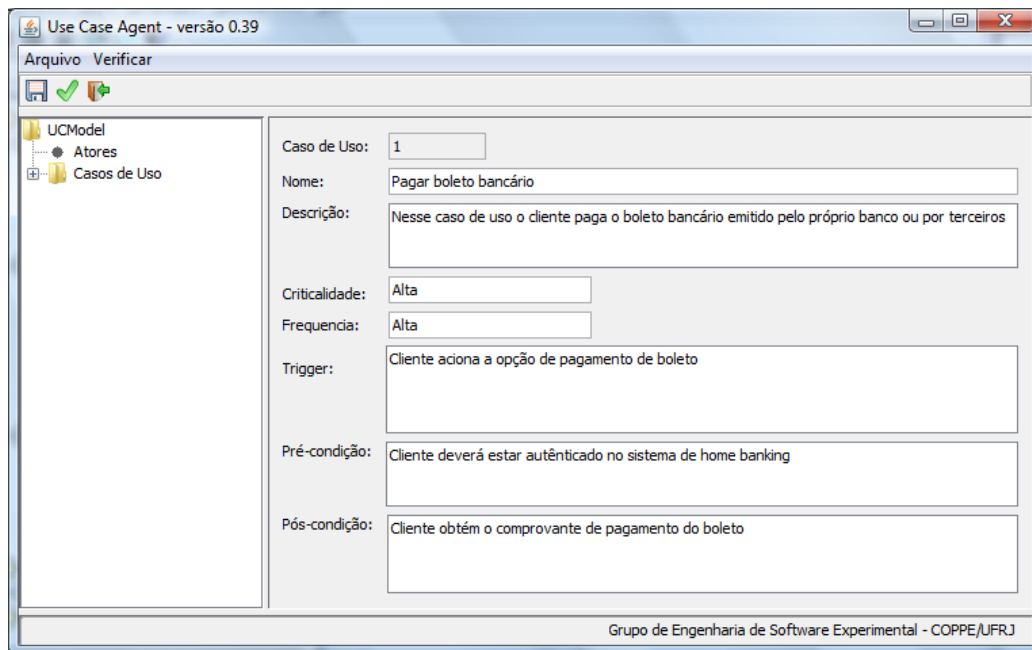


Figura 6-3 - Tela inicial da UseCaseAgent com os elementos básicos do caso de uso

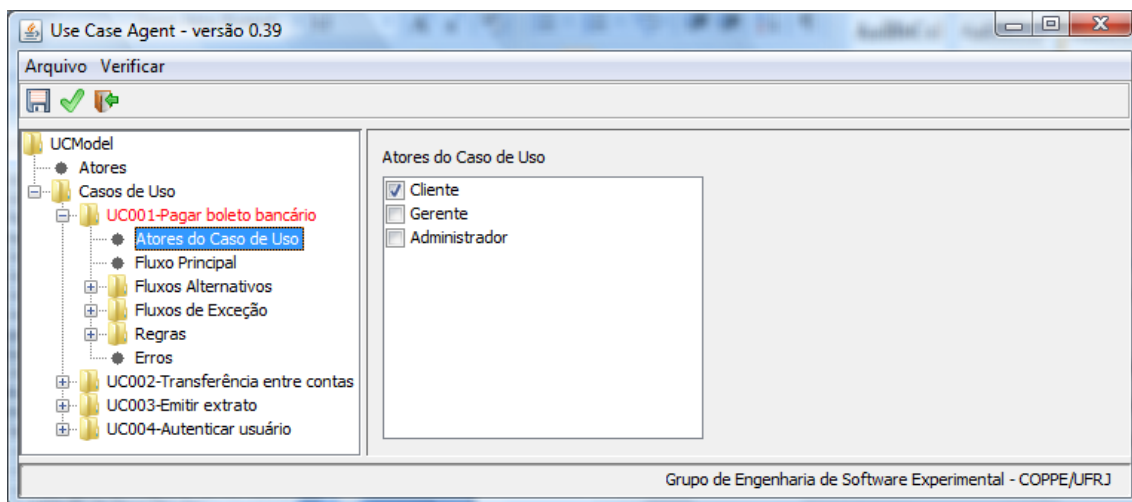


Figura 6-4 - Tela da UseCaseAgent para definição dos atores do caso de uso

As Figura 6-5 e 6-6 apresentam as tela dos UseCaseAgent para definição do fluxo principal e fluxo alternativo A2, respectivamente. Nessa telas é possível criar a seqüência de ações que compõem esses fluxo. Para cada ação é possível definir:

- Tipo do ação
- Descrição da ação
- Complemento da ação
- Alternativas e exceções
- Regras associadas à ação

Para definição dos fluxos alternativos e de exceção a tela é idêntica a da Figura 6-5, porém com a inclusão da condição que dispara o fluxo (Figura 6-6).

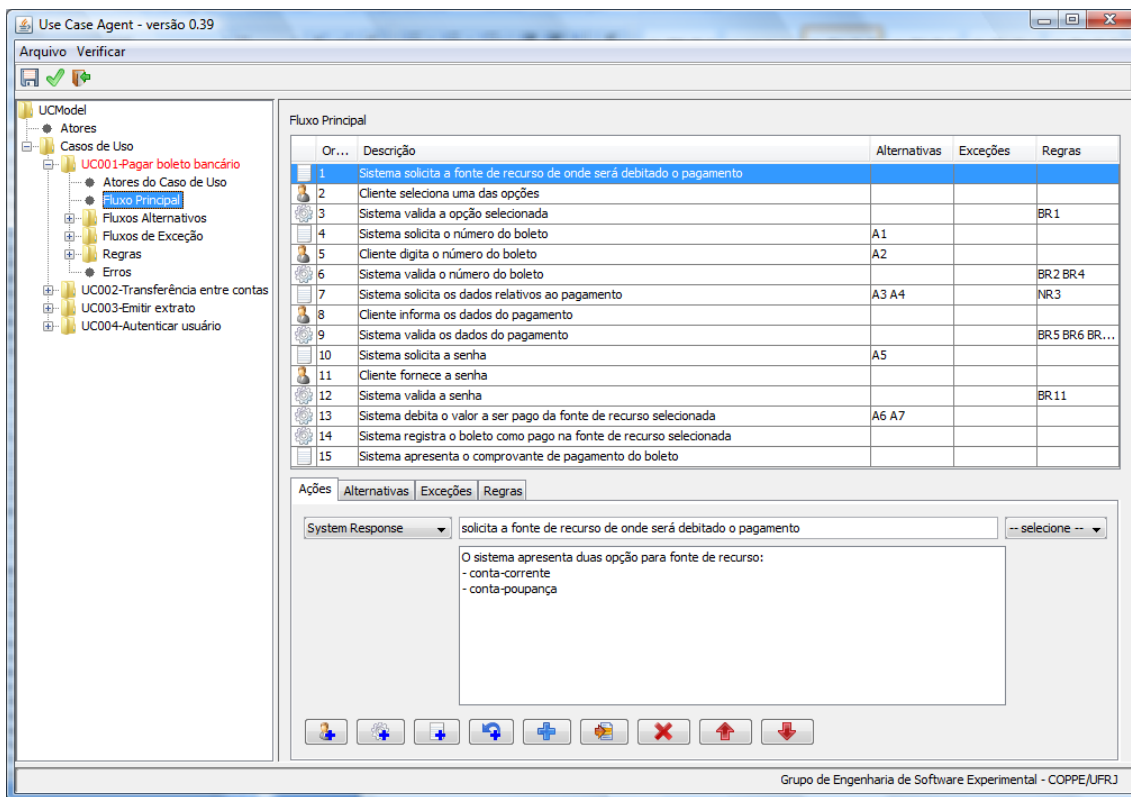


Figura 6-5 - Tela da UseCaseAgent com o fluxo principal do caso de uso "Pagar boleto bancário"

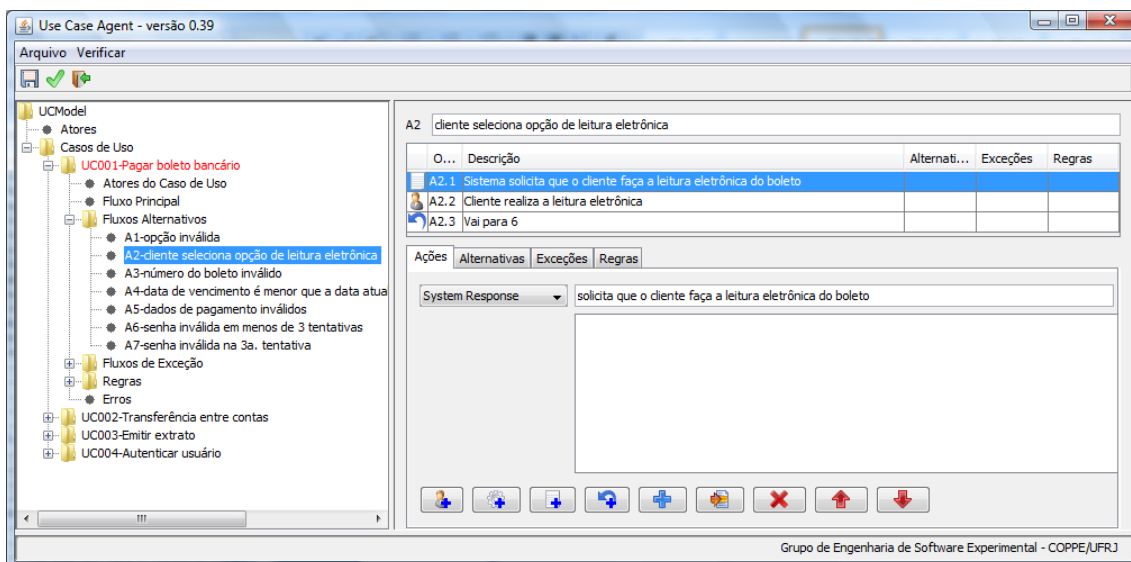


Figura 6-6 - Tela da UseCaseAgent com o fluxo alternativo A2 do caso de uso "Pagar boleto bancário"

A Figura 6-7 apresenta a tela para definição das regras a serem observadas no escopo do caso de uso. Cada uma das regras deve ser definida como regra de conceituação, navegação ou apresentação.

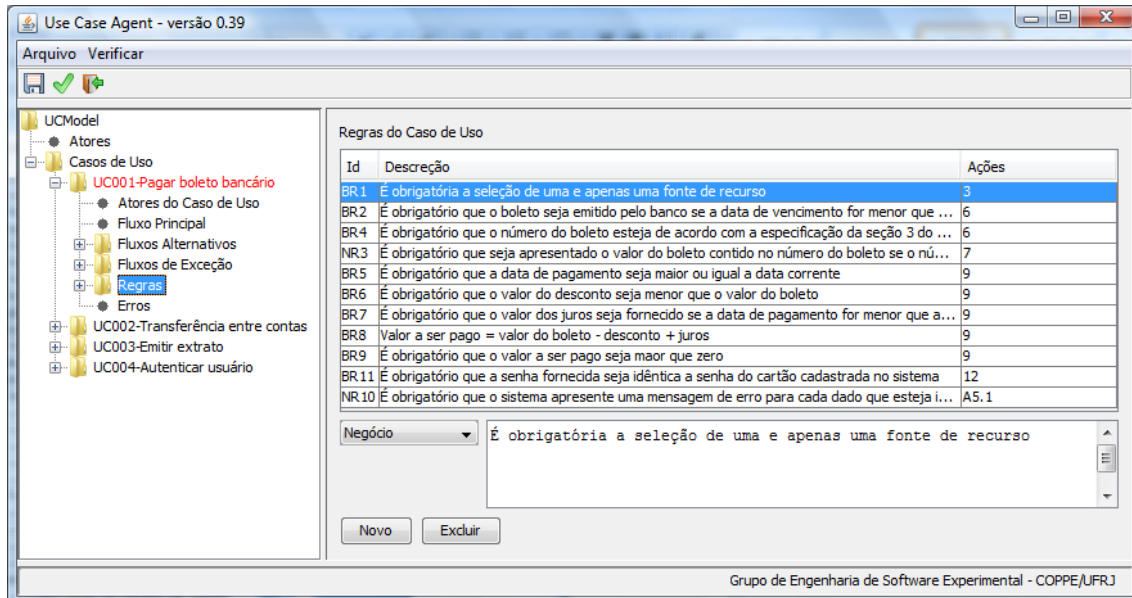


Figura 6-7 - Tela da UseCaseAgent com as regras do caso de uso "Pagar boleto bancário"

6.2.1.2 Verificação do Caso de Uso

A qualquer momento da criação do caso de uso é possível acionar a opção de verificação dessa especificação. A verificação avalia a especificação de acordo com as restrições definidas na seção 5.3.6 (página 114). Para cada defeito detectado é apresentado o local e a posição do mesmo. A Figura 6-8 apresenta a tela com a lista de defeitos detectados para o caso de uso usado como exemplo.

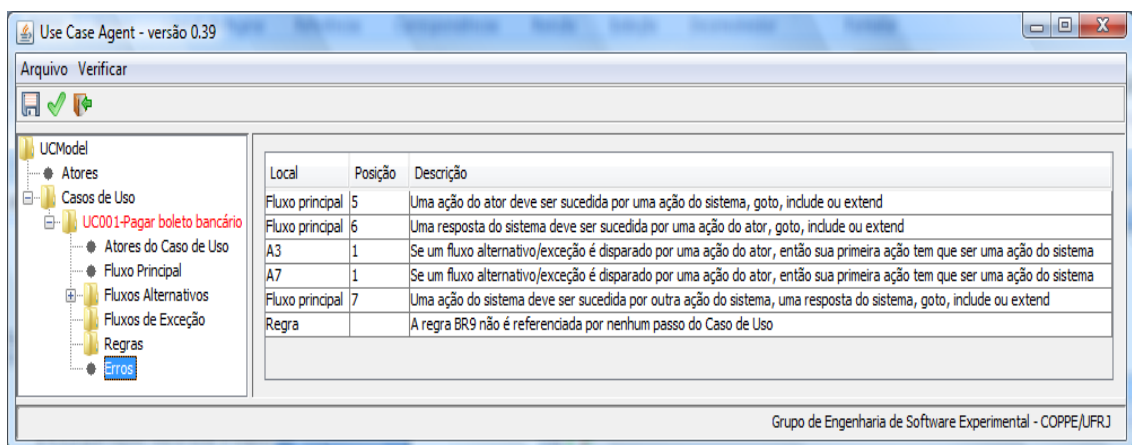


Figura 6-8 - Tela da UseCaseAgent com a lista de defeitos do caso de uso "Pagar boleto bancário"

6.2.1.3 Geração do Diagrama de Atividades

Para gerar o diagrama de atividades correspondente à especificação do caso de uso, basta salvar essa especificação. Antes de realizar a geração propriamente dita, o UseCaseAgent aciona a opção de verificação do caso de uso. Caso algum defeito seja detectado a operação de geração do diagrama é cancelada e o desenvolvedor tem a oportunidade de analisar e eliminar os defeitos existentes. Ou seja, a geração do diagrama só é realizada caso a especificação não contenha nenhum defeito. A Figura 6-9 mostra o diagrama de atividades gerado automaticamente a partir da especificação do caso de uso de exemplo “Pagar boleto bancário” (somente a diagramação do modelo foi realizada manualmente).

6.2.1.4 Edição do Caso de Uso

Uma vez criado o diagrama de atividades relativo à especificação do caso de uso, conforme exemplo da Figura 6-9, é possível editá-lo diretamente através da ferramenta BOUML, ou seja, incluir, excluir ou alterar elementos do diagrama, ou editá-lo com a UseCaseAgent. Na segunda opção, as telas apresentadas anteriormente para criação da especificação do caso de uso (Figura 6-3 à Figura 6-8) são as mesmas usadas na edição dessa especificação.

6.2.1.5 Geração da Especificação em Formato texto

Após a especificação de um ou mais casos de uso usando a ferramenta UseCaseAgent, é possível gerar um documento de especificação, em formato RTF, contendo um ou mais casos de uso. Para obter essa especificação basta selecionar o pacote-raiz estereotipado `<<uc_model>>` e, em seguida, selecionar a opção *Tools->Print Use Case*. Em seguida será apresentada a tela da Figura 6-10, onde o desenvolvedor poderá selecionar quais casos de uso farão parte do documento de especificação. Ao clicar em verificar, os casos de uso selecionados serão verificados quanto à sua aderência ao metamodelo UCMModel. A tela apresentada na Figura 6-11 mostra a situação de cada caso de uso após a verificação e permite a seleção do documento onde a especificação será gerada em formato RTF. Somente os casos de uso que atenderem à todas as restrições do metamodelo UCMModel farão parte do documento de especificação.

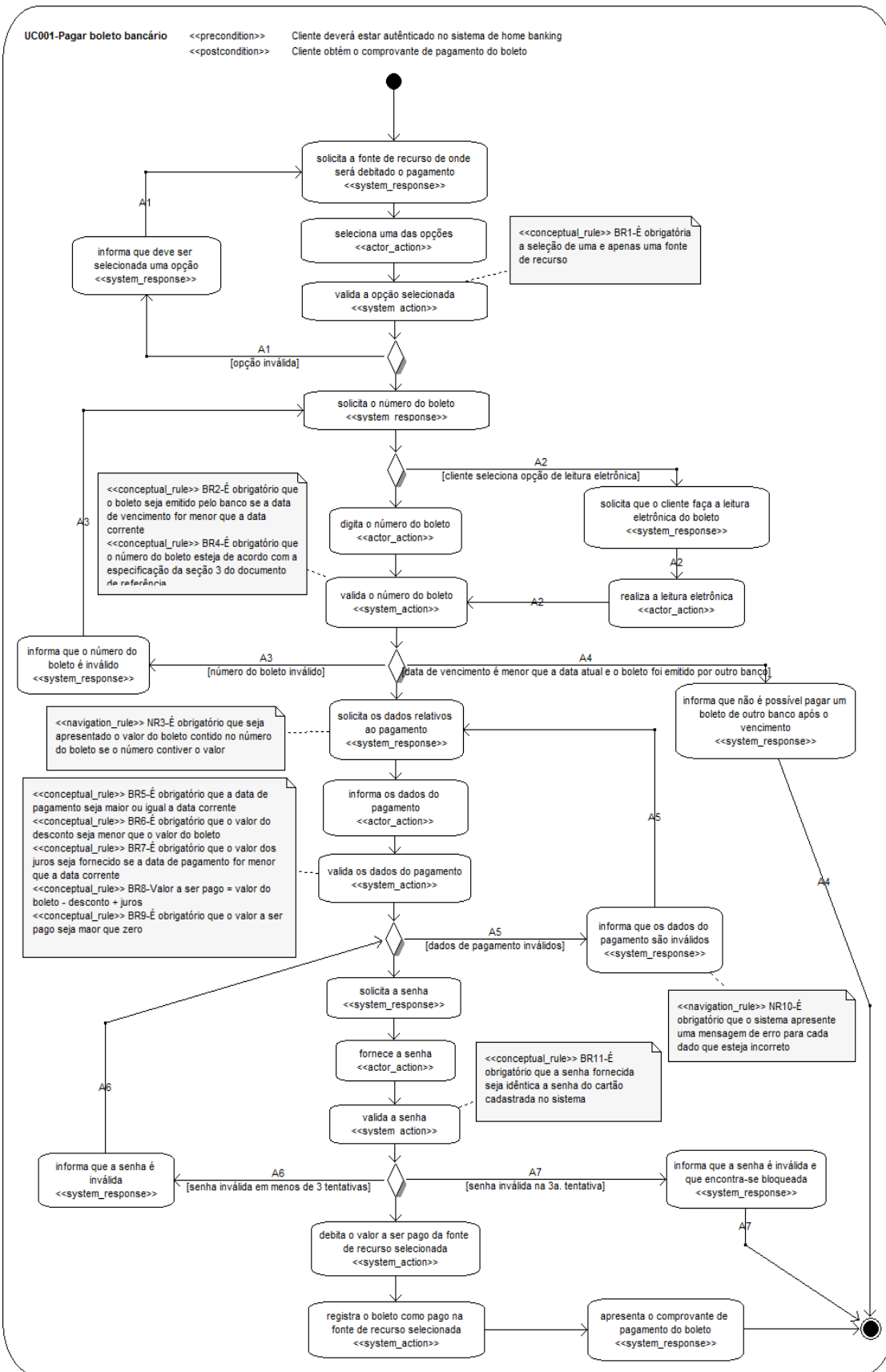


Figura 6-9 - Diagrama de atividades com a especificação do caso de uso "Pagar boleto bancário"

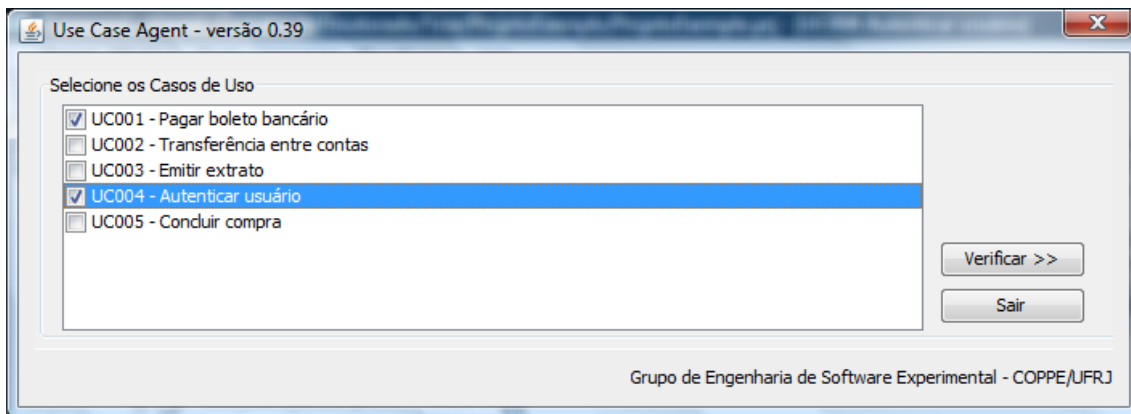


Figura 6-10 - Tela da UseCaseAgent para seleção dos casos de uso que farão parte do documento de especificação

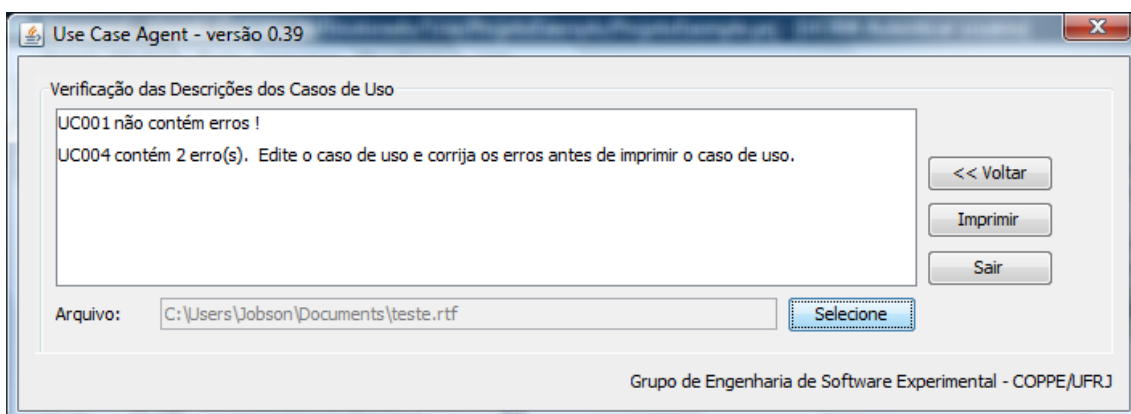


Figura 6-11 - Tela da UseCaseAgent para impressão das especificações de casos de uso em formato RTF

Finalmente, a Figura 6-12 apresenta a especificação do caso de uso “Pagar boleto bancário” gerada pelo UseCaseAgent a partir do diagrama de atividades que especifica esse caso de uso.

UC001 – Pagar boleto bancário

Objetivo:	Nesse caso de uso o cliente paga o boleto bancário emitido pelo próprio banco ou por terceiros
Atores:	Cliente
Criticalidade:	Alta
Frequência de uso:	Alta
Pré-condições:	Cliente deverá estar autenticado no sistema de home banking
Invariantes:	Não há
Pós-condições:	Cliente obtém o comprovante de pagamento do boleto
Trigger:	Cliente aciona a opção de pagamento de boleto
Fluxo Principal:	1. Sistema solicita a fonte de recurso de onde será debitado o pagamento O sistema deve apresentar duas opção para fonte de recurso: - conta-corrente - conta-poupança

2. Cliente seleciona uma das opções
 3. Sistema valida a opção selecionada [BR1]
 4. Sistema solicita o número do boleto [A1]
 5. Cliente digita o número do boleto [A2]
 6. Sistema valida o número do boleto [BR2, BR4]
 7. Sistema solicita os dados relativos ao pagamento [A3, A4] [NR3]
- Os dados a serem solicitados são:
- data de pagamento
 - valor do boleto
 - valor do desconto
 - juros
 - valor a ser pago
8. Cliente informa os dados do pagamento
 9. Sistema valida os dados do pagamento [BR5, BR6, BR7, BR8, BR9]
 10. Sistema solicita a senha [A5]
 11. Cliente fornece a senha
 12. Sistema valida a senha [BR11]
 13. Sistema debita o valor a ser pago da fonte de recurso selecionada [A6, A7]
 14. Sistema registra o boleto como pago na fonte de recurso selecionada
 15. Sistema apresenta o comprovante de pagamento do boleto
- Os dados do comprovante são:
- nome do correntista
 - data e hora do pagamento
 - numero da conta-corrente ou conta-poupança
 - numero do boleto
 - data de vencimento
 - valor do boleto
 - valor do desconto
 - juros
 - valor pago

Fluxos Alternativos: [A1] opção inválida

- A1.1. Sistema informa que deve ser selecionada uma opção
- A1.2. Vai para 1

[A2] cliente seleciona opção de leitura eletrônica

- A2.1. Sistema solicita que o cliente faça a leitura eletrônica do boleto
- A2.2. Cliente realiza a leitura eletrônica
- A2.3. Vai para 6

[A3] número do boleto inválido

- A3.1. Sistema informa que o número do boleto é inválido
- A3.2. Vai para 4

[A4] data de vencimento é menor que a data atual e o boleto foi emitido por outro banco

- A4.1. Sistema informa que não é possível pagar um boleto de outro banco após o vencimento
- A4.2. Encerra o caso de uso

[A5] dados de pagamento inválidos

- A5.1. Sistema informa que os dados do pagamento são inválidos [NR10]
- A5.2. Vai para 7

[A6] senha inválida em menos de 3 tentativas

- A6.1. Sistema informa que a senha é inválida
- A6.2. Vai para 10

[A7] senha inválida na 3a. tentativa

A7.1. Sistema informa que a senha é inválida e que encontra-se bloqueada

Deve ser apresentada uma mensagem para que o cliente compareça à agência bancária

A7.2 Encerra o caso de uso

Fluxos de Exceção: Não há

Regras:

- BR1 - É obrigatória a seleção de uma e apenas uma fonte de recurso
- BR2 - É obrigatório que o boleto seja emitido pelo banco se a data de vencimento for menor que a data corrente
- BR4 - É obrigatório que o número do boleto esteja de acordo com a especificação da seção 3 do documento de referência
- NR3 - É obrigatório que seja apresentado o valor do boleto contido no número do boleto se o número contiver o valor
- BR5 - É obrigatório que a data de pagamento seja maior ou igual a data corrente
- BR6 - É obrigatório que o valor do desconto seja menor que o valor do boleto
- BR7 - É obrigatório que o valor dos juros seja fornecido se a data de pagamento for menor que a data corrente
- BR8 - Valor a ser pago = valor do boleto - desconto + juros
- BR9 - É obrigatório que o valor a ser pago seja maior que zero
- BR11 - É obrigatório que a senha fornecida seja idêntica a senha do cartão cadastrada no sistema
- NR10 - É obrigatório que o sistema apresente uma mensagem de erro para cada dado que esteja incorreto

Figura 6-12 - Representação em formato texto da especificação do caso de uso "Pagar boleto bancário"

A combinação das funcionalidades da ferramenta UseCaseAgent com um editor de diagramas de atividades (nesse caso, a ferramenta BOUML), oferece aos desenvolvedores três opções para especificação dos casos de uso:

- Criação/edição das especificações usando somente a ferramenta BOUML, ou seja, o caso de uso é manipulado somente graficamente através do diagrama de atividades;
- Criação/edição das especificações de casos de uso usando somente a UseCaseAgent, ou seja, o caso de uso é manipulado usando somente o formato textual, e;
- Criação/edição das especificações usando uma abordagem híbrida onde o caso de uso pode ser manipulado tanto no formato texto (UseCaseAgent) quanto no formato gráfico (BOUML).

A princípio, a preferência do desenvolvedor e a sua habilidade e familiarização com uma ou outra abordagem é que vão determinar a opção a seguir. Duas características comuns que acompanham essas três opções são:

- Sempre é possível gerar a especificação do caso de uso no formato texto, o que pode ser útil ao lidar com *stakeholders*, ou até desenvolvedores, não familiarizados com a sintaxe e a semântica dos diagramas de atividades, e;
- O metamodelo UCModel é a base de apoio da especificação do caso de uso, independente do formato em que ela se materialize.

Outro recurso disponível é a possibilidade de usar a ferramenta UseCaseAgent para criar as especificações dos casos de uso em modo texto e, posteriormente, gerar essas especificações também em modo texto. Nesse caso, em momento algum o desenvolvedor ou outros *stakeholders* tratam diretamente os diagramas de atividades, apesar da sua existência apoiar a criação e verificação da especificação. Essa flexibilidade pode ser útil ao lidar com *stakeholders* não familiarizados com a sintaxe e a semântica dos diagramas de atividades ou mesmo ser utilizada por desenvolvedores que preferem trabalhar com o estilo textual de representação

6.2.2 Trabalhos Relacionados

A partir dos trabalhos apresentados no capítulo 5 que relatam abordagens para modelagem da especificação de casos de uso, foram analisadas as ferramentas que apóiam essas abordagens. As seções a seguir descrevem um breve resumo de cada uma delas.

6.2.2.1 SCORESTOOL

SCORESTOOL (KOSTER *et al.*, 2001) é uma ferramenta que automatiza o método de modelagem do domínio chamado SCORES, que se baseia, dentre outras características, no refinamento de casos de uso para especificação do comportamento do sistema. A especificação dos casos de uso é realizada através de um diagrama de atividades. A ferramenta não trabalha com um metamodelo, somente com um conjunto de estereótipos que define os tipos possíveis que uma ação do caso de uso pode assumir: ação do sistema, ação do ator e ação do ator fora do escopo do sistema. Não é realizado nenhum tipo de verificação (semi) automática do diagrama de atividades produzido, pois o processo de garantia da qualidade dos modelos produzidos é totalmente manual.

6.2.2.2 REM (REquirements Manager)

A REM (DURAN *et al.*, 2002) é uma ferramenta para gerenciamento de requisitos que engloba a sua classificação e especificação. Casos de uso são classificados como requisitos do cliente e a sua especificação é apoiada por um

metamodelo que classifica as ações em ação do ator ou ação do sistema. A ferramenta REM também utiliza a abordagem de formulários para capturar as informações do caso de uso: nome, descrição, importância, frequência de uso, *trigger*, ações e exceções (fluxos alternativos). As descrições das ações é realizada em linguagem natural e a verificação da especificação é realizada através da aderência da especificação ao metamodelo e de um conjunto de heurísticas que tratam do balanceamento do número de ações do ator e ações do sistema em um caso de uso, da complexidade ciclomática do caso de uso, dentre outras. A descrição do caso de uso não é representada usando nenhum modelo, somente texto.

6.2.2.3 UCEd

É uma ferramenta para modelagem de casos de uso que implementa o metamodelo descrito em SOMÉ (2006) e SOMÉ (2009). O UCEd é um editor orientado a formulários que captura um conjunto de informações acerca do caso de uso, a saber: título, descrição, ator primário, pré e pós-condições e *trigger*, fluxo principal e fluxos alternativos. Com relação às ações do caso de uso, o UCEd captura essas ações em linguagem natural e verifica se estas seguem uma sintaxe pré-definida que envolve substantivos (entidades do domínio) e verbos (operações relacionadas à entidades do domínio). A partir da descrição do caso de uso, a ferramenta UCEd pode gerar uma máquina de estados (OMG, 2010a) onde os estados são definidos pelos valores dos atributos das entidades do domínio e as transições representam as operações, isto é, os verbos usados nas descrições das ações.

6.2.2.4 Ferramenta de GUTIÉRREZ *et al.* (2008)

GUTIÉRREZ *et al.* (2008) propõem uma ferramenta, cujo nome não é citado no referido artigo, para transformação de descrições textuais em diagramas de atividades. Está previsto que a descrição textual esteja armazenada em um arquivo XML, de acordo com um esquema pré-definido, usando linguagem natural para descrição dos seus diversos elementos (nome, descrição, fluxo principal e alternativos). A ferramenta se limita, então, a realizar a transformação deste arquivo em um diagrama de atividades. O metamodelo que apóia essa transformação é o próprio esquema do arquivo XML e o metamodelo da UML. Na realidade existe uma correlação quase direta entre os elementos do arquivo XML e os elementos do diagrama de atividades, tornando a transformação uma tarefa trivial. Não há nenhum tipo de verificação realizada sobre a especificação do caso de uso, pois o XML usado como entrada para o processo de transformação já está construído de acordo com o esquema pré-definido. Também não há nenhum tipo de classificação para as ações

que descrevem o caso de uso, ou seja, todas são tratadas segundo a mesma perspectiva.

A partir dessa análise é possível destacar três diferenciais proporcionados pela ferramenta UseCaseAgent em relação às descritas anteriormente:

- Permite a definição da especificação em um modelo híbrido, onde o desenvolvedor pode optar por especificar o caso de uso diretamente com o diagrama de atividades (abordagem gráfica) ou usando uma descrição textual (abordagem textual), a partir da qual o diagrama de atividades é gerado;
- Permite a verificação da aderência da especificação do caso de uso ao metamodelo UCMModel (verificação sintática), independente da abordagem (gráfica ou textual) utilizada na criação da especificação, e;
- Permite a geração da descrição textual da especificação, independente da abordagem (gráfica ou textual) utilizada na criação da especificação.

6.3 Ferramenta ModelT²

A ferramenta ModelT² (*Model Transformations applied to Test*) foi desenvolvida com o objetivo de apoiar a definição do plano de testes funcionais explorando a abordagem de Testes Baseados em Modelos (MBT – *Model Based Testing*) (APFELBAUM e DOYLE, 1997) a partir de diagramas de atividades. Para tal, sua principal funcionalidade é a derivação dos modelos de testes funcionais a partir dos diagramas de atividades que representam as especificações dos casos de uso. Posteriormente, o modelo de testes é usado na geração dos casos e procedimentos de testes funcionais.

Para construção do modelo de testes foi adotado o metamodelo TDE (HARTMANN et al., 2005). A escolha desse metamodelo TDE se deu por oportunidade e conveniência, em virtude do conhecimento prévio de membros do grupo de Engenharia de Software Experimental (ESE) da UFRJ sobre esse modelo e à colaboração existente entre o grupo ESE e a Siemens Corporation Research/USA.

A primeira versão da ferramenta ModelT² foi implementada como uma transformação XSLT¹³, onde os diagramas de entrada e saída eram armazenadas em arquivos XMI (ALBUQUERQUE *et al.*, 2010). Essa versão foi usada para avaliar a viabilidade da derivação do modelo de testes no padrão TDE a partir do modelo de

¹³ WWW.w3.org/TR/xslt

casos de uso descritos usando o UCMModel, mas tinha a desvantagem de obrigar o desenvolvedor a realizar uma série de passos para geração e tratamento dos arquivos XMI. Na versão atual, a ferramenta ModelT² foi implementada como um *plug-in* da ferramenta BOUML, sendo assim capaz de ler e gerar modelos diretamente nessa ferramenta.

Serão apresentadas, a seguir, as três etapas desse processo usando como exemplo o caso de uso “Pagar boleto bancário” (Figura 6-9, página140).

6.3.1.1 Derivação do Modelo de Testes

Essa etapa ocorre após o término das atividades relacionadas à engenharia de requisitos, ou seja, após a conclusão da especificação e validação dos casos de uso. O analista de testes utiliza a ferramenta ModelT² para derivar o modelo de testes preliminar a partir dos diagramas de atividades que representam as especificações dos casos de uso. Esse modelo de testes é chamado de preliminar porque, no momento em que é gerado, ainda não está pronto para ser usado na geração dos casos e procedimentos de testes, pois algumas informações relacionadas exclusivamente aos testes funcionais não podem ser derivadas a partir da especificação de casos de uso, como, por exemplo, os valores ou categorias de valores a serem usados nos testes funcionais.

O metamodelo TDE está baseado em conceitos bastante semelhantes aos conceitos existentes no metamodelo UCMModel. As principais diferenças do metamodelo TDE em relação ao metamodelo UCMModel:

- 1) Ele só possui ações do tipo *ActorAction* e *SystemResponse*, ou seja, ele não modela a ação do sistema. A explicação está no fato de que TDE modela o sistema como uma caixa-preta e, portanto, só está interessado nas informações que são enviadas para o sistema (ação do ator) e a posterior resposta do mesmo (resposta do sistema).
- 2) Os nós de decisão são modelados com apenas duas possibilidades de desvio. A modelagem de múltiplas possibilidades de desvio a partir de um único ponto de decisão é realizada encadeando-se dois ou mais nós de decisão para cobrir todas as possibilidades existentes.
- 3) São usadas classes de equivalência para definir os valores a serem usados nos testes funcionais, o que não tem contrapartida no modelo de casos de uso, e;
- 4) São utilizadas expressões booleanas com sintaxe estruturada nas condições de guarda, o que normalmente não é feito no modelo de casos de uso, que usa expressões em linguagem natural.

As diferenças 1 e 2 dessa lista são passíveis de serem resolvidas de forma automática, porém os itens 3 e 4 requerem a intervenção do analista de testes. Para apoiar essa intervenção, são inseridos comentários no modelo de testes gerados a partir das regras associadas às ações no modelo de casos de uso. O objetivo desses comentários é deixar claro para o analista de testes as regras que influenciam o comportamento do sistema naquele ponto. A partir da análise desses comentários espera-se que o analista seja capaz de definir as classes de equivalência de forma a cobrir as possibilidades estabelecidas nas regras.

A Tabela 6-1 define as transformações entre os elementos do metamodelo UCModel e TDE:

Tabela 6-1 – Transformação dos elementos do metamodelo UCModel para o metamodelo TDE

UCModel	TDE
Ação do ator (<i>ActorAction</i>)	Ação do ator (<i>ActorAction</i>)
Ação do sistema (<i>SystemAction</i>)	Eliminado do modelo. A ação anterior à ação do sistema deve ser ligada à ação posterior à ação do sistema
Resposta do sistema (<i>SystemResponse</i>)	Resposta do sistema (<i>SystemResponse</i>)
Nó de decisão	Devem ser gerados n-1 nós de decisão, onde n é o número de desvio que existem a partir do nó de decisão do modelo de casos de uso
Nós de decisão antecidos por uma resposta do sistema	É gerada uma classe de equivalência com o nome do fluxo e as opções “sim” e “não”, indicando que esse caminho pode ou não ser percorrido. A condição de guarda do fluxo é substituída por “fluxo = sim”.
Regras (<i>Rules</i>)	Comentários. Caso as regras estejam associadas a uma ação do sistema, os comentários devem ser associadas às ações anteriores à esta.

Por ser implementado como um *plug-in* do BOUML (PAGÉS, 2011), a ModelT² é capaz de ler o modelo de casos de uso, aplicar as regras de derivação e gerar o modelo de testes diretamente na ferramenta BOUML, sem a necessidade de exportar ou importar esses modelos. Assim, basta o analista de testes selecionar o pacote-raiz estereotipado <<*tc_model*>> e, sem seguida, acionar a opção *Tools->Create Test Models* para executar a ModelT². A Figura 6-13 e a Figura 6-14 mostram, respectivamente, a tela de seleção dos casos de uso para os quais os modelos de teste serão gerados e a tela que confirma quais casos de uso estão em conformidade com o metamodelo UCModel, ou seja, antes da geração do modelo de testes é necessário que o modelo de casos de uso atenda a todas as restrições do metamodelo UCModel.

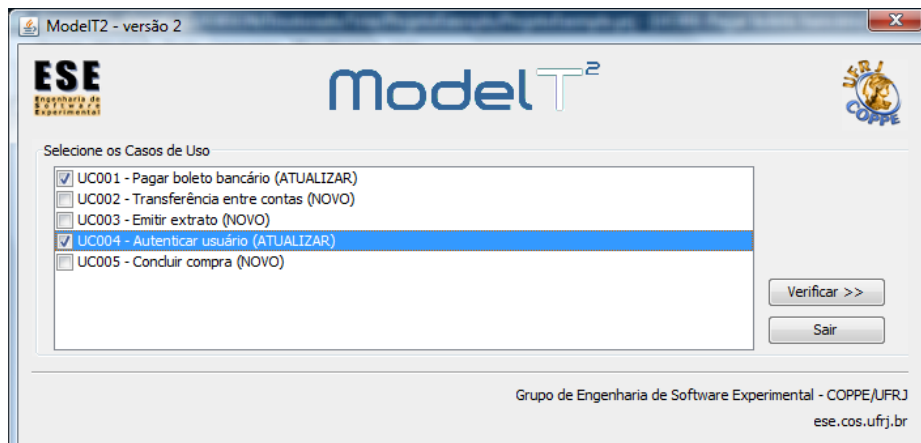


Figura 6-13 – Tela inicial da ModelT² para seleção dos casos de uso

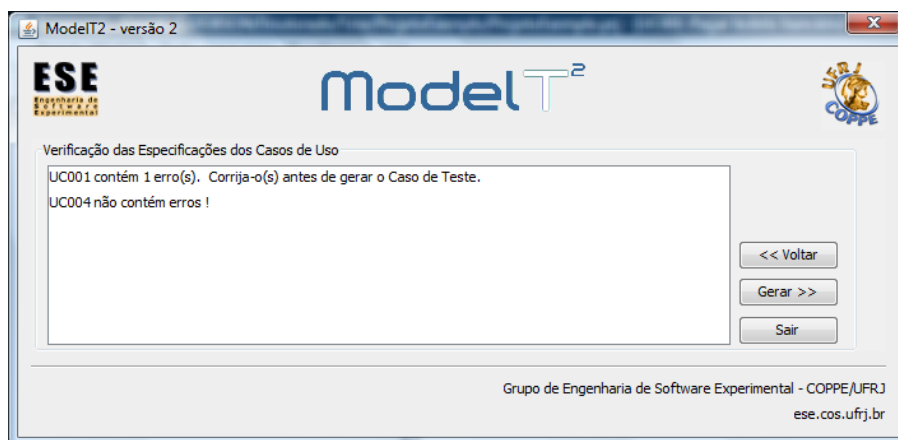


Figura 6-14 – Segunda tela da ModelT² que indica os casos de uso que estão aptos a terem seus modelos de teste gerados

A Figura 6-15 apresenta o modelo de testes referente à esse caso de uso gerado pela ferramenta ModelT2. Repare que:

- 1) As ações do sistema foram eliminadas no modelo de testes enquanto as ações do ator e as respostas do sistema foram mantidas;
- 2) Os comentários associados às ações reportam as regras relevantes nesse ponto de execução do caso de uso;
- 3) Os comentários associados às ações destacam as expressões condicionais que precisam ser refatoradas para refletir os valores das classes de equivalência;
- 4) Para o fluxo alternativo A2, que representa a escolha explícita do ator pela opção “cliente seleciona a opção de leitura eletrônica”, a ModelT2 cria automaticamente uma classe de equivalência “a2” com os valores “sim” e “não” e substitui a expressão “cliente seleciona a opção de leitura eletrônica” pela expressão “a2 = sim”. Dessa forma, o procedimento de testes gerado ao final vai percorrer o fluxo A2 quando o valor de “a2 = sim” e vai seguir pelo fluxo normal quando valor de “a2 = não”.

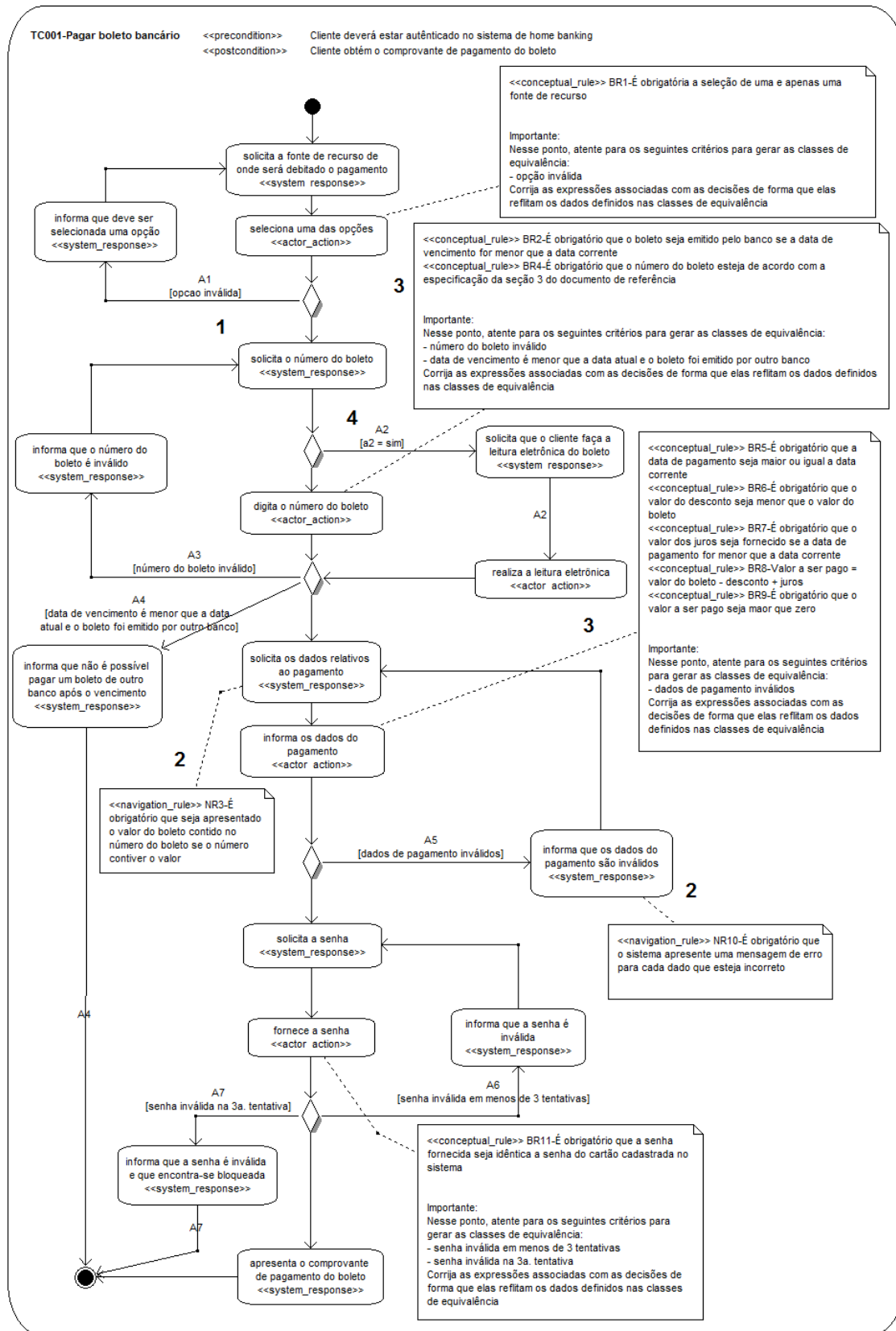


Figura 6-15 - Modelo de testes gerado pela ModelT2 referente ao caso de uso "Pagar boleto bancário"

6.3.1.2 Edição do Modelo de Testes

Do ponto de vista estrutural, nada precisa ser alterado no modelo de testes gerado pela ModelT2, pois as seqüências de ações e desvios definidos nesse modelo cobrem todos os caminhos especificados nos casos de uso. Nessa etapa o analista de testes deve concentrar seus esforços na definição das classes de equivalência e seus valores de forma a cobrir as condições estabelecidas nas regras e nas expressões de guarda associadas aos desvios. Para essa tarefa o analista de testes utiliza a própria ferramenta BOUML e é apoiado pelos comentários criados no modelo, que destacam as regras relevantes em cada ponto de decisão, e por outros artefatos da especificação como o glossário e os modelos conceituais do domínio, nos quais os conceitos relacionados ao domínio do problema devem estar definidos.

A Figura 6-16 apresenta o modelo de testes do caso de uso “Pagar boleto bancário” alterado pelo analista de testes. Neste caso, foram realizadas as seguintes alterações:

- 1) Criação das classes de equivalência opcao1 com os valores “valida” e “invalida”;
- 2) Criação da classe de equivalência boleto1 com os valores “numero_invalido”, “vencido” e “valido”;
- 3) Criação da classe de equivalência dadosPagamento1 com os valores “data_invalida”, “desconto_invalido”, “juros_invalidos”, “valor_final_invalido” e “valido”;
- 4) Criação das classes de equivalência senha1 com os valores “valida” e “invalida”;
- 5) Criação das classes de equivalência erro1 com os valores 2 e 3. Esses valores são usados para simular a senha invalida na 2ª e 3ª tentativas;
- 6) Refatoração das expressões de guarda associadas aos desvios para que estas refletissem as variáveis e valores definidos nas classes de equivalência.

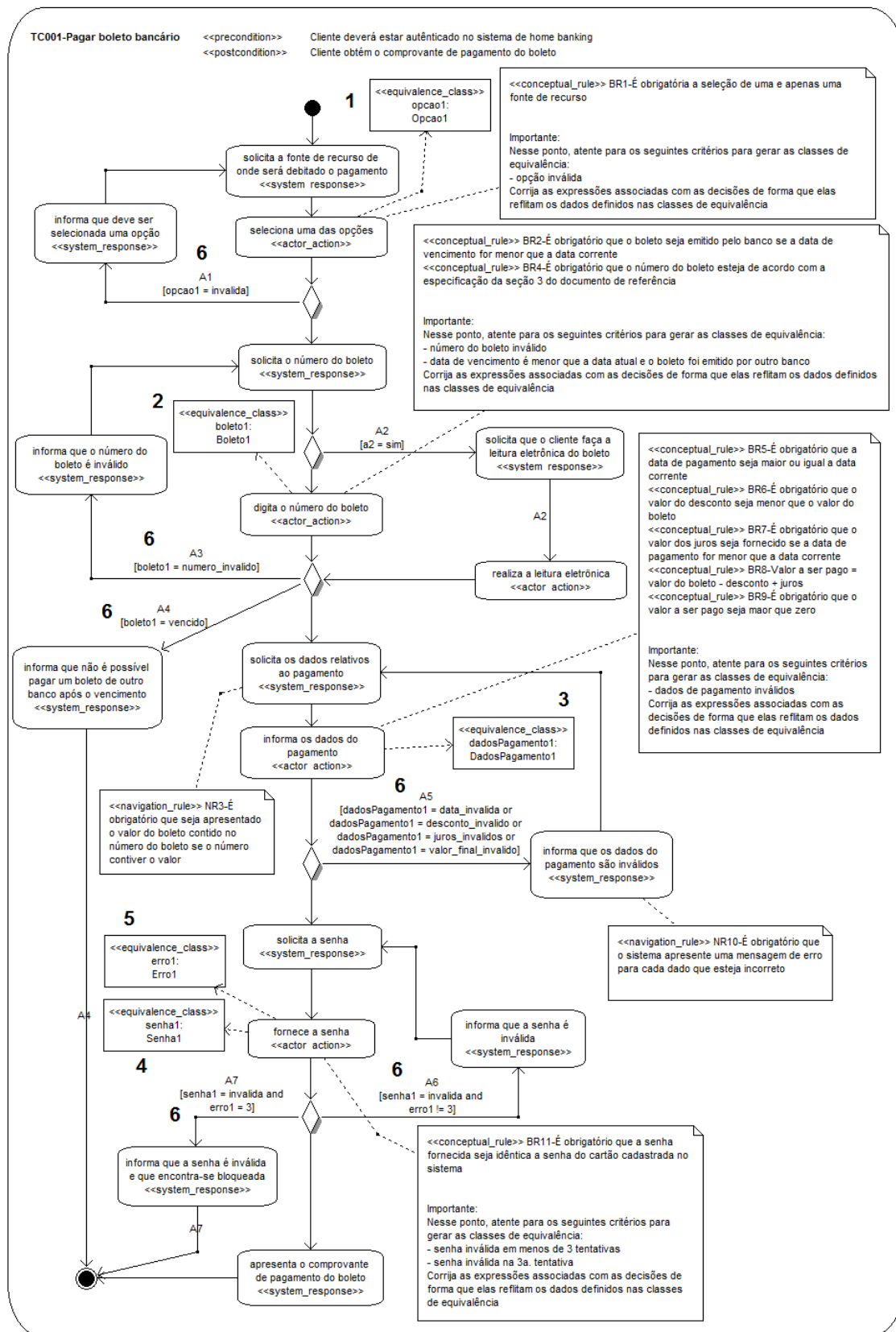


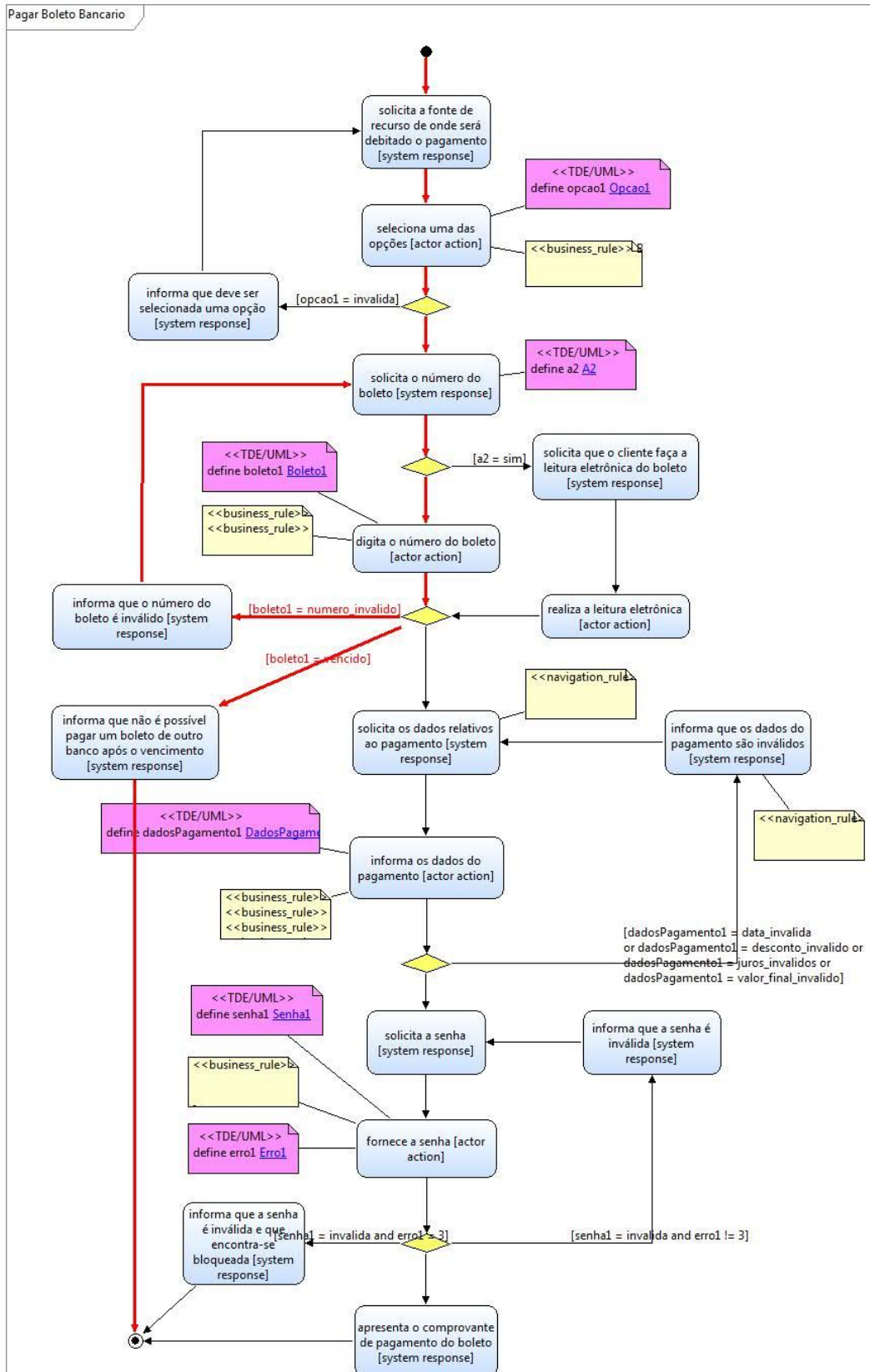
Figura 6-16 - Modelo de testes referente ao caso de uso "Pagar boleto bancário" alterado pelo analista de testes

6.3.1.3 Geração dos Casos e Procedimentos de Teste Funcionais

Após a complementação do modelo de testes é possível utilizá-lo na geração dos casos e procedimentos de teste funcionais. Para essa tarefa é utilizada a ferramenta TDE/UML[®] que interpreta o modelo de testes e as classes de equivalência com os respectivos valores e percorre os caminhos especificados pelo diagrama do modelo de testes aplicando combinações desses valores. O comportamento padrão da TDE/UML[®] é percorrer todos os caminhos possíveis e combinações entre eles, além de todas as combinações de valores possíveis em cada ponto de decisão, mas essa estratégia pode ser alterada na ferramenta. Usando a estratégia padrão o modelo da Figura 6-16 gerou 56 combinações de caminhos possíveis com 80 possibilidades de exploração desses caminhos a partir dos valores definidos para as classes de equivalência. A Figura 6-17 apresenta parte do documento de casos e procedimentos gerados pela TDE/UML[®] ressaltando um dos caminhos gerados e as combinações de valores que permitem percorrer esse caminho de duas formas distintas. O apêndice D apresenta os casos e procedimentos de teste gerados a partir do modelo da Figura 6-16, onde foi selecionada a estratégia de percorrer todas as transições, mas sem realizar todas as combinações entre elas. Como resultado foram gerados 3 caminhos possíveis no diagrama com 6 possibilidades para percorrê-los.

1.1. Test Case 'TC001_TP1'

1.1.1. Test Case Path



1.1.2. Test Case Steps

1	solicita a fonte de recurso de onde será debitado o pagamento [system response]	<no description>
2	seleciona uma das opções [actor action]	<no description>
3	solicita o número do boleto [system response]	<no description>
4	digita o número do boleto [actor action]	<no description>
5	informa que o número do boleto é inválido [system response]	<no description>
6	solicita o número do boleto [system response]	<no description>
7	digita o número do boleto [actor action]	<no description>
8	informa que não é possível pagar um boleto de outro banco após o vencimento [system response]	<no description>

1.1.3. Test Case Data Variations

	opcao1		a2		boleto1		
	valida	invalida	sim	nao	numero_invalido	vencido	valido
TC001_TP1							
Proc. 0							
Step 2							
Step 3							
Step 4							
Step 6							
Step 7							

Figura 6-17 – Trecho do documento de casos e procedimentos de testes do caso de uso “Pagar boleto bancário”, com um caminho e os valores para percorrer esse caminho

6.3.1.4 Trabalhos Relacionados

A utilização da ModelT² para geração do modelo de testes inicial se propõe a:

- Minimizar o esforço na criação desse modelo, no momento em que permite obter todos os caminhos possíveis de exploração do caso de uso de forma automática, e;
- Apoiar a definição das classes de equivalência e seus valores através de comentários no modelo de testes que destacam as regras a serem observadas em determinado ponto de decisão

Nas abordagens para modelagem de testes funcionais TDE (HARTMANN *et al.*, 2005) e TSD (GÓIS *et al.*, 2010), os diagramas que descrevem os modelos de teste são gerados manualmente pelo analista de testes, apoiado pelas respectivas ferramentas (TDE/UML[®] e TestKase). A abordagem aqui proposta procura reusar a especificação dos casos de uso e criar artefatos que permitam explorar abordagens de MBT para testes funcionais, tentando minimizar a interferência manual na derivação dos modelos de teste e destacando os pontos onde ela se faz necessária. Nesse

sentido, o mapeamento realizado entre o metamodelo UCModel e o metamodelo TDE pode servir como ponto de partida para que outras abordagens MBT possam se valer da UCModel para geração de informações relevantes para os testes funcionais.

6.4 Conclusão

O objetivo principal da infra-estrutura computacional organizada no escopo deste trabalho é oferecer apoio à especificação dos casos de uso, descritos como diagramas de atividades baseados na UCModel, e à geração de casos e procedimentos de teste, duas atividades que fazem parte da abordagem proposta. Para tal, foram implementadas duas ferramentas chamadas UseCaseAgent e ModelT². A concepção dessas ferramentas procurou levar em consideração a organização de um ambiente de trabalho no qual o desenvolvedor tivesse acesso às várias funcionalidades oferecidas pelas ferramentas da forma mais transparente possível, minimizando a necessidade de transferência de informações (diagramas) entre as diversas ferramentas que compõem a infra-estrutura proposta. Por isso, ambas as ferramentas foram implementadas como *plug-ins* da ferramenta BOUML (PAGÉS, 2011), o que possibilita a exploração das funcionalidades oferecidas por essas ferramentas dentro do mesmo ambiente de trabalho.

A geração de casos e procedimentos de testes e a ferramenta ModelT² não foram alvos de uma avaliação experimental no escopo desse trabalho. A decisão de não avaliação está baseada em um dos objetivos da própria abordagem aqui proposta e materializado através da ferramenta ModelT²: apoiar a geração de casos e procedimentos de testes funcionais usando uma abordagem MBT, através da geração automática de um modelo de testes preliminar, em substituição à geração manual desse modelo. Ou seja, como a abordagem preconizada por TDE e a própria ferramenta TDE/UML[®] são usados em um ambiente industrial de larga escala, a preocupação era mostrar a viabilidade da derivação automática de uma versão preliminar do modelo de testes, a fim de evitar que este seja gerado manualmente pelo analista de testes. Entretanto, existem avaliações que devem ser conduzidas nesse contexto e que serão objetos de trabalhos futuros, como, por exemplo, avaliar se o apoio criado no modelo de testes preliminar para auxiliar o analista de testes na complementação do modelo é adequado.

Por outro lado, a ferramenta UseCaseAgent foi utilizada na condução da avaliação experimental da abordagem, que será apresentada no capítulo 7.

7 Avaliação Experimental

Neste capítulo são apresentados dois estudos conduzidos no contexto da abordagem proposta nesta tese. O resultado desses estudos nos trazem indicações de que o uso da abordagem para especificação de casos de uso baseada no metamodelo UCMoel, apoiada pela ferramenta UseCaseAgent, pode auxiliar na redução de defeitos nesses artefatos.

7.1 Introdução

O objetivo principal dos dois estudos apresentados neste capítulo é de avaliar, de forma conjunta, a abordagem proposta nesta tese, da qual o metamodelo UCMoel faz parte, e as ferramentas que apóiam a atividade de especificação.

A Figura 7-1 ilustra as principais fases principais da especificação de requisitos (SAWYER e KOTONYA, 2004), onde é possível observar a área de atuação da ferramenta UseCaseAgent e da técnica de inspeção ActCheck (MELLO, 2010).

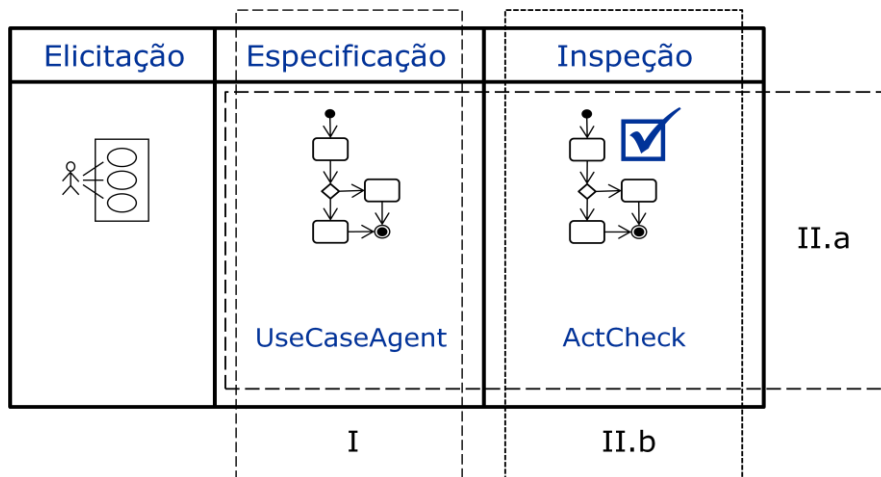


Figura 7-1 – Visão geral dos estudos para avaliação da abordagem de especificação de requisitos

Nesse contexto, algumas questões com relação à avaliação da abordagem proposta:

- 1) É viável construir diagramas de atividades com o apoio da ferramenta UseCaseAgent?

- 2) O quão eficiente (tempo) é a construção de diagramas de atividades com a ferramenta UseCaseAgent em comparação com o uso de um editor de diagramas de atividades comum?
- 3) Os diagramas de atividades construídos com a ferramenta UseCaseAgent apresentam maior qualidade (menor número de defeitos) em comparação com os diagramas produzidos com um editor de diagramas de atividades comum?
- 4) A eficiência (número de defeitos detectados x número de defeitos total) e a eficácia (número de defeitos detectados por unidade de tempo) da técnica de inspeção ActCheck é maior que a eficiência e eficácia das inspeções *ad-hoc*?

A primeira e segunda questões estão relacionadas à forma com que os diagramas de atividades são construídos e, portanto, estão relacionadas unicamente com a fase de especificação. Da mesma forma, a quarta questão está fortemente relacionada à fase de verificação, mas depende dos modelos produzidos na fase de especificação. Por outro lado, a terceira questão é ortogonal às fases de especificação e verificação, podendo ser indiretamente avaliada através dos resultados da inspeção conduzida nos diagramas de atividades produzidos na fase de especificação.

Dessa forma, com o objetivo de avaliar a abordagem proposta, dois estudos foram planejados e conduzidos no escopo desta tese (Figura 7-1), nos meses de outubro e novembro de 2010:

- I. O primeiro estudo trata as questões 1 e 2 e cria os diagramas de atividades necessários ao segundo estudo, e;
- II. O segundo estudo trata as questões 3 e 4 sob dois pontos de vista: II.a) observando o número de defeitos nas especificações funcionais, independente da técnica usada para encontrá-los, e; II.b) observando questões relacionadas à eficiência e eficácia da técnica ActCheck, independente de qual ferramenta foi usada na elaboração dos diagramas de atividades inspecionados.

É importante ressaltar que este trabalho descreve somente os estudos I e II.a, que fazem parte do contexto de pesquisa desta tese. Todos os detalhes sobre o estudo II.b podem ser encontrados em MELLO (2011).

Com relação à construção dos diagramas de atividades, duas possibilidades são contempladas através de artefatos e ferramentas definidos e construídos no escopo desta tese. Essas possibilidades foram exploradas no estudo I:

- Especificação usando o perfil UML ProfileUCModel: neste caso o desenvolvedor conta, apenas, com a ferramenta BOUML e o perfil ProfileUCModel para construção dos diagramas de atividades relativos aos casos de uso. No contexto deste estudo, essa abordagem de especificação foi considerada *ad-hoc*, devido a essa ser a forma usual de trabalho dos participantes do estudo, ou seja, construção de modelos com o uso de ferramentas case. Além disso, a aplicação dos estereótipos do perfil ao diagrama não garante que este está em conformidade com o metamodelo UCMModel, ficando a cargo do desenvolvedor a tarefa de verificação. Nesse caso, o desenvolvedor pode ou não estar atento às questões de conformidade.
- Especificação usando UseCaseAgent: neste caso o desenvolvedor conta com uma ferramenta especialmente projetada para tratar os elementos do metamodelo UCMModel e suas restrições. Essa ferramenta permite que o desenvolvedor especifique o caso de uso em formato puramente textual, a partir do qual o diagrama de atividades correspondente é automaticamente gerado na ferramenta BOUML.

Assim, a Figura 7-2 apresenta uma visão geral dos estudos de avaliação I e II.a planejados e executados no escopo desta tese.

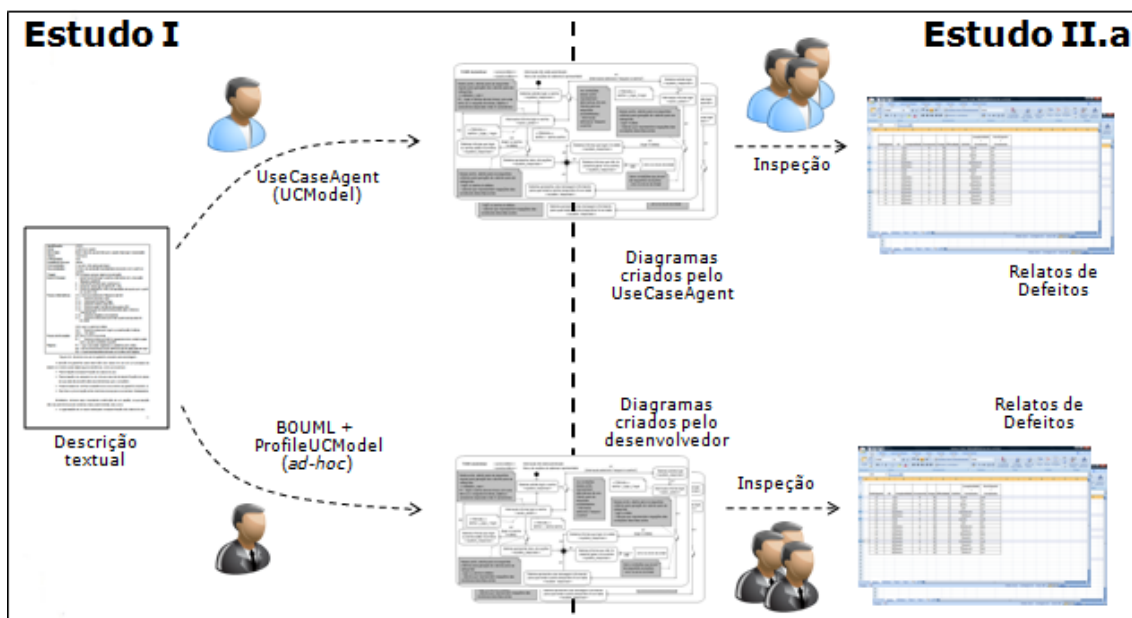


Figura 7-2 – Visão geral dos estudos de avaliação

As seções 7.2 e 7.3 relatam os estudos I e II.a respectivamente, detalhando os objetivos, contexto, planejamento execução e análise dos dados. A seção 7.4 discute

as ameaças à validade levando em consideração a combinação dos dois estudos. Finalmente a seção 7.5 apresenta a conclusão com um breve resumo dos resultados obtidos.

7.2 Primeiro Estudo

7.2.1 Objetivo

O objetivo do primeiro estudo pode ser formalizado, usando o paradigma GQM (BASILI e ROMBACH, 1998), como:

Analisar	a ferramenta UseCaseAgent (doravante denominada UCA) e um editor comum de diagrama de atividades (doravante denominado EDA)
com o propósito de	caracterizar
com respeito ao	tempo gasto na construção de diagramas de atividades
do ponto de vista	dos engenheiros de software
no contexto da	construção de casos de uso do módulo financeiro de um sistema de informação Web de larga escala por desenvolvedores pertencentes à equipe do projeto.

7.2.2 Seleção do Contexto

A seleção dos casos de uso foi realizada por conveniência, a partir do documento de requisitos funcionais do módulo financeiro do SiGIC. Foram selecionados dez casos de uso. Esses casos de uso já haviam sido especificados anteriormente usando um gabarito de casos de uso próprio do projeto e um editor de textos. Não foi adotado, naquele momento, nenhum modelo formal de casos de uso que apoiasse a especificação nem tampouco uma forma de verificação semi-automática desses documentos. Todos os dez casos de uso selecionados já haviam sido inspecionados e corrigidos, de forma *ad-hoc*, por membros da equipe do projeto que não participaram desse estudo.

Existem algumas propostas na literatura técnica para medição de casos de uso (DIEV, 2006; LAVAZZA *et al.*, 2008; KANJILAL *et al.*, 2009; LAVAZZA e ROBILOLO, 2010). Entretanto, até o presente momento, não há consenso na comunidade científica sobre um método que permita obter uma medida de tamanho ou complexidade para este artefato. Dessa forma, usando como ponto de partida o contexto do desenvolvimento e os conceitos explorados na descrição dos casos de uso selecionados para o estudo, foi definida uma métrica, com seus valores de medida apresentados na escala razão, para determinar a complexidade desses casos de uso e permitir a comparação entre eles. Essa métrica pode ser descrita pela fórmula:

$$C = \text{NAFP} + \text{NFA} + \text{NR}$$

onde :

NAFP = número de ações no fluxo principal

NFA = número de fluxos alternativos

NR = número de regras

A comparação dos casos de uso usando a complexidade calculada a partir dessa fórmula demonstra-se coerente em virtude do contexto e das características do ambiente onde esses casos de uso foram especificados:

- Todos os casos de uso descreviam funcionalidades pertencentes ao mesmo domínio de problema;
- Todos os casos de uso foram descritos pela mesma equipe de requisitos, ou seja, o estilo de redação dos casos de uso era bastante semelhante;
- Todos os casos de uso foram elicitados com o mesmo *stakeholder*, e;
- Todos os casos de uso adotaram o mesmo gabarito na sua descrição.

A Tabela 5-1 apresenta o cálculo de complexidade para os dez casos de uso selecionados. A partir desses números foram calculadas a média (74,63) e o desvio-padrão (28,91) da amostra e, a partir desses valores, foram criados 4 faixas com os limites: média – (2 x desvio-padrão), média – desvio-padrão, média + desvio-padrão e média + (2 x desvio-padrão). Cada uma dessas faixas foi associada aos valores ordinais Baixo, Médio-Baixo, Médio-Alto e Alto. A faixa em que o caso de uso é posicionado determina o seu nível de complexidade (Figura 5-1). O nível de complexidade foi usado, posteriormente, para apoiar a distribuição dos casos de uso entre os participantes de forma mais equilibrada.

Tabela 7-1 - Cálculo da complexidade dos casos de uso

Caso de uso	Nº de ações do fluxo principal	Número de fluxos alternativos	Número de regras	Complexidade
26	20	1	5	26
10	28	2	1	31
6	30	3	2	35
27	42	4	4	50
16	43	5	6	54
40	51	9	7	67
36	72	3	7	82
35	72	3	7	82
17	98	5	5	108
15	92	8	19	119

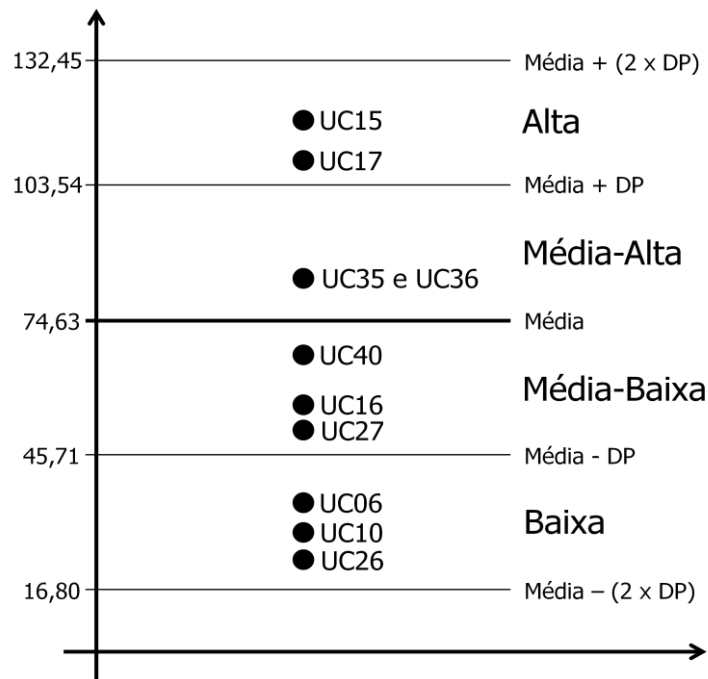


Figura 7-3 - Avaliação do nível de complexidade dos casos de uso selecionados

7.2.3 Seleção de Variáveis

Nesse estudo a variável independente é representada pela ferramenta usada para especificar os casos de uso (UCA, EDA) (escala nominal), enquanto a variável dependente é o tempo gasto na especificação de cada caso de uso (escala razão). Assim, esse estudo pode ser caracterizado como um fator (ferramenta) e dois tratamentos (UCA e EDA).

7.2.4 Participantes

Dois desenvolvedores da equipe de projeto foram escolhidos por conveniência. Esses desenvolvedores tinham conhecimento do domínio do problema e estavam diretamente envolvidos com a especificação funcional e as atividades de teste do módulo financeiro. O primeiro desenvolvedor tinha larga experiência na especificação de casos de uso e conhecimento detalhado do módulo financeiro, enquanto o segundo, por ter sido integrado ao projeto em período mais recente, tinha pouca experiência nesses tópicos.

7.2.5 Projeto do Estudo

Os dez casos de uso selecionados foram separados em dois conjuntos com cinco casos de uso cada. Esses conjuntos foram criados baseando-se na

complexidade dos casos de uso (seção 7.2.2) de forma a equilibrar a complexidade dos dois conjuntos. A Tabela 7-2 apresenta os conjuntos A e B com os respectivos casos de uso e a complexidade de cada um.

Tabela 7-2 - Conjuntos de casos de uso balanceados pelo nível de complexidade

Conjunto de casos de uso	Caso de uso	Complexidade do caso de uso	Complexidade média do conjunto
A	26	Baixa	79,5 (Média-Alta)
	10	Baixa	
	27	Médio-Baixo	
	40	Médio-Baixo	
	35	Média-Alta	
	15	Alta	
B	26	Baixa	69,75 (Média-Baixa)
	10	Baixa	
	06	Baixa	
	16	Médio-Baixo	
	36	Média-Alta	
	17	Alta	

Os casos de uso 10 e 26 fizeram parte dos conjuntos A e B porque foram usados como “exercício de aquecimento” a fim de minimizar os efeitos do aprendizado da ferramenta de especificação, embora os desenvolvedores não soubessem desse detalhe.

Após a definição dos conjuntos A e B, estes foram associados a cada um dos participantes P1 e P2, alternadamente, de forma que cada participante especificasse casos de uso de ambos os conjuntos. Esse arranjo entre os dois conjuntos e os dois participantes demandou que a execução fosse realizada em duas rodadas, conforme apresentado na Tabela 7-3.

Tabela 7-3 - Conjuntos de casos de uso atribuídos a cada participante por rodada

Rodada	Ferramenta	Participante	Nível de experiência	Grupo do caso de uso
1	EDA	P1	Alto	A
		P2	Baixo	B
2	UCA	P1	Alto	B
		P2	Baixo	A

A rodada 1 foi executada de forma *ad-hoc*, ou seja, os participantes construíram as especificações dos casos de uso usando somente um editor comum de diagramas de atividades com o perfil ProfileUCModel. Na segunda rodada, as especificações dos casos de uso foram construídas usando a ferramenta UseCaseAgent, que gera automaticamente os diagramas de atividades correspondentes a cada caso de uso a partir das especificações textuais de cada um.

Com relação ao editor de diagramas de atividades, a ferramenta BOUML (PAGÉS, 2011) foi escolhida. Adicionalmente, cada participante executou a especificação individualmente e foi solicitado que fosse seguida a ordem definida na Tabela 7-1.

7.2.6 Instrumentação

Os instrumentos preparados e usados como apoio ao estudo foram:

1. Especificação funcional original do módulo financeiro contendo a descrição dos casos de uso;
2. Projeto na ferramenta case BOUML vazio, ou seja, sem nenhum modelo, contendo somente o perfil ProfileUCModel com os estereótipos necessários para criação dos diagramas de atividades;
3. Planilha eletrônica para registrar o tempo gasto (em minutos) na especificação de cada caso de uso;
4. Material de treinamento sobre casos de uso;
5. Material de treinamento sobre diagramas de atividades da UML 2;
6. Material de treinamento sobre o metamodelo UCModel;
7. Material de treinamento sobre a ferramenta UseCaseAgent, e;
8. Material de treinamento sobre a ferramenta BOUML, mais especificamente sobre o editor de diagramas de atividades.

O segundo instrumento dessa lista foi criado para evitar que os desenvolvedores precisassem configurar a ferramenta BOUML para uso do perfil ProfileUCModel.

7.2.7 Preparação

O pacote de treinamento dos participantes foi dividido em cinco módulos:

1. Treinamento sobre conceitos gerais relacionados a casos de uso, o gabarito organizado para descrição dos casos de uso e seus elementos;
2. Treinamento sobre o diagrama de atividades da UML 2 e seus elementos;
3. Treinamento sobre o metamodelo UCModel, suas definições e restrições;
4. Treinamento sobre a ferramenta BOUML, mais especificamente sobre o editor de diagramas de atividades, e;
5. Treinamento sobre a criação e edição de casos de uso com a ferramenta UseCaseAgent.

Antes da primeira rodada, os treinamentos de 1 a 4 foram ministrados para ambos os participantes na mesma sessão de treinamento. O treinamento 5 foi ministrado somente após a rodada 1 para evitar que o conhecimento sobre a ferramenta UseCaseAgent influenciasse no resultado final do estudo.

7.2.8 Execução

Baseados no documento original de requisitos do módulo financeiro do SiGIC, cada desenvolvedor especificou os casos de uso atribuídos para ele. Vale a pena lembrar que os casos de uso já haviam sido especificados dentro do processo normal de desenvolvimento do módulo financeiro e a tarefa dos participantes consistia em criar o diagrama de atividades para cada caso de uso de acordo com os conceitos preconizados pelo metamodelo UCMoel. Na primeira rodada a criação dos diagramas foi feita por EDA somente com o apoio do perfil ProfileUCMoel. Cabe ressaltar que a existência do perfil e a aplicação dos estereótipos não garantem que o diagrama de atividades criado esteja aderente ao metamodelo UCMoel, ou seja, não há como garantir que o desenvolvedor irá criar, sem nenhum apoio computacional extra, um diagrama totalmente aderente ao UCMoel. Assim, se o caso de uso no documento original contiver alguma não conformidade com o UCMoel, existe a possibilidade de que essa não conformidade seja traduzida para o diagrama de atividades. Na segunda rodada, a especificação dos casos de uso foi feita por UCA e os diagramas de atividades correspondentes a cada caso de uso foram gerados automaticamente pela ferramenta. Nesse caso, como a UCA só aceita especificações de caso de uso totalmente aderentes ao UCMoel, se o caso de uso no documento original contiver alguma não conformidade com o UCMoel, o desenvolvedor terá de corrigi-la para, posteriormente, gerar o diagrama de atividades.

Os desenvolvedores tiveram uma semana para executar cada rodada, e, ao final de cada uma, foi enviada uma planilha eletrônica contendo o tempo gasto (em minutos) para cada caso de uso especificado.

7.2.9 Análise Quantitativa

A Tabela 7-4 apresenta os tempos reportados pelos participantes para cada caso de uso. Somente oito dos dez casos de uso foram considerados nessa análise, pois os casos de uso 10 e 26 foram usados como “exercício de aquecimento” no uso da ferramenta. Nessa mesma tabela também é possível observar o tempo total e a mediana de tempo por grupo de caso de uso e nível de experiência dos participantes, além do tempo total e mediana de tempo por ferramenta.

Observando o tempo e a mediana por ferramenta, os dados coletados sugerem que o tempo gasto com o uso da ferramenta UCA pode ser menor que o tempo gasto com a ferramenta EDA. Também é possível observar que o desenvolvedor mais experiente teve uma redução significativa no tempo de especificação usando a ferramenta UCA, enquanto o desenvolvedor menos experiente não apresentou redução no tempo, independente do uso da ferramenta ADE ou UCA. Esse último comportamento sugere que pode existir alguma relação entre a experiência do desenvolvedor e o tempo gasto com uso da ferramenta UCA. Por outro lado, analisando os tempos gastos com a especificação dos casos de uso do conjunto A, o tempo gasto e a mediana de tempo com o uso da ferramenta EDA foi menor que o tempo gasto e a mediana de tempo com a ferramenta UCA, o que pode indicar que o nível de experiência do desenvolvedor influencia no desempenho dessa tarefa com a ferramenta UCA.

Entretanto, nenhuma dessas tendências pôde ser avaliada estatisticamente devido ao pequeno tamanho da amostra disponível. Estudos futuros com maior quantidade de participantes podem tirar vantagem do arranjo desse estudo e verificar a validade estatística dessas suposições.

Tabela 7-4 - Tempos reportados pelos participantes para especificação dos casos de uso

Ferramenta	Participante	Nível de experiência	Conjunto do Caso de uso	Caso de uso	Nível de complexidade	Tempo (em minutos)		Mediana (em minutos)		
EDA	P1	Alta	A	27	Média-Baixa	50	180	356	47,5	50
				40	Média-Baixa	45				
				35	Média-Alta	35				
				15	Alta	50				
	P2	Baixa	B	6	Baixa	24	176	176	50,5	50
				16	Média-Baixa	51				
				36	Média-Alta	50				
				17	Alta	51				
UCA	P1	Alta	B	6	Baixa	20	86	271	20,5	23
				16	Média-Baixa	25				
				36	Média-Alta	20				
				17	Alta	21				
	P2	Baixa	A	27	Média-Baixa	20	185	185	51,5	23
				40	Média-Baixa	43				
				35	Média-Alta	62				
				15	Alta	60				

7.2.10 Análise Qualitativa

Após a execução do estudo foi realizada uma reunião com cada participante individualmente a fim de coletar dados qualitativos acerca da execução das atividades. O objetivo dessa reunião foi coletar a percepção de viabilidade de uso da ferramenta UCA e as dificuldades e facilidades observadas no processo de especificação dos casos de uso. Para tal, seis questões foram formuladas:

1. Na sua opinião, os conceitos preconizados pelo metamodelo UCModel trouxeram algum benefício se comparado com o gabarito original dos casos de uso?
2. Quais foram as principais dificuldades na especificação de casos de uso com EDA?
3. Quais foram as principais dificuldades na especificação de casos de uso com UCA?
4. Se você tivesse que usar novamente essa abordagem hoje, você adotaria a ferramenta UCA?
5. O treinamento para uso da EDA foi adequado e suficiente?
6. O treinamento para uso da UCA foi adequado e suficiente?

Ambos os desenvolvedores consideraram os conceitos do UCModel adequados para serem usados no projeto. O participante mais experiente relatou que a padronização é importante, mas expressou preocupação quanto à possibilidade de perda de liberdade na descrição dos casos de uso, embora ele não tenha percebido essa situação nos casos de uso especificados. Com relação à especificação usando EDA, ambos os participantes relataram dificuldades ao lidar com:

- As limitações da tela, especialmente para diagramas maiores;
- Os estereótipos e *tagged-values* do perfil UCModel;
- A criação e edição das ações (passos) do caso de uso, e;
- A grande quantidade de nós de decisão (que representam fluxos alternativos).

Nenhum problema foi relatado com relação ao uso da UCA. Os participantes relataram que trabalhar com a UCA foi "muito mais fácil", pois ela é semelhante a um editor de textos comum, além de permitir a correção de casos de uso durante a edição. Assim, eles preferem trabalhar com a abordagem defendida por UCA. Finalmente, ambos os participantes relataram que a carga de treinamento e o material disponibilizado foram suficientes para especificar os casos de uso solicitados.

Com relação ao tempo reportado, foi perguntado ao participante P2 se ele teve alguma dificuldade ao usar a UCA, já que o tempo de especificação com a UCA foi similar ao tempo de especificação com a EDA. Ele respondeu que não se lembrava de nenhuma dificuldade em particular e que preferia usar a UCA ao invés da EDA. Assim, nenhuma explicação plausível sobre a diferença de tempo foi dada. Uma hipótese para esse comportamento é que a correção dos defeitos apontados por UCA levou algum tempo no processo de especificação, pois a ferramenta não permite geração do diagrama de atividades caso as especificações não estejam em total conformidade com o metamodelo UCMModel. Os dados coletados no segundo estudo foram usados para avaliar esta possibilidade.

7.3 Segundo Estudo

7.3.1 Objetivo

O objetivo do segundo estudo pode ser formalizado, usando o paradigma GQM (BASILI e ROMBACH, 1998), como:

<p>Analisar com o propósito de com respeito ao do ponto de vista dos no contexto de</p>	<p>especificações de casos de uso construídos com a ferramenta UseCaseAgent (doravante denominada UCA) e com um editor comum de diagramas de atividades (doravante denominado EDA) caracterizar número de defeitos inspetores de requisitos inspeção <i>ad-hoc</i> de diagramas de atividades descrevendo casos de uso do módulo financeiro de um sistema de informação Web de larga escala por estudantes de pós-graduação de Engenharia de Software.</p>
--	--

7.3.2 Seleção do Contexto

Do total de oito casos de uso especificados e analisados no primeiro estudo, quatro foram selecionados para o segundo estudo: 6, 15, 36 e 40. Só foi possível selecionar quatro casos de uso devido ao número de participantes disponíveis para este segundo estudo (número de alunos inscritos na disciplina onde foi conduzido o estudo). Os quatro casos de uso foram escolhidos por conveniência, tentando selecionar aqueles com diferentes níveis de complexidade e especificados anteriormente pelos desenvolvedores tanto com a ferramenta UCA quanto com a ferramenta EDA.

7.3.3 Seleção de Variáveis

Nesse estudo a variável independente é a especificação dos casos de uso (escala nominal), enquanto a variável dependente é o número de defeitos detectados na especificação de casos de uso (escala razão). Assim, esse estudo também pode ser caracterizado como um fator (especificação de caso de uso) e dois tratamentos (especificações de casos de uso criadas com a UCA e com a EDA).

7.3.4 Participantes

Os participantes foram selecionados por conveniência em uma disciplina de Verificação, Validação e Teste (VV&T) oferecida em um curso de pós-graduação na COPPE/UFRJ. Oito participantes assinaram o termo de consentimento e preencheram um formulário de caracterização usado para avaliar sua experiência na indústria de software e conhecimentos com relação à especificação de requisitos, inspeções de software e UML, dentre outros. Tomando por base as informações do formulário de caracterização, os participantes foram classificados com nível de experiência Baixa, Média-Baixa, Média, Média-Alta ou Alta. O processo de classificação dos participantes foi realizado por dois pesquisadores que analisaram os questionários de caracterização e chegaram a um consenso sobre o nível de experiência de cada participante. Essa classificação foi posteriormente discutida com um terceiro pesquisador sênior para obtenção da classificação final.

7.3.5 Projeto do Estudo

A Tabela 7-5 apresenta a distribuição dos quatro casos de uso selecionados para o segundo estudo entre seus participantes. Essa distribuição procurou equilibrar o nível de complexidade dos casos de uso atribuídos a cada um dos participantes e, para evitar efeitos colaterais relacionados à ferramenta usada para especificar o caso de uso, atribuiu a cada um deles diagramas de atividades criados tanto com a ferramenta UCA quanto com a ferramenta EDA. Vale a pena ressaltar que os participantes não foram informados se os casos de uso haviam sido especificados com uma ou outra ferramenta. Além disso, a fim de mitigar efeitos colaterais relacionados ao nível de experiência do inspetor, o mesmo caso de uso foi inspecionado por participantes com diferentes habilidades. Com esse arranjo, cada caso de uso foi inspecionado oito vezes por diferentes participantes com diferentes níveis de experiência.

Tabela 7-5 - Distribuição dos casos de uso pelos participantes do segundo estudo

Participante	Nível de experiência	Casos de uso especificados com	
		EDA	UCA
P1	Alto	15, 06	40, 36
P2	Alto	06, 15	36, 40
P3	Médio-Alto	36, 06	40, 15
P4	Médio-Alto	06, 36	15, 40
P5	Médio-Baixo	36, 40	06, 15
P6	Médio-Baixo	40, 36	15, 06
P7	Baixo	15, 40	06, 36
P8	Baixo	40, 15	36, 06

7.3.6 Instrumentação

Os instrumentos preparados e usados para apoiar o estudo foram:

1. Formulário de consentimento;
2. Formulário de caracterização;
3. Planilha eletrônica para relato das discrepâncias e do tempo de duração da inspeção;
4. Material de treinamento sobre categorização de defeitos;
5. Material de treinamento sobre diagramas de atividades da UML 2;
6. Material de treinamento sobre o metamodelo UCMoel, e;
7. Material de treinamento sobre a ferramenta BOUML, mais especificamente sobre o editor de diagramas de atividades.

Os materiais 5, 6 e 7 foram os mesmos utilizados no primeiro estudo (seção 7.2).

7.3.7 Preparação

Ao longo do curso de VV&T todos os participantes tiveram acesso à informações teóricas sobre inspeção de software e a oportunidade de praticar inspeções utilizando diferentes técnicas (*ad-hoc*, técnicas baseadas em *checklist*, técnicas baseadas em heurísticas e técnicas de leitura baseadas em perspectivas).

Os oito participantes foram treinados nos mesmos tópicos, a saber:

1. Treinamento sobre a categorização de defeitos a ser utilizada no contexto do estudo;
2. Treinamento sobre o diagrama de atividades da UML 2 e seus elementos;
3. Treinamento sobre o metamodelo UCMoel, suas definições e restrições, e;
4. Treinamento sobre a ferramenta BOUML, mais especificamente sobre o editor de diagramas de atividades.

Devido à ausência de alguns participantes, o treinamento foi ministrado em dois momentos (o primeiro com cinco participantes e o segundo com os três restantes), porém usando o mesmo material, com o mesmo tempo de duração e com os mesmos instrutores. Acreditamos que o treinamento dos participantes em duas sessões separadas não causou impacto no resultado final, pois as inspeções ocorreram de forma individual e foi solicitado aos participantes que não trocassem informações entre eles sobre as inspeções que estavam sendo realizadas.

7.3.8 Execução

Os participantes tiveram duas semanas para executar a inspeção *ad-hoc* e, ao final, enviaram a planilha eletrônica com as discrepâncias encontradas e os tempos de inspeção (em minutos) por caso de uso.

7.3.9 Validação dos Dados

Todos os oito participantes enviaram seus formulários e nenhum relatório de discrepâncias foi descartado. Como não havia um oráculo que informasse os defeitos reais existentes nos artefatos, todas as trinta e duas planilhas foram verificadas individualmente por dois pesquisadores e cada discrepância foi classificada como um defeito ou falso positivo individualmente. Após essa etapa, uma reunião de consenso entre esses mesmos pesquisadores foi realizada para decidir se determinada discrepância era ou não um defeito. Nos casos em que a classificação inicial dos pesquisadores era conflitante, eles chegavam a um consenso argumentando contra ou a favor de determinada interpretação acerca da discrepância relatada.

7.3.10 Resultados

A análise estatística apresentada nessa seção foi executada com a ferramenta JMP v.4 e $\alpha = 0.05$. A Tabela 5-6 resume o resultado geral das inspeções apresentando o número de defeitos reportados e a mediana dos defeitos por tipo de especificação (EDA ou UCA). Pode-se observar que houve um decréscimo de 22,8% no número de defeitos reportados nas especificações produzidas com a ferramenta UCA em relação aquelas produzidas com a ferramenta EDA, o que sugere que os casos de uso especificados com UCA apresentam menos defeitos que os mesmos casos de uso especificados com EDA.

Entretanto, o caso de uso 15, de alta complexidade, apresentou um comportamento diferente dos demais casos de uso, pois foi o único no qual o número

de defeitos na especificação com UCA aumentou em comparação com EDA (destacado em cinza na Tabela 7-6). A análise dos defeitos apontados pelos inspetores revelou que um único inspetor dos oito que realizaram a inspeção relatou 7 defeitos relacionados ao nível de abstração das descrições das ações desse caso de uso. Nenhum dos outros sete inspetores considerou isso um defeito, o que causou o desbalanceamento somente neste caso de uso.

Tabela 7-6 - Número de defeitos reportados por caso de uso e ferramenta

Ferramenta	Caso de Uso	Nível de complexidade	Número de defeitos reportados	Mediana	
EDA	6	Baixa	11	92	10,5
	15	Alta	18		
	36	Média-Baixa	24		
	40	Média-Alta	39		
UCA	6	Baixa	7	71	7,0
	15	Alta	22		
	36	Média-Baixa	10		
	40	Média-Alta	32		

Os *boxplots* com a distribuição de defeitos por tipo de especificação (Figura 7-4)¹⁴ mostram que o número de defeitos reportados para casos de uso especificados com UCA é menor que o número de defeitos reportados para casos de uso especificados com EDA. A análise dos *boxplots* nos permite observar que:

- Há indícios de que o número de defeitos dos casos de uso especificados com UCA seja menor que o número de defeitos dos casos de uso especificados com ADE, pois a mediana de defeitos para distribuição UCA é menor que a mediana para a distribuição EDA;
- Há indícios de que casos de uso especificados com UCA são mais estáveis em termos de número de defeitos, não sofrendo tanto a influência de fatores externos quanto os casos de uso especificados com EDA. Esse indício é sustentado pelo intervalo quartílico mais concentrado do *boxplot* UCA em relação ao *boxplot* EDA, que se apresenta mais disperso, e;
- Há indícios de que a diferença entre o número de defeitos apresentados por UCA e EDA seja estatisticamente significativa, pois todo o intervalo quartílico do *boxplot* da distribuição de UCA, ou seja, pelo menos 75% da amostra, está abaixo da mediana de defeitos da distribuição de EDA.

¹⁴ Na Figura 7-4 a ferramenta EDA está representada pelo rótulo ADE (Activity Diagram Editor)

O teste estatístico t de *Student* confirmou que o número de defeitos reportados para os casos de uso especificados com UCA é significativamente menor que o número de defeitos reportados para os casos de uso especificados com EDA ($\alpha = 0.05$).

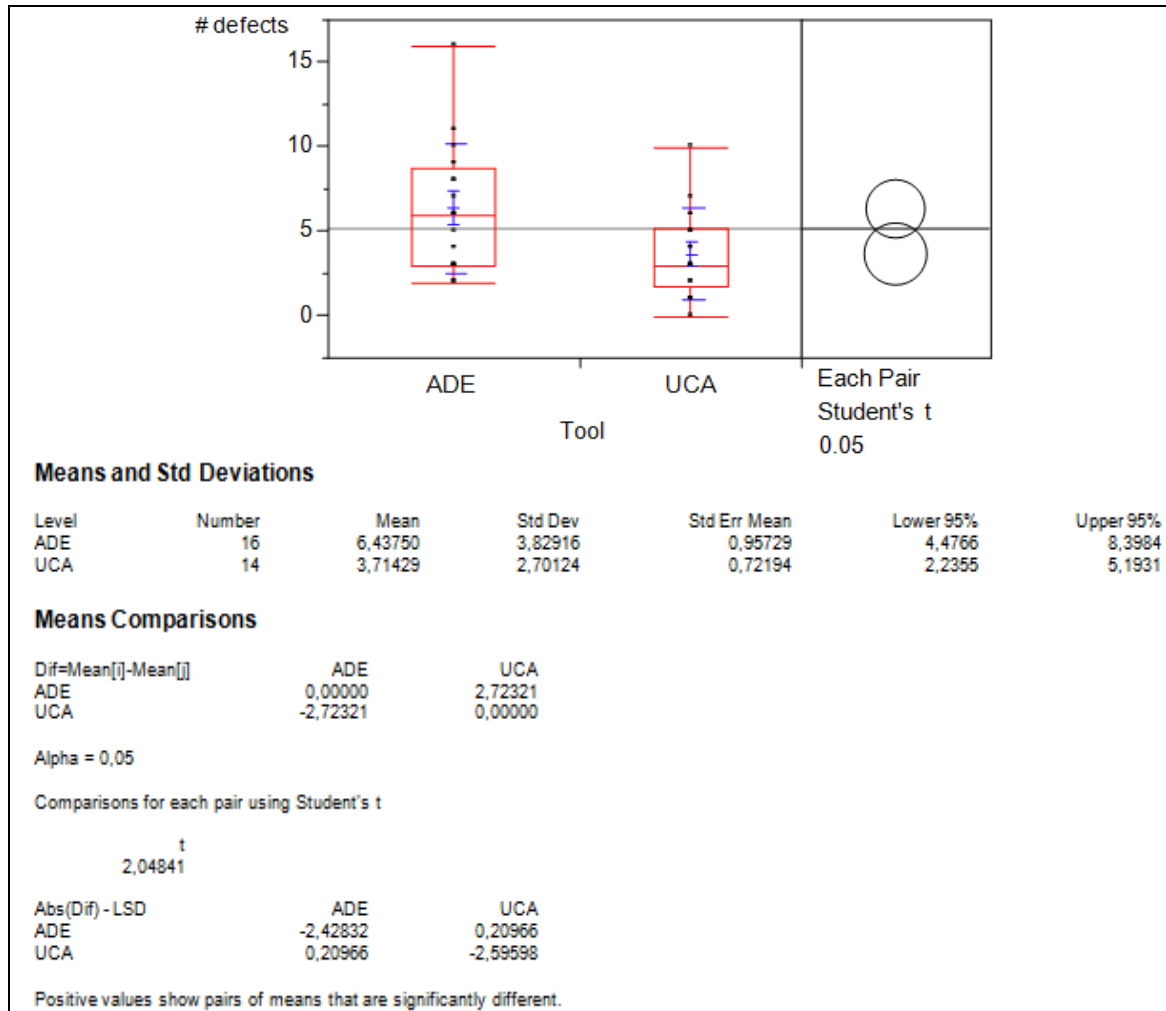


Figura 7-4 - Boxplots relativos aos defeitos reportados nas especificações de caso de uso construídas com as ferramentas EDA e UCA (tela da ferramenta JMP v.4)

Uma questão crucial que emerge nesse contexto é: os defeitos detectados nos diagramas foram introduzidos durante o processo de criação destes no primeiro estudo com o uso da ferramenta EDA ou a ferramenta UCA ajudou os desenvolvedores a remover os defeitos previamente existentes na especificação original de requisitos usada como base para criação dos diagramas? É importante lembrar que na análise qualitativa do primeiro estudo, ambos os desenvolvedores informaram que UCA apoiou a remoção de defeitos, mas somente essa informação não é suficiente para responder tal questão. Uma investigação adicional foi realizada

para tentar compreender quais defeitos a ferramenta UCA de fato ajudou a remover e quais defeitos foram introduzidos com o uso da ferramenta EDA.

Para listar os defeitos existentes nos diagramas construídos com EDA, os pesquisadores usaram a própria ferramenta UseCaseAgent para editar tais diagramas, mais precisamente os diagramas da primeira rodada do primeiro estudo. Vale a pena lembrar que os diagramas da segunda rodada do primeiro estudo já haviam sido construídos com a ferramenta UseCaseAgent, logo os defeitos existentes na especificação original do módulo financeiro que causavam algum tipo de não-conformidade com o metamodelo UCModel foram removidos durante o processo de criação da especificação.

Ao editar os diagramas especificados com EDA usando a ferramenta UseCaseAgent, são listados os erros de não conformidade com o metamodelo UCModel que devem ser corrigidos. Essa análise foi feita para os oito casos de uso do primeiro estudo e não somente para aqueles inspecionados no segundo estudo. A partir da lista de defeitos de cada diagrama apontados pela ferramenta UseCaseAgent, a especificação original foi analisada para verificar se esses defeitos pertenciam de fato ao documento original ou foram introduzidos pelo desenvolvedor ao transcrever o caso de uso em modo textual da especificação original para o diagrama de atividades usando EDA. A Tabela 7-7 apresenta os resultados dessa análise.

Tabela 7-7 - Defeitos nos requisitos detectados pela ferramenta UseCaseAgent

Caso de uso	Nível de complexidade	Defeitos apontados por UCA	Defeitos introduzidos pelo desenvolvedor	Defeitos previamente existentes no documento de requisitos original
6	Baixa	5	2	3
15	Alta	6	3	3
36	Média-Alta	11	5	6
40	Média-Baixa	7	6	1
16	Média-Baixa	14	9	5
27	Média-Baixa	6	4	2
35	Média-Alta	5	2	3
17	Alta	11	5	6
Total	-	65	36	29

Os dados obtidos a partir dessa análise reforçam o resultado estatístico do segundo estudo: diagramas de atividades de casos de uso especificados com a ferramenta UseCaseAgent tendem a apresentar um número menor de defeitos quando comparados com diagramas de atividades de casos de uso especificados sem esse apoio. No contexto desse estudo, as principais razões para esse comportamento são:

- O metamodelo UCModel apóia a remoção de defeitos de requisitos;

- A ferramenta UseCaseAgent previne a introdução de defeitos nos diagramas de atividades de casos de uso quando comparado ao uso de um editor comum de diagramas de atividades, e;
- A ferramenta UseCaseAgent evita transformações desnecessárias, o que apóia a afirmação de que parte dos defeitos existentes em determinado artefato é introduzido por falhas humanas durante o processo de transformação desses artefatos (RAMLER *et al.*, 2010).

A análise das categorias dos 29 defeitos encontrados no documento de requisitos original (Tabela 7-7, quinta coluna) mostrou que 8 eram fatos incorretos (2 relacionados a desvios equivocados no controle de fluxo e 6 relacionados a descrições de ações que juntavam ações de tipos diferentes) e 21 eram omissões, todos relacionados a comportamentos sistêmicos não explicitamente especificados.

A existência dessas omissões pode ter vários impactos negativos, como por exemplo:

- Os desenvolvedores podem ter múltiplas interpretações do que deve ser projetado/implementado/testado;
- Detalhes sobre os procedimentos ou restrições associadas a esses comportamentos sistêmicos podem ser negligenciados;
- A geração semi-automática de outros artefatos a partir do modelo de requisitos funcional (como por exemplo, modelos de projeto estruturais ou comportamentais, casos de teste ou procedimentos de teste) pode se tornar incompleto, e;
- O conhecimento tácito do domínio pode influenciar na qualidade da especificação.

7.4 Ameaças à Validade dos Estudos

As principais ameaças relacionadas à validade dos dois estudos realizados são listadas a seguir:

- **Medição do tempo de especificação e inspeção:** foi solicitado aos participantes que fossem o mais preciso possível com relação à medição do tempo gasto nas tarefas de especificação (primeiro estudo) e inspeção (segundo estudo). Entretanto, não há garantia quanto à exatidão do tempo reportado pelos participantes;

- **Efeitos colaterais do treinamento:** a mitigação desse risco foi feita ministrando treinamentos equivalentes, com os mesmos exemplos e os mesmos instrutores. A influência da experiência do instrutor foi mitigada através da organização do material de treinamento, que procurou definir o que deveria ser apresentado, tanto em termos teóricos como nos exemplos, para minimizar possíveis desvios desses tópicos;
- **Aprendizado das ferramentas:** os desenvolvedores que participaram do estudo estão habituados, no cotidiano do projeto, a trabalhar com ferramentas para modelagem. Entretanto, esses desenvolvedores não tinham experiência na utilização das ferramentas que seriam usadas para especificação dos casos de uso (UCA e EDA). Havia, portanto, um risco de despender um tempo maior para especificação dos primeiros casos de uso em virtude do aprendizado relativo à manipulação dos elementos da interface dessas ferramentas . Para mitigar esse risco foi solicitado que os participantes especificassem, inicialmente, dois casos de uso que não foram incluídos na análise dos dados (casos de uso 10 e 26). Essa ameaça afeta somente o primeiro estudo;
- **Editor BOUML:** embora a ferramenta BOUML (PAGÉS, 2011) utilize uma interface gráfica semelhante a outras ferramentas UML (barra de ferramentas apresentada de acordo com o tipo do diagrama, janelas *pop-up* para edição das propriedades dos elementos, *drag-and-drop* de elementos gráficos, dentre outros), não é possível ter garantias de que o tempo gasto com o uso dessa ferramenta específica teria a mesma ordem de grandeza se a criação dos diagramas fosse feita com outra ferramenta UML. Nesse caso, a influência da qualidade da ferramenta no desempenho do participante pode ser minimizada com o uso de várias ferramentas UML distintas. Entretanto, esse arranjo requer um aumento substancial no número de participantes do estudo;
- **Tamanho da amostra:** pequenas amostragens são um problema conhecido em experimentos na área de Engenharia de Software e difíceis de serem superados. Neste caso explorou-se o máximo possível a participação através do arranjo de distribuição dos casos de uso, o que permitiu gerar 32 pontos de observação considerando 8 participantes;

- **Classificação das discrepâncias em defeitos ou falsos positivos:** poucas discrepâncias não tiveram consenso entre os pesquisadores sobre se estas eram ou não defeitos. Entretanto, esse mecanismo de avaliação precisa ser aprimorado em estudos futuros, com a participação de outros pesquisadores, a fim de tornar o resultado da reunião de discussão mais preciso e/ou a utilização dos defeitos detectados no segundo estudo como um ponto de partida para futuras avaliações;
- **Medição da complexidade dos casos de uso:** em um cenário ideal, a complexidade ou o tamanho dos casos de uso deveria ser obtido através de algum método, com evidências na literatura técnica, para medição desses artefatos. Entretanto, como a especificação dos casos de uso usada no estudo foi realizada dentro do mesmo contexto no que diz respeito ao domínio do problema, tecnologias empregadas e recursos humanos, o risco de comparação entre esses casos de uso usando as medidas obtidas com o método *ad-hoc* descrito na seção 7.2.2, é minimizado, e;
- **Medição da experiência dos participantes:** a mitigação desse risco foi feita através da avaliação dos perfis dos participantes por dois pesquisadores envolvendo, ainda, um terceiro pesquisador para avaliar e referendar a classificação final.

7.5 Conclusão

Os resultados do primeiro estudo sugerem a viabilidade no uso da abordagem proposta neste trabalho apoiada pela ferramenta UseCaseAgent. O segundo estudo, que utilizou-se dos artefatos gerados no primeiro estudo, indicou uma redução de 22,8% dos defeitos nos casos de uso especificados de acordo com o UCModel se comparado aos casos de uso especificados de forma *ad-hoc*. A comparação entre os número de defeitos reportados pelas inspeções nas especificações construídas de forma *ad-hoc* e o número de defeitos reportados pelas inspeções nas especificações construídas de acordo com o UCModel mostra que há uma diferença estatisticamente significativa ($\alpha = 0.05$) a favor do segundo.

Entretanto, a repetição do primeiro estudo deve ser realizada com um número maior de participantes para avaliar as tendências lá observadas e que não puderam ser estatisticamente testadas devido ao pequeno tamanho da amostra. Com relação ao segundo estudo, uma repetição com um maior número de participantes também

pode fortalecer os resultados aqui apresentados. A repetição desses dois estudos pode ser realizada reusando o planejamento e a instrumentação já definidos nos estudos aqui relatados, bastando apenas alterar a distribuição dos artefatos pelo novo grupo de participantes seguindo os mesmos critérios adotados desses estudos.

Apesar das limitações dos estudos, acreditamos que os resultados obtidos possam contribuir para o julgamento da viabilidade da abordagem proposta nesta tese.

8 Conclusão e Trabalhos Futuros

Neste capítulo as conclusões desta tese são apresentadas, resumindo sua motivação e proposta, e destacando suas contribuições. Os trabalhos futuros indicam direções que podem ser tomadas no sentido de dar continuidade à pesquisa aqui apresentada.

8.1 Considerações Finais

Há alguns anos as tecnologias Web eram vistas como meros coadjuvantes no cenário de desenvolvimento de software e, nas organizações, eram responsáveis por aplicações cuja missão tinha pouca interseção com os objetivos do negócio. Hoje esse cenário mudou radicalmente e a demanda por aplicações Web complexas vem sendo acompanhada pela necessidade de abordagens capazes de garantir a qualidade de tais aplicações.

Esta tese tratou de questões relacionadas à especificação dos requisitos funcionais das aplicações Web e seu desdobramento em termos de garantia da qualidade da aplicação. Assim, este trabalho definiu e avaliou uma abordagem para especificação de requisitos funcionais de aplicações Web alinhada aos conceitos explorados pelos métodos Web contemporâneos e estruturada visando prover um arcabouço para a garantia da qualidade da especificação e do produto final.

A experimentação, explorada através de uma metodologia baseada em evidências, apoiou as atividades de pesquisa no escopo desta tese desde a concepção da abordagem, onde foram conduzidos três estudos de observação que forneceram indicações sobre o foco a ser adotado na abordagem, até a sua avaliação, onde foram realizados dois estudos envolvendo a abordagem proposta, o metamodelo UCMModel e a ferramenta UseCaseAgent.

Nesse sentido, as seções a seguir apresentam, respectivamente, as contribuições desta pesquisa, suas limitações e questões em aberto e as possibilidades de trabalhos futuros.

8.2 Contribuições da Pesquisa

As contribuições conseguintes do desenvolvimento desta pesquisa são:

- Definição de uma abordagem para análise, classificação e especificação de requisitos funcionais alinhada aos conceitos explorados pelos métodos Web contemporâneos e estruturada visando prover um arcabouço para a garantia da qualidade da especificação e do produto final, e envolvendo:
 - Elaboração de um conjunto de diretrizes para tratamento das regras de negócio/domínio, de forma consistente e integrada com a perspectiva de tratamento dos requisitos, visando destacar os elementos da especificação impactados por essas regras;
 - Estruturação do metamodelo UCMModel, que permite a organização das especificações de casos de uso segundo um conjunto de critérios e restrições bem definido e oferece a estruturação sintática e semântica a partir do qual é possível tratar questões relacionadas à garantia da qualidade;
 - Elaboração, a partir da representação semântica proporcionada pelo metamodelo UCMModel, de um conjunto de orientações voltadas para a redação das descrições das ações do caso de uso e que foram usadas como base para a definição de algumas questões de uma técnica de inspeção de diagramas de atividades baseada em *checklist* (MELLO, 2010).
 - Especificação e implementação da ferramenta UseCaseAgent, destinada à criação e verificação sintática de especificações de casos de uso, que implementa totalmente a estrutura e restrições definidas no metamodelo UCMModel;
 - Especificação e implementação da ferramenta ModelT2, destinada à geração de modelos preliminares de testes funcionais a partir de modelos de especificação de casos de uso, e;
 - Integração dos modelos de testes gerados a partir dos modelos de especificação à ferramenta TDE/UML[®], para obtenção, de forma automática, de planos de testes funcionais contendo casos e procedimentos de testes.
- Definição de pacotes de laboratório abrangendo o planejamento, execução e análise de estudos de observação e de caracterização que apóiam a avaliação e evolução das tecnologias propostas, e;
- Revisão da literatura técnica na área de Engenharia de Aplicações Web com o objetivo de identificar características dos métodos de desenvolvimento Web no que diz respeito ao tratamento de requisitos funcionais. Para tal, foi

definido um protocolo, baseado nos conceitos da revisão sistemática, que pode servir de base para futuras investigações sobre esse tema.

8.3 Limitações

As principais limitações identificadas nessa pesquisa se referem a:

- O uso da abordagem proposta nesta pesquisa tem sua viabilidade associada ao apoio das ferramentas UseCaseAgent e ModelIT². Como essas ferramentas foram desenvolvidas como *plug-ins* de uma ferramenta genérica para modelagem UML, no caso a BOUML (PAGÉS, 2011), a sua evolução fica limitada às características implementadas na BOUML. Além disso, mesmo sendo uma aplicação *open source*, existe o risco de descontinuação da ferramenta BOUML por parte do seu autor;
- O pequeno número de participantes no primeiro estudo de avaliação, apresentado no capítulo 7, não permitiu obter indícios mais fortes da viabilidade da abordagem e do apoio computacional que a compõe. Não foi possível obter respostas para as questões quantitativas relacionadas a custo ou esforço de uso dessa abordagem em comparação a outras existentes. Avaliações adicionais são necessárias visando fortalecer as indicações obtidas, e;
- A obtenção do modelo preliminar de testes, a partir da especificação dos casos de uso de forma automática, nos deu indicações da viabilidade de explorar esse mecanismo para construção do modelo de testes final. Entretanto, apenas a geração dos modelos preliminares de teste foi realizada, não sendo avaliada a continuação das atividades visando a complementação do planejamento dos testes, o que envolve a descrição dos procedimentos e casos de teste, e;
- Os casos e procedimentos de teste são gerados, atualmente, pela ferramenta TDE/UML[©], que devido à sua característica industrial limita a sua utilização de forma mais abrangente.

8.4 Questões em Aberto

No contexto da pesquisa apresentada nesta tese, algumas questões permanecem em aberto e, conseqüentemente, candidatas a futuras investigações:

- A utilização dessa abordagem em diferentes categorias de software e em diferentes escalas também se mostra viável?

- A construção da especificação de casos de uso seguindo a abordagem apresentada nesta teste apresenta benefícios em termos de custo se comparada a uma abordagem *ad-hoc*?
- O modelo de testes gerado a partir da especificação de casos de uso é capaz de apoiar a geração de um conjunto de casos e procedimentos de testes funcionais que cubra todas as regras e restrições estabelecidas nas visões estruturais e comportamentais definidas na especificação?
- A combinação da geração automática dos modelos de teste preliminares com a complementação deste modelo pelo analista de testes e a posterior geração do plano de casos e procedimentos de testes funcionais é efetiva em termos de custo e esforço se comparada a uma abordagem *ad-hoc*?

8.5 Trabalhos Futuros

A abordagem de especificação e garantia da qualidade de aplicações Web proposta nesta teste oferece oportunidades de pesquisa que diz respeito a:

Repetição dos Estudos sobre a Abordagem

Apesar dos indícios de viabilidade e redução de defeitos em documentos de especificação obtidos com os estudos conduzidos no escopo deste trabalho, a repetição desses estudos reusando o mesmo pacote pode fortalecer as evidências obtidas, através do uso de uma quantidade maior de participantes, e indicar direções para o aprimoramento da abordagem.

Avaliação do Arcabouço de Testes Funcionais

Faz-se necessária a avaliação do arcabouço para geração dos casos e procedimentos de testes funcionais, mais precisamente no que diz respeito à: (1) cobertura do modelo de testes em relação às regras e pré/pós condições estabelecidos no modelo de especificação, e; (2) adequação do apoio criado no modelo de testes preliminar para auxiliar o analista de testes na sua complementação.

Aprimoramento do Metamodelo de Testes Funcionais

O metamodelo TDE implementa o modelo de testes com uma visão caixa-preta do sistema. Essa abordagem é adequada na exploração de cenário onde a interface reflete totalmente o comportamento esperado do sistema. Nesse contexto, a oportunidade de pesquisa esta no aprimoramento do metamodelo TDE para tratamento das ações e estados internos do sistema visando a avaliação de pós-

condições de forma independente da interface deste com o mundo externo e a adaptação de ModelT² para refletir esse novo metamodelo. A avaliação do arcabouço para geração de casos e procedimentos de testes funcionais poderia ser replicada nesse novo contexto.

Técnicas de Inspeção

Definição ou adaptação de técnicas de inspeção, alinhadas aos conceitos definidos na abordagem, e que possam explorar os elementos que a compõem: modelos de especificação de casos de uso, modelos conceituais, regras, dentre outras.

Tratamento de requisitos não-funcionais

A especificação de requisitos não-funcionais, como características relacionadas à ubiquidade, no escopo da abordagem proposta requer uma avaliação do impacto da incorporação desses novos requisitos nos conceitos nos quais a abordagem se apóia. A partir dessa avaliação será possível definir mecanismos que permitam a especificação desse tipo de requisito à luz dos conceitos já existentes ou a adaptação da abordagem de acordo com esse novo contexto de atuação.

Para finalizar, entendemos que a abordagem proposta nesta tese visa contribuir com os avanços e com o corpo de conhecimento relacionado à Engenharia de Aplicações Web no que diz respeito ao desenvolvimento de aplicações. Assim, esta pesquisa vai ao encontro dos avanços já realizados nessa área e procura oferecer uma solução relacionada ao tratamento de requisitos que esteja alinhada com os conceitos que permeiam os métodos de desenvolvimento Web propostos na literatura técnica e nos quais foram observadas lacunas relacionadas ao tema central dessa pesquisa, conforme visto no capítulo 3. Além disso, entendemos que a abordagem proposta nesta tese, embora em sua versão inicial, já consegue proporcionar uma série de oportunidades de pesquisa visando oferecer um conjunto de tecnologias, desenvolvidas com o apoio da experimentação, que permitam a evolução do corpo de conhecimento da Engenharia de Aplicações Web.

REFERÊNCIAS BIBLIOGRÁFICAS

- ABRAHÃO, S., CONDORI-FERNANDEZ, N., OLSINA, L., PASTOR, O. (2003). "Defining and Validating Metrics for Navigational Models", Proceedings of the 9th International Symposium on Software Metrics, pp. 200 - 210.
- ACKERMAN, A., BUCHWALD, L., LEWSKI, F. (1989). "Software Inspections: An Effective Verification Process", IEEE Software, v. 6, n. 3, pp.31-37.
- AGUILAR, J. A., GARRIGÓS, I., MAZÓN, J., TRUJILLO, J. (2010). "An MDA approach for goal-oriented requirement analysis in Web engineering", Journal of Universal Computer Science, v. 16, n. 16, pp. 2475-2495.
- ALBUQUERQUE, P. P. B., MASSOLLAR, J. L., TRAVASSOS, G. H. (2010). "ModelT2: Apoio Ferramental à Geração de Casos e Procedimentos de Testes Funcionais a partir de Casos de Uso", Anais do XXIV Simpósio Brasileiro de Engenharia de Software (SBES'10).
- ALMENDROS-JIMÉNEZ, J. M., IRIBARNE, L. (2005). "Describing Use Cases with Activity Charts", LNCS, v. 3511, pp. 153-167.
- ANDA, B., SJOBERG, D., JORGENSEN, M. (2001). "Quality and understanding of use case models", In: Lindskov Knudsen, J. (ed.), 15th European Conference on Object-Oriented Programming, LNCS, Springer-Verlag, pp. 402–428.
- APFELBAUM, L., DOYLE, J. (1997). "Model Based Testing", Software Quality Week Conference (QWE), Bruxelas, Bélgica.
- ARLOW, J., NEUSTADT, I. (2005). "UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design", Addison-Wesley, 2^a edição.
- ARMOUR, F., MILLER, G.(2001). "Advanced Use Case Modeling", Addison-Wesley.
- BARESI, L., GARZOTTO, F., PAOLINI, P. (2001). "Extending UML for Modelling Web Applications", Proceedings of the 34th Annual Hawaii International Conference on System Science.
- BARESI, L., GARZOTTO, F., MARITATI, M. (2002). "W2000 as a MOF Metamodel", Proceedings of the 2002 World Multiconference on Systemics, Cybernetics and Informatics, volume I.

- BARESI, L., MAINETTI, L. (2006). "W2000 meets J2ME for the fast prototyping of mobile web applications", Proceedings of 24th IASTED International Multi-Conference on APPLIED INFORMATICS, pp. 59-64.
- BASILI, V., ROMBACH, H. (1998). "The TAME Project: Towards Improvement-Oriented Software Environments", IEEE Transactions on Software Engineering, n. 14.
- BELGAMO, A., FABBRI, S. (2004). "GUCCRA: Contribuindo para a Identificação de Defeitos em Documentos de Requisitos Durante a Construção de Modelos de Casos de Uso", Proceedings of the VII Workshop on Requirements Engineering (WER'04), pp. 100-111.
- BELGAMO, A., FABBRI, S., MALDONADO, J. C. (2005). "Avaliando a Qualidade da Técnica GUCCRA com Técnica de Inspeção", Proceedings of the VIII Workshop on Requirements Engineering (WER'05).
- BERNARDI, M. L.; CIMITILE M.; DISTANTE, D.; MAZZONE F. (2010). "Web Application Design Evolution with UWA", Proceedings of 12th IEEE International Symposium on Web System Evolution, pp. 3-12.
- BÉZIVIN, J. (2006). "Model Driven Engineering: An Emerging Technical Space", LNCS, v. 4143, pp. 36-64.
- BIEBER M., GALNARES, R., LU, Q. (1998). "Web Engineering and Flexible Hypermedia", Proceedings of the 2nd Workshop on Adaptive Hypertext and Hypermedia (Hypertext'98).
- BIOLCHINI, J. C. D. A., MIAN, P. G., NATALI, A. C. C., CONTE, T., TRAVASSOS, G. H. (2007). "Scientific research ontology to support systematic review in software engineering" Advanced Engineering Informatics, v. 21, n. 2, pp. 133-151.
- BITTNER, K., SPENCE, I. (2002). "Use Case Modeling", Addison-Wesley.
- BOEHM, B. W., BASILI, V. R. (2001). "Software Defect Reduction Top 10 List", IEEE Computer, v. 34, n. 1, pp. 135-137.
- BOOCH, G., MAKSIMCHUK, R.A., ENGLE, M.W., YOUNG, B.J., CONALLEN, J., HOUSTON, K. (2007). "Object-Oriented Analysis and Design with Applications", Addison-Wesley, 3a. edição.
- CÁCERES, P., De CASTRO, V., VARA, J. M., MARCOS, E. (2006). "Model Transformations for Hypertext Modeling on Web Information Systems",

Proceedings of the 21st ACM Symposium on Applied Computing, v. 2, pp. 1232-1239.

CACHERO, C., GÓMEZ, J., PASTOR, O. (2001). "Conceptual Modeling of Device-Independent Web Applications", IEEE Multimedia, (abr-jun), pp: 26 – 39.

CACHERO, C., KOCH, N. (2002). "Navigation Analysis and Navigation Design in OO-H and UWE", Technical Report 0205, Institute of Computer Science, Ludwig-Maximilians University of Munich.

CACHERO, C., KOCH, N., GÓMEZ, J., PASTOR, O. (2002). "Conceptual Navigation Analysis: a device and platform independent navigation specification", Proceedings of 2nd International Workshop on Web-oriented Software Technology (IWWOST'02).

CARVER, J., JACCHRI, L., MORASCA, S., SHULL, F. (2003). "Issues in Using Students in Empirical Studies in Software Engineering Education", Proceedings of 9th International Software Metrics Symposium, pp. 3-5.

CASALÁNGUIDA, H., DURÁN, J. E. (2009). "Aspect oriented navigation modeling for web applications based on UML", IEEE Latin American Transactions, v. 7, n. 1, pp. 92-100.

CASTRO, de V., MARCOS, E., CÁCERES, P. (2004). "A User Service Oriented Method to model Web Information Systems", Proceedings of the 5th International Conference on Web Information Systems Engineering (WISE'04), pp. 41-52.

CERI, S., FRATERNALI, P., BONGIO, A. (2000). "Web Modeling Language (WebML): a Modeling Language for Designing Web Sites", WWW9 Conference, Amsterdam.

CHEN, P. (1976). "The Entity-Relationship Model – Toward a Unified View of Data". ACM Transaction on Database Systems, v.1, n. 1, pp. 9-36.

COCKBURN, A. (2000). "Writing Effective Use Cases", Addison-Wesley.

CONALLEN, J. (2002). "Building Web Applications with UML". Addison-Wesley, ISBN 0-201-73038-3, 2^a edição.

CONSTANTINE, L., YOURDON, E. (1979). "Structured Design", Prentice Hall, 1979.

CONTE, T., MENDES, E., TRAVASSOS, G. H. (2005). "Processos de Desenvolvimento para Aplicações Web: Uma Revisão Sistemática". Anais do XI Simpósio Brasileiro de Sistema Multimídia e Web (WebMedia'05), pp. 107-116.

- COX, K., AURUM, A., JEFFERY, R. (2004). "An Experiment in Inspecting the Quality of Use Case Descriptions", Proceedings of 7th International Conference on Requirements Engineering: Foundation for Software Quality, v. 36, n. 4, pp. 211-229.
- COX, K., PHALP, K., SHEPPERD, M. (2001). "Comparing Use Case Writing Guidelines", Journal of Research and Practice in Information Technology, pp. 101-112.
- DA CRUZ, A. M. R., FARIA, J. P. (2010). "A Metamodel based Approach for Automatic User Interface Generation", LNCS, v. 6394, pp. 256-270.
- DE SILVA, B., GINIGE, A., BAJAJ, S., EKANAYAKE, A., SHIRODKAR, R., SANTA, M. (2009). "A tool to support end-user development of web applications based on a use case model", LNCS, v. 5684, pp. 527-530.
- DE TROYER, O., LEUNE, C. (1997). "WSDM: A User Centered Design Method for Web Sites", Technical Report of Tilburg University, Infolab, Bélgica.
- DIAS NETO, A. C., TRAVASSOS, G. H. (2006). "Maraká: Uma Infra-estrutura Computacional para Apoiar o Planejamento e Controle de Testes de Software", Anais do V Simpósio Brasileiro de Qualidade de Software (SBQS'06), pp. 250-264.
- DIAZ, P., MONTERO, S., AEDO, I. (2005). "Modelling hypermedia and web applications: the Ariadne Development Method", Information Systems, v. 30, no. 8, pp. 649-673.
- DIEV, S. (2006). "Use cases modeling and software estimation: Applying Use Case Points", ACM Software Engineering Notes, v. 31, n. 6, pp. 1-4.
- DESHPANDE, Y., MURUGESAN, S., GINIGE, A., HANSEN, S., SCHWABE, D., GAEDKE, M., WHITE, B. (2002). "Web Engineering", Journal of Web Engineering, v. 1, n. 1, pp. 3-17.
- DURÁN, A., RUIZ, A., TORO, M. (2002). "Implementing Automatic Quality Verification of Requirements with XML and XSLT", Proceedings of 6th ECOOP Workshop on Quantitative Approaches in Object-Oriented Software Engineering (QAOOSE'02)
- DURÁN, A., BERNARDEZ, B., GENERO, M., PIATINNI, M. (2004). "Empirically Driven Use Case Metamodel Evolution", LNCS, v. 3273, pp. 1-11.

- ESCALONA, M. J., MEJÍAS, M., TORRES, J., REINA, A. M. (2003). "The NDT development process", Proceedings of IV International Conferences on Web Engineering. LNCS, vol. 2722, pp. 463–7.
- ESCALONA, M. J., KOCH, N. (2004). "Requirements Engineering for Web Applications – A Comparative Study", Journal of Web Engineering, v. 2, n. 3, pp: 193 – 212.
- ESCALONA, M. J., KOCH, N. (2006). "Metamodeling the Requirements of Web Systems", Proceedings of 2nd International Conference on Web Information Systems (WEBIST'06), pp. 310-317.
- ESCALONA, M. J., TORRES, J., MEJÍAS, M., GUTIÉRREZ, J. J., VILLADIEGO, D. (2007). "The Treatment of Navigation in Web Engineering", Advances in Engineering Software, v. 38, n. 4, pp. 267-282.
- ESCALONA, M. J., ARAGÓN, G. (2008). "NDT. A Model-driven Approach for Web Requirements", IEEE Transactions on Software Engineering, v. 34, n. 3, pp. 377-394.
- FAGAN, M. E. (1976). "Design and Code Inspection to Reduce Errors in Program Development", IBM Systems Journal, v. 15, n. 3, pp. 182-211.
- FRANCE, B., RUMPE, R. (2007). "Model-driven Development of Complex Software: A Research Roadmap", Proceeding of Future of Software Engineering (FOSE), pp. 37-54.
- FREIRE, A. P., GOULARTE, R., FORTES, R. P. M. (2007). "Techniques for Developing More Accessible Web Applications: A Survey Towards a Process Classification", Proceedings of 25th ACM International Conference on Design of Communication (SIGDOC'07).
- GARRIGÓS, I., MAZÓN, J. N., TRUJILLO, J. (2009). "A requirement analysis approach for using i* in web engineering", LNCS, v. 5648, pp.151-165.
- GARZOTTO, F., PAOLINI, P., SCHWABE, D. (1993). "HDM – A Model-based Approach to Hypertext Application Design", ACM Transactions on Information Systems, v. 11, n. 1, pp: 1-26.
- GINIGE, A., MURUGESAN, S. (2001). "Web Engineering: an Introduction", IEEE Multimedia, v. 8, n. 1, pp: 14 – 18.
- GÓIS, F., FARIAS, P., OLIVEIRA, R. (2010). "Test Script Diagram – Um modelo para geração de scripts de testes", Anais do IX Simpósio Brasileiro de Qualidade de Software (SBQS'10), pp. 73-87.

- GREGOLIN, R., DEBONI, J. E. Z. (2008). "Inspeção de Qualidade em Descrições de Casos de Uso: Uma Proposta de Modelos e Artefatos", Anais do VII Simpósio Brasileiro de Qualidade de Software (SBQS'08), Florianópolis, Brasil.
- GUTIÉRREZ, J. J., NEBUT, C., ESCALONA, M. J., MEJÍAS, M., RAMOS, I. M. (2008). "Visualization of use cases through automatically generated activity diagrams", Proceedings of the 11th International Conference on Model Driven Engineering Languages and Systems (MODELS'08), pp: 83-96.
- HALPIN, T. (2006). "Business Rule Modality", Proceedings of 11th International Workshop on Exploring Modeling Methods in Systems Analysis and Design (EMMSAD).
- HAN, W.-M., HUANG, S.-J (2007). "An empirical analysis of risk components and performance on software projects". Journal of Systems and Software, v. 80, n. 1, pp. 42-50.
- HARTMANN, J., VIEIRA, M., FOSTER, H., RUDER, A. (2005), "A UML-based approach to system testing", Journal of Innovations in Systems and Software Engineering, v. 1, n. 1, p.12-24.
- HENNICKER, R, KOCH, N. (2000). "A UML-Based Methodology for Hypermedia Design", Proceedings of 3rd International Conference on Unified Modeling Language: Advancing the Standard (UML'00), pp. 410-424.
- IEEE (1998). "IEEE Recommended Practice for Software Requirements Specifications", IEEE Std 830-1998, The Institute of Electrical and Electronics Engineers, <http://standards.ieee.org/findstds/standard/830-1998.html>.
- INSFRAN, E., PASTOR, O., WIERINGA, R. (2002). "Requirements Engineering-Based Conceptual Modeling", Requirements Engineering Journal, v. 7, pp. 61-72.
- ISAKOWITZ, T., STOHR, E., BALASUBRAMANIAN, P. (1995). "RMM: a methodology for structured hypermedia design", Communications ACM, v. 38, n. 8, pp 34 – 44.
- ISO (2001). "Information Technology – Software Product Quality. Part 1: Quality Model", ISO/IEC 9126-1:2001, International Organization for Standardization.
- JACOBSON, I. (1992). "Object-oriented software engineering: a use case driven approach", Addison- Wesley.
- JACOBSON, I., BOOCH, G., RUMBAUGH, J. (1999). "The Unified Software Development Process", Addison-Wesley.

- JURISTO, N., MORENO, A. M., VEGAS, S. (2004). "Reviewing 25 years of testing technique experiments", *Empirical Software Engineering: An International Journal*, n. 9, pp.7-44.
- KANJILAL, A., SENGUPTA, S., BHATTACHARYA, S. (2009). "Analysis of complexity of requirements: A metrics based approach", *Proceedings of the 2nd India Software Engineering Conference (ISEC'09)*, pp 131-132.
- KAPPEL, G., PRÖLL, B., REICH, S., RETSCHITZEGGER, W. (2006). "An Introduction to Web Engineering". In: Kappel, G., Pröll, B., Reich, S., Retschitzegger, W. (eds), *Web Engineering: The Discipline of Systematic Development of Web Applications*, John Wiley & Sons, ISBN: 978-0470015544.
- KITCHENHAM, B., (2004). "Procedures for Performing Systematic Reviews", *Joint Technical Report Keele University TR/SE-0401 and NICTA Technical Report 0400011T.1*, Keele University and NICTA.
- KOCH, N., ZHANG, G., ESCALONA, M. J. (2006). "Model transformations: from requirements to web system design", *Proceedings of 6th International Conference on Web Engineering (ICWE'06)*, pp. 281–288.
- KOSTERS, G., SIX, H. W., WINTER, M. (2001). "Coupling Use Cases and Class Models as a Mean for Validation and Verification of Requirements Specifications", *Requirements Engineering*, v. 6, n. 1, pp. 3–17.
- KRUCHTEN, P. (2004). "The Rational Unified Process: an Introduction", Addison-Wesley, 3^a edição.
- KULAK, D., GUINEY, E. (2003). "Use Cases: Requirements in Context", Addison-Wesley, 2^a edição.
- LAMSWEERDE, A. van (2009). "Requirements Engineering", John Wiley & Sons, ISBN: 978-0-470-01270-3.
- LARMAN, C. (2003). "Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and the Iterative Development", Prentice Hall PTR, 3^a edição.
- LAVAZZA, L., BIANCO, V., GARAVAGLIA, C. (2008), "Model-based Functional Size Measurement", *Proceedings of the 2nd ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'08)*.
- LAVAZZA, L., ROBILOLO, G. (2010). "Introducing the evaluation of complexity in functional size measurement: A UML-based approach", *Proceedings of the 4th*

ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'10).

LEE, H., LEE, C., YOO, C. (1998). "A Scenario-based Object-oriented Methodology for Developing Hypermedia Information Systems". Proceedings of 31st Annual Conference on Systems Science.

LEFFINGWELL, D., WIDRIG, D. (2003). "Managing Software Requirements: A Use Case Approach", Addison-Wesley, 2^a edição.

LINEHAN, M. H. (2008). "SBVR Use Cases. Rule Representation, Interchange and Reasoning on the Web", LNCS, v. 5321/2008, pp. 182-196, Springer, Heidelberg.

LOWE D., EKLUND J. (2002). "Client Needs and the Design Process in Web Projects", Web Engineering Track of the WWW2002 Conference.

LOZINSKY, S. (1998). "Enterprise-wide Software Solutions: Integration Strategies and Practices", Addison-Wesley, ISBN 978-0201309713.

LU, C., SONG, I. (2008). "A Comprehensive Aspect-Oriented Use Case Method for Modeling Complex Business Requirements", Advances in Conceptual Modeling – Challenges and Opportunities, LNCS, v. 5253, pp. 133-143.

LUNA, E. R., GARRIGÓS, I., ROSSI, G. (2010). "Capturing and validating personalization requirements in web applications", Proceedings of the 1st International Workshop on the Web and Requirements Engineering, pp. 13-20.

LUNA, E. R., GARRIGÓS, I., GRIGERA, J., WINCKLER, M. (2010). "Capture and Evolution of Web Requirements Using Webspec", Proceedings of the 10th International Conference on Web Engineering (ICWE'10), pp. 173-188.

MAFRA, S., BARCELOS, R., TRAVASSOS, G. H. (2006). "Aplicando uma Metodologia Baseada em Evidência na Definição de Novas Tecnologias de Software", Anais do XX Simpósio Brasileiro de Engenharia de Software (SBES'06), pp. 239 – 254, Florianópolis, Brasil.

MAFRA, S., TRAVASSOS, G. H. (2006). "Leitura Baseada em Perspectiva: A Visão do Projetista Orientada a Objetos", Anais do V Simpósio Brasileiro de Qualidade de Software (SBQS'06), Vitória, Brasil.

MELLO, R. M., MARTINHO, W., TRAVASSOS, G. H. (2010). "Inspeção de Diagramas de Atividades da Especificação de Requisitos", Anais do XXIV Simpósio Brasileiro de Engenharia de Software (SBES'10), Salvador, Brasil.

- MELLO, R. M. (2011). "Uma Técnica para Inspeção de Diagramas de Atividades", Dissertação de mestrado, Programa de Engenharia de Sistemas e Computação (PESC), COPPE/UFRJ.
- MENDES, E., MOSLEY, N., COUNSELL, S. (2005). "The Need for Web Engineering: an Introduction", Web Engineering - Theory and Practice of Metrics and Measurement for Web Development, Springer-Verlag, ISBN: 3-540-28196-7, capítulo 1.
- MOLINA, F., PARDILLO, J., TOVAL, A. (2008). "Modelling Web-Based Systems Requirements Using WRM", Web Information Systems Engineering - WISE 2008 Workshops, LNCS, v. 5176, pp. 122-131.
- MONTERO, S., DIAZ, P., AEDO, I. (2006). "From requirements to implementations: A model-driven approach for web development", European Journal of Information Systems, v. 16, n. 4, pp. 407-419.
- MORENO, N., FRATERNALLI, P., VALLECILLO, A. (2006). "A UML 2.0 profile for WebML modeling", Proceedings of the 6th International Conference on Web Engineering (ICWE'06).
- MORENO, N., FRATERNALLI, P., VALLECILLO, A. (2007). "WebML modeling in UML", IET Software, v. 1, n. 3, pp. 67-80.
- MYERS, G. J. (2004). "The Art of Software Testing", John Wiley & Sons, 2a. edição.
- NAKATANI, T., URAI, T., OHMURA, S., TAMAI, T. (2001). "A requirements description metamodel for use cases", Proceeding of 8th Asia-Pacific Software Engineering Conference, pp. 251-258.
- MUSE, J. D. (1992). "The Operational Profile in Software Reliability Engineering: An Overview", Proceedings of 3rd International Symposium on Software Reliability Engineering, pp. 140-154.
- OFFUT, J. (2002). "Quality attributes of Web software applications", IEEE Software, v. 19, n. 2.
- OGATA, S., MATSUURA, S. (2009). "An evaluation of a use case driven requirements analysis using web UI prototype generation tool", Proceedings of the 9th WSEAS International Conference on Applied Computer Science (ACS'09), pp. 235-240.
- OGATA, S., MATSUURA, S. (2010). "Evaluation of a use-case-driven requirements analysis tool employing web UI prototype generation", WSEAS Transactions on Information Science and Applications, v. 7, n. 2, pp. 273-282.

- OGREN, I. (2000). "On principles for model-based systems engineering", *Systems Engineering Journal*, v. 3, n. 1, pp. 38-49.
- OLSINA, L. (1998). "Building a Web-based Information System applying the Hypermedia Flexible Process Modeling Strategy", *Proceeding of 1st International Workshop on Hypermedia Development (Hypertext'98)*, Pittsburg, EUA.
- OMG (2008). "Semantics of Business Vocabulary and Business Rules – version 1.0", Object Management Group, <http://www.omg.org/spec/SBVR/1.0/>.
- OMG (2010a). "Unified Modeling Language Superstructure - version 2.3", Object Management Group, <http://www.omg.org/spec/UML/2.3/>.
- OMG (2010b). "Object Constraint Language – version 2.2", Object Management Group, <http://www.omg.org/spec/OCL/2.2/>.
- PAGÉ, B. (2011). BOUML, <http://bouml.free.fr>.
- PASTOR, O., ABRAHÃO, S. M., FONS, J. J., RAMÓN S. (2001). "An Object-Oriented Approach to Automate Web Applications Development", *Proceedings of 2nd International Conference on Electronic Commerce and Web Technologies, LNCS*, v. 2115, pp. 16-28.
- PASTOR, O. (2004). "Fitting the Pieces of the Web Engineering Puzzle", *Palestra convidada, XVIII Simpósio Brasileiro de Engenharia de Software (SBES'04)*, Brasília, Brasil.
- PATERNO, F., MANCINI, C., MENICONI, S. (1997). "ConcurTaskTrees: a Diagrammatic Notation for Specifying Task Models", *INTERACT'97*, Chapman & Hall, pp. 362-369
- PERONE, V., PAOLINI, P. (2003). "An Approach for Designing Ubiquitous Web Applications: A Case Study", *Proceedings of IASTED International Conference on Communications, Internet and Information Technology (CIIT'03)*.
- PORTER, A. A., VOTTA, L. G., BASILI, V. R. (1995). "Comparing detection methods for software requirements inspections: a replicated experiment", *IEEE Transactions on Software Engineering*, v.21, n. 6, pp. 563-575.
- PRECIADO, J. C., LINAJE, M., MOLARES-CHAPARRO, R., SANCHEZ-FIGUEROA, F., ZHANG, G., KROIB, C., KOCH, N. (2009). *Proceedings of 8th International Conference on Web Engineering (ICWE'08)*, pp. 148-154.
- PRESSMAN, R. (2000). "What a tangled Web we weave", *IEEE Software*, v. 17, n. 1, pp. 18-21.

- PRESSMAN, R. (2000). "Software Engineering: A practitioner's approach", McGraw-Hill, 5ª edição, ISBN 978-0077096779.
- QURESHI, A., PERINI, A. (2009). "Engineering adaptive requirements," ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems, pp.126-131.
- RAMLER, R., KLAMMER, C., NATSCHLAGER, T. (2010). "The usual suspects: A case study on delivered defects per developer", Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM'10).
- ROBERTSON, S., ROBERTSON, J. C. (2006). "Mastering the Requirements Process", Addison-Wesley, 2ª edição.
- ROBERTSON, J., ROBERTSON, S. (2010). "Volere Requirements Specification Template", 15ª edição, <http://www.volere.co.uk/templates.htm>.
- ROSENBERG, D., STEPHENS, M. (2007). "Use Case Driven Object Modeling with UML: Theory and Practice", Apress.
- RUI, K., BUTLER, G. (2003). "Refactoring Use Case Models: The Metamodel", Proceedings of 26th Australasian Computer Science Conference, pp. 484-491.
- SANTOS, P. S. M. ; TRAVASSOS, G. H. (2010). "Inspeção de Qualidade em Descrições de Casos de Uso: uma Avaliação Experimental em um Projeto Real", Anais do IX Simpósio Brasileiro de Qualidade de Software, pp. 261-275, Belém, Brasil.
- SAWYER, P., KOTONYA, G. (2004). "Software Requirements", IEEE Software Engineering Body of Knowledge (SWEBok), capítulo 2.
- SCHNEIDER, G., WINTERS, J. P. (2001). "Applying Use Cases: A Practical Guide", Addison-Wesley, 2ª edição.
- SCHWABE, D., ROSSI, G., BARBOSA, S. (1996). "Systematic Hypermedia Application Design with OOHDM", Proceedings of 7th ACM International Conference on Hypertext.
- SCHWINGER, W., KOCH, N. (2006). "Modeling Web Applications", In: Kappel, G., Pröll, B., Reich, S., Retschitzegger, W. (eds), Web Engineering: The Discipline of Systematic Development of Web Applications, John Wiley & Sons, ISBN: 978-0470015544.

- SHULL, F., CARVER, J., TRAVASSOS, G. (2001). "An Empirical Methodology for Introducing Software Processes", Proceedings of the Joint 8th European Software Engineering Conference (ESEC) and 9th ACM SIGSOFT Symposium on the Foundations of Software Engineering (FSE-9), pp. 288-296.
- SOMÉ, S. (2006). "Supporting use case based requirements engineering", Journal of Information and Software Technology, n. 48, pp. 43-58.
- SOMÉ, S. (2009). "A Meta-Model for Textual Use Case Description", Journal of Object Technology, v. 8, n. 7, pp. 87-106.
- SOMMERVILLE, I. (2006) Software Engineering, Addison-Wesley, ISBN: 978-0-321-31379-9, 8a edição.
- SOUSA, G., SOARES, S., BORBA, P., CASTRO, J. (2004). "Separation of Crosscutting Concerns from Requirements to Design: Adapting an Use Case Driven Approach", Early Aspects 2004: Aspect-Oriented Requirements Engineering and Architecture Design. Workshop at International Conference on Aspect-Oriented Software Development (AOSD'04), pp. 93-102.
- TRAVASSOS, G. H., SANTOS, P. S. M. D., MIAN, P. G., DIAS NETO, A. C., BIOLCHINI, J. C. D. A. (2008). "An Environment to Support Large Scale Experimentation in Software Engineering", Proceedings of the 13th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'08), pp. 193-202.
- TRAVASSOS, G. H., SHULL, F., FREDERICKS, M., BASILI, V. (1999). "Detecting defects in object-oriented designs: using reading techniques to increase software quality", ACM SIGPLAN Notices, v. 34, n. 10, pp. 47-56.
- URBIETA, M., ROSSI, G., GINZBURG, J., SCHWABE, D. (2007). "Designing the interface of rich internet applications", Proceedings of the 2007 Latin American Web Conference (LA-WEB'07), pp. 144-153.
- VALDERAS, P., FONS, J., PELECHANO, V. (2005). "Transforming web requirements into navigational models: AN MDA based approach", LNCS, v. 3716, pp. 320-336.
- VALDERAS, P., PELECHANO, V., PASTOR, O. (2007). "A transformational approach to produce web application prototypes from a web requirements model", International Journal of Web Engineering and Technology, v. 3, n. 1, pp. 4-42.
- VILAIN, P., SCHWABE, D., SIECKENIUS, C. (2000). "A diagrammatic Tool for Representing User Interaction in UML", Proceeding of UML'2000, LNCS, v. 1939, pp. 133-147.

- WAN-KADIR, W. M. N., LOUCOPOULOS, P. (2004). "Relating evolving business rules to software design", *Journal of System Architecture*, n. 50, pp. 367-382.
- WANG, W., GENG, G., ZHOU, M. (2010). "A Rule Repository Model for Rule-Driven Question-Answer-Based Web Applications", *Proceedings of 2010 International Conference on Artificial Intelligence and Computational Intelligence (AICI'10)*, pp. 17-22.
- WANG, H., LI, J., WANG, Q. , YANG, Y. (2010). "Quantitative Analysis of Requirements Evolution across Multiple Versions of an Industrial Software Product", *Proceedings of 17th Asia-Pacific Software Engineering Conference*, pp. 43-49.
- WOOKJIN, L., SANGHYUN, P., KEEYOULL, L., CHUNWOO, L., BYUNGJEONG, L., WOOSUNG, J., TAEKSU, K., HEECHERN, K., CHISU, W. (2005). "Agile development of web application by supporting process execution and extended UML model", *Proceedings of 12th Asia-Pacific Software Engineering Conference, (APSEC)*, pp. 193-200.
- WIEGERS, K. (2003). "Software Requirements", . Microsoft Press, 2^a edição.
- WILLIAMS, C., KAPLAN, M., KLINGER, T., PARADKAR, A. (2005). "Toward Engineered, Useful Use Cases", *Journal of Object Technology*, v. 4, n. 6, pp. 45-57.
- WU, J., FU, S. (2008). "Business Process Modeling Based On Norm Analysis", *Proceedings of 2008 International Seminar on Business and Information Management (ISBIM'08)*, v. 2, pp. 489-493.
- YU, E. S. L. (1997). "Towards modelling and reasoning support for early-phase requirements engineering", *Proceedings of 3rd International Symposium of Requirements Engineering*, pp. 226-235.

APÊNDICE A – Instrumentos dos Estudos de Observação Preliminares

Este apêndice apresenta os instrumentos utilizados nos estudos relatados nesta tese.

A.1. Questionário do 1º estudo de observação preliminar

Nome: _____

1) Da abordagem de Desenvolvimento

A) Do ponto de vista do desenvolvimento do software, a abordagem que vocês usaram se demonstrou adequada? Comente.

B) As dimensões WEB (conceitual, navegação, visualização e estruturação) sugeridas para o processo provocaram algum tipo de dificuldade/facilidade para o desenvolvimento dos artefatos de projeto?

C) Destes artefatos (mapa de atores, diagrama de contexto, mapa de navegação, telas de visualização) qual você considera que não se aplicaria diretamente ao seu projeto? Comente.

D) O que você sugeriria ser modificado na abordagem utilizada para apoiar o desenvolvimento de futuros projetos (ex. notação, modelos, UML, etc.)?

2) Do trabalho da Disciplina:

A) Comente sobre o projeto em geral. Quais os principais problemas encontrados? O que dificultou o andamento das atividades?

B) Em relação às estimativas iniciais, o que você acredita que não foi considerado que poderia ter alterado sua percepção sobre o projeto (tempo, custo, ferramentas, plataformas, etc.)?

C) Como foi a participação da equipe no projeto? Numa escala de 0-10 classifique a participação de cada um (incluindo a sua), segundo sua percepção, no contexto deste projeto.

D) Você usaria os princípios discutidos no curso em futuros projetos? Comente.

A.2. Questionário do 3º estudo de observação preliminar

EEL-884 – Questionário

Esse questionário deve ser preenchido **individualmente** e enviado para jobson@cos.ufrj.br.

Suas respostas NÃO serão usadas para efeito de avaliação do curso e servem apenas para complementar o conjunto de informações referentes às atividades envolvidas no processo de desenvolvimento de aplicações WEB.

Nome da Equipe que participou: _____

- 1) Durante o projeto, indique aqueles que você ajudou a construir ou revisar (0-não construí/revisei, 1-Apenas revisei, 2- Apenas construí, 3-Construí e Revisei)

Diagrama de classes	Diagrama de seqüência	Diagrama de estado	Diagrama de atividades	Mapa de atores	Modelo navegacional

- 2) Na sua percepção, qual foi o esforço para criação desses modelos? (A) alto, (M) médio, (B) baixo.

Diagrama de classes	Diagrama de seqüência	Diagrama de estado	Diagrama de atividades	Mapa de atores	Modelo navegacional

- 3) No seu entendimento, indique na tabela 3 fatores que trouxeram mais dificuldades na construção de cada um dos modelos? Utilize a lista abaixo como roteiro inicial. Acrescente outros se desejar ou tenha percebido no desenvolvimento do trabalho:

- (A) Treinamento (exemplos, qualidade do material disponibilizado, etc.).
- (B) Complexidade do sistema (dificuldade de entendimento do domínio).
- (C) Documento de especificação (documentação confusa e/ou inadequada).
- (D) Falta de experiência na criação do modelo.
- (E) Ferramenta case (componentes inadequados para construção do modelo).
- (F) Complexidade do diagrama.
- (G) Ordem com que os modelos foram construídos
- (H) Divisão de tarefas/responsabilidades entre a equipe
- (I) Perspectiva de projeto usada pela equipe inconsistente
- (J) _____
- (K) _____
- (L) _____

Atenção: a ordem é importante. De cada 3 fatores escolhidos, o 1º fator representa o que trouxe mais dificuldade e o 3º fator menos.

	1º fator	2º fator	3º fator
Diagrama de classes			
Diagrama de seqüência			
Diagrama de estado			
Diagrama de atividades			
Mapa de atores			
Modelo navegacional			

4) Na sua opinião, o que poderia ser feito para diminuir o esforço com estes modelos?

	Iniciativas para diminuição do esforço
Diagrama de classes	
Diagrama de seqüência	
Diagrama de estado	
Diagrama de atividades	
Mapa de atores	
Modelo navegacional	

APÊNDICE B – Análise Qualitativa

Neste apêndice é apresentada a análise qualitativa realizada no terceiro estudo de observação preliminar.

B.1 Introdução

Na análise qualitativa realizada no terceiro estudo de observação preliminar, cada participante respondeu a uma única pergunta, que deveria ser respondida para cada um dos seis modelos elaborados durante o estudo (classes, seqüência, atividades, estado, atores e navegacional):

Na sua opinião de engenheiro de software, o que poderia ser feito para diminuir o esforço/dificuldade com estes modelos?

Dos 24 participantes do estudo, apenas 22 enviaram as respostas solicitadas, sendo que nem todos os 22 participantes enviaram respostas para todos os modelos.

Na seção a seguir são apresentadas as respostas dos participantes (numeradas de 1 a 24).. Nestas respostas estão marcados os trechos analisados pelos pesquisadores, os comentários sobre esses trechos (em itálico) e os códigos e categorias extraídos (em negrito), conforme apresentado no exemplo a seguir:

1)

Diagrama de classes: A diagramação das classes pode ser feita de modo a agrupar classes de modelo, visão e controle [1].

[1][Nesse comentário o participante parece preocupado com a dificuldade de localizar as informações no diagrama de classes. Na visão dele a ferramenta CASE poderia prover algum tipo de diagramação automática (nesse caso baseada no tipo da classe) que facilitasse essa localização]

FERRAMENTA CASE: facilidades para visualização de modelos

B.2 Análise dos Dados

1)

Diagrama de classes: A diagramação das classes pode ser feita de modo a agrupar classes de modelo, visão e controle [1].

[1][Nesse comentário o participante parece preocupado com a dificuldade de localizar as informações no diagrama de classes. Na visão dele a ferramenta CASE poderia

prover algum tipo de diagramação automática (nesse caso baseada no tipo da classe) que facilitasse essa localização]

FERRAMENTA CASE: facilidades para visualização de modelos

Modelo navegacional: Esboço dos modelos navegacionais, em nível de ferramenta, pode ser gerado a partir da criação do mapa de atores e dos diagramas de seqüência [2].

[2][Essa sugestão propõe, em uma análise mais abstrata, a possibilidade explorar a geração de modelos (de forma automática ou semi-automática), a partir de modelos previamente construídos. A partir desse texto podemos extrair também uma sugestão de ordem para a construção dos modelos citados (mapa de atores, diagramas de seqüência => mapa navegacional)]

MODELO: possibilidade de geração (semi) automática, estabelecer uma ordem para construção

2)

Diagrama de classes: Maior nível de detalhes no documento de requisitos [1]. Além disso, este poderia adiantar questões sobre futuras evoluções do software, ou seja, coisas que o cliente já mencionou que seria interessante fazer, mas que não é uma demanda imediata [2]. Nesse sentido, algumas decisões de projeto podem ser tomadas com mais seguranças, Além disso, decisões que podem parecer corretas e que, na verdade, representarão um problema no futuro, podem ser reestruturadas.

[1][Esse comentário sugere que o documento de requisitos pode não estar “adequado” como base para construção dos modelos. Esse problema pode estar na estrutura do documento, no detalhamento insuficiente das informações existentes, na falta de algum tipo de informação, etc.]

DOCUMENTO DE REQUISITOS: necessidade de um nível de detalhamento adequado , estrutura do documento deve estar adequada aos modelos

[2][Nesse trecho o participante sugere que é importante que o documento de requisitos contemple as futuras evoluções como forma de direcionar a modelagem atual no sentido de atender a essas demandas futuras.]

DOCUMENTO DE REQUISITOS: contemplar evoluções previstas

Diagrama de seqüência: Poderia ser feito pela mesma pessoa que desenvolve o diagrama de atividades, visto que este parece representar o diagrama de seqüência numa visão mais alto nível [3].

[3][Esse texto sugere que deve existir alguma preocupação com a distribuição de atividades entre os membros da equipe. Modelos que possuem algo em comum deveriam ser tratados pelos mesmos desenvolvedores. Podemos inferir alguns critérios para esse agrupamento: modelos que trabalham sobre os mesmos aspectos (estrutural ou dinâmico), modelos que trabalham sobre a mesma perspectiva (conceitual, navegação,apresentação, estruturação), modelos que possuem visões complementares (classes e seqüência, etc.)

PROCESSO DE MODELAGEM: critérios objetivos de divisão das atividades (aspectos, perspectivas, complementaridade, etc.)

Modelo navegacional: A disponibilização de uma documentação [4] mais completa e de exemplos aplicados a sistemas maiores [5].

[4][Aqui novamente o participante sugere uma melhor documentação do modelo navegacional. Podemos traduzir isso como a necessidade de conhecer bem o modelo com o qual estamos trabalhando, o que nos remete à questão do treinamento]

TREINAMENTO NO MODELO: guia de referência

[5][O uso de exemplos proposto pelo participante pode estar relacionado à construção de modelos tendo como base outros modelos desenvolvidos para situações semelhantes em outros projetos, o que poderia trazer uma maior segurança para desenvolvedores menos experientes. Vejo aqui dois desdobramentos possíveis: a) consultar modelos existentes para entender como os elementos do modelo foram organizados para representar determinada situação; b) consultar modelos existentes em busca de padrões que possam ser reaproveitados/aplicados no projeto em questão]

TREINAMENTO NO MODELO: uso de uma abordagem baseada em exemplos

3)

Diagrama de classes: Treinar mais a capacidade de olhar para um sistema seguindo o paradigma Orientado a Objeto [1]. Buscar não confundir Classes de um sistema OO com tabelas de banco de Dados, pois como eu estava acostumado com a modelagem de banco de dados eu fiquei com um pouco de dificuldades de enxergar as classes no início do projeto, mas com o desenvolver do mesmo pude gradativamente perceber a diferença.

[1][Esse participante sugere que o domínio por parte do desenvolvedor do paradigma empregado nos modelos (nesse caso OO) é um fator importante, pois o desconhecimento do paradigma ou a influência de outros paradigmas mais familiares ao desenvolvedor trazem dificuldades. Em um nível mais abstrato parece estar relacionado a treinamento no paradigma.]

TREINAMENTO NO PARADIGMA: enfatizar diferenças entre paradigma OO e ER

Diagrama de seqüência: Estudar mais a visão de trocas de mensagens entre usuário, sistema e base de dados. Para diminuir a dificuldade com este modelo restaria apenas estudar mais [2]. O bom deste modelo é que podemos ver o momento em que um objeto é instanciado e quando ele faz acesso à base de dados, sem dizer que melhora e muito enxergar a troca de mensagens.

[2][Esse participante sugere que entender/dominar a sintaxe e a semântica dos elementos que compõem um modelo e a própria visão que o modelo procura capturar é importante para diminuir as dificuldades com o mesmo.]

MODELO: entender/dominar a sintaxe e a semântica dos modelos

Diagrama de estado: No caso do diagrama de estados, particularmente eu precisaria estudar um pouco mais a fundo, pois mesmo conseguindo construí-lo ainda é algo um pouco abstrato para mim enxergar a contribuição que o mesmo dá ao projeto [3]. Por fim eu consigo ver bem a contribuição que os outros modelos dão ao projeto como por exemplo a troca de mensagens, a dinâmica, as restrições de acesso, os caminhos de navegação. Já com o diagrama de estados eu ainda não consigo perceber bem a visão que o mesmo disponibiliza do sistema.

[3][*Esse participante sugere que a falta de entendimento do que o modelo procura capturar dificulta o seu uso. Provavelmente isso ocorre porque a falta desse entendimento influencia diretamente na percepção da utilidade do modelo.*]

TREINAMENTO NO MODELO: enfatizar os aspectos que os modelos procuram capturar e representar, definir as possibilidades de uso dos modelos no contexto do desenvolvimento

MODELO: ter a percepção da utilidade do modelo no contexto do desenvolvimento

Diagrama de atividades: Apenas treinamento diminuiria a dificuldade, visto que a princípio não tive tantas dificuldades com esse modelo; Contudo percebi que meus diagramas neste projeto poderiam ter ficado melhores. **Faltou apenas tempo para usar mais afundo os recursos da ferramenta case [4].**

[4][*Esse trecho sugere que ter o domínio dos recursos da ferramenta CASE facilita a construção dos modelos. Podemos interpretar também como a necessidade de uma ferramenta mais intuitiva ou dotada de mecanismos que agilizem a construção dos modelos*]

FERRAMENTA CASE: domínio dos recursos, mecanismos que agilizem a construção dos modelos.

Mapa de atores: Não tive maiores dificuldades com este modelo, o difícil foi entender material que estava em Espanhol e de certa forma um pouco confuso. Em outras palavras um material em português ou inglês escrito de forma mais simples e com alguns exemplos práticos, isto já serviria para estudar e poder construir mapas de atores mais adequados, pois estes são fundamentais para a construção do Modelo Navegacional.

Modelo navegacional: **Pesquisar sobre modelos navegacionais e participar de outros projetos de sistemas WEB diminuiria as minhas dificuldades [5], visto que até então só tinha participado de desenvolvimento de sistemas desktop, sendo que nestes foi praticamente inexistente a aplicação das técnicas de Engenharia de Software Orientada a Objeto [6].** Durante o curso pude perceber quanto tempo foi perdido nos projetos dos quais participei, tudo isso por conta da falta de planejamento, especificação de requisitos, modelagens ou seja por falta de artefatos adequados.

[5][*Esse trecho sugere que, como o conceito de modelo navegacional está muito associado ao desenvolvimento de aplicações Web, é importante que o desenvolvedor tenha participado em outros projetos de aplicações Web para estar mais familiarizado com esse conceito.*]

MODELO: ter a percepção da utilidade do modelo no contexto do desenvolvimento

[6][*idem comentário 1*]

4)

Diagrama de classes: **Gosto de fazer a comparação das classes nesse diagrama com tabelas em BD por ter mais facilidade com esta visualização [1].** Com isso as relações seriam as chaves estrangeiras.

[1][*Esse comentário contradiz o que foi exposto em 3[1]]. Aqui o participante sugere que o conhecimento do paradigma ER facilitaria a aplicação do paradigma OO. Parece justamente o contrário: a falta de conhecimento do paradigma OO faz com que o desenvolvedor lance mão daquilo que lhe é mais familiar (paradigma ER, nesse caso). O treinamento no paradigma correto parece ser a questão aqui*]

TREINAMENTO NO PARADIGMA: enfatizar as diferenças com outros paradigmas

Mapa de atores: **Pensar hierarquicamente em quais são as entidades que se interagem no sistema [2]** (por qualquer relação que seja)

[2][*O participante sugere uma heurística para criar o mapa de atores: olhar para a hierarquia existente nas entidades que vão interagir com o sistema*]

PROCESSO DE MODELAGEM: definição de heurísticas que apoiem a construção do modelo

Modelo navegacional: Acho que no modelo navegacional deveria existir um **treinamento na ferramenta [3]**

[3][*Fiquei em dúvida se a sugestão está somente relacionada com a ferramenta, como o texto sugere, ou se existe algo também relacionado ao modelo propriamente dito. Prefiro inferir que a dificuldade não está somente no uso da ferramenta CASE, mas também no uso do modelo em si*]

TREINAMENTO NO MODELO: explorar a construção dos modelos propostos usando uma ferramenta CASE

5)

Diagrama de classes: **Uma ferramenta que importasse diretamente dos diagramas de seqüência as operações para as classes [1]**.

[1][*Novamente surge uma referência direta a possibilidade de gerar um modelo a partir de outros*]

MODELO: possibilidade de geração (semi) automática

Diagrama de seqüência: **Treinamento com vasto conjunto de exemplos [2]**, dado que esse diagrama é um dos mais difíceis de absorver, na minha opinião.

[2][*O participante sugere a exploração de exemplos como forma de treinamento no modelo*]

TREINAMENTO NO MODELO: exploração de exemplos

Diagrama de estado: Talvez um **treinamento em grafos [3]**.

[3][*Apesar de referenciar diretamente grafos, a sugestão está relacionada ao entendimento desse modelo, ou seja, treinamento no modelo*]

TREINAMENTO NO MODELO: enfatizar os aspectos que os modelos procuram capturar e representar

Diagrama de atividades: **Treinamento em processos e workflow [4]**, usando muitos **exemplos para fixar os conceitos [5]**.

[4][*Apesar de referenciar workflow, a sugestão está relacionada ao entendimento do modelo, ou seja, treinamento no modelo*]

TREINAMENTO NO MODELO: enfatizar os aspectos que os modelos procuram capturar e representar

[5][Uso de exemplos]

TREINAMENTO NO MODELO: uso de exemplos

Mapa de atores:

Modelo navegacional: Este modelo é muito complicado se usado por completo, acho que usar apenas os componentes mais superficiais como contexto, subsistemas, links de seqüência e de exploração é suficiente para dar uma visão do sistema a ser modelado [5].

[5][Este comentário está relacionado a possibilidade de realizar uma modelagem "parcial". Esse corte pode ser feito sob diversos ângulos: a) selecionar quais elementos modelar baseado em algum critério (maior complexidade, maior criticalidade, maior relevância para o negócio, etc.); b) dado um modelo definir que nível de detalhamento utilizar (visão mais geral ou mais detalhada, etc.)]

PROCESSO DE MODELAGEM: critérios para seleção de modelos mais relevantes dentro do contexto de desenvolvimento, critérios para seleção das características mais relevantes a serem modeladas.

6)

Diagrama de classes: Realizar inspeções periódicas dos requisitos e dos casos de uso, visando verificar consistência e detectar incorreção em estágios iniciais da modelagem [1]

[1][O comentário está um pouco confuso, mas me parece que o participante sugere inspecionar os modelos gerados x requisitos e UCs para detectar inconsistências (e não inspecionar os requisitos e UCs como a primeira frase deixa transparecer)]

GARANTIA DA QUALIDADE: processos de inspeção para detectar inconsistências nos modelos

Diagrama de seqüência: Somente representar em diagramas de seqüência os casos de uso suficientemente complexos, evitando criar diagramas de pouca utilidade [2]

[2][Aqui é sugerido um critério para a modelagem parcial: a complexidade do UC. Tem que ser definido de forma objetiva como "medir" essa complexidade]

PROCESSO DE MODELAGEM: tratamento diferenciado para Casos de Uso complexos (exploração de diagramas de seqüência, por exemplo)

Modelo navegacional: Criar protótipos estáticos do sistema web, o que facilita a verificação das definições de contextos e enlaces navegacionais [3]

[3][Creio que esse comentário foi feito devido a falta de entendimento do uso dos modelos. Não faz sentido criar um protótipo do sistema e *depois* gerar o modelo navegacional (gerar o protótipo a partir dos modelos faz sentido). Não vejo nenhum coding aqui.]

7)

Diagrama de classes: Modelagem progressiva ao longo de um período, ao invés de se partir de um modelo complexo para um modelo simples [1]

[1][Não consegui identificar o que o participante propõe. Para mim não existe nenhum codig aqui]

Diagrama de seqüência: **Realizá-los como complementos ou alternativas aos diagramas de atividades [2].**

[2][O participante sugere que exista algum critério para a seleção dos modelos mais adequados a serem usados. Parece que existe alguma relação com a modelagem parcial citada anteriormente, mas não tenho certeza]

PROCESSO DE MODELAGEM: critérios para seleção dos modelos mais adequados

Mapa de atores : **Desenvolvê-lo em conjunto com o mapa navegacional [3]**

[3][O participante sugere um ordem para construção dos modelos]

PROCESSO DE MODELAGEM: ordem de construção

Modelo navegacional: **Uma simplificação do processo de construção do modelo qualificando características desejáveis na modelagem navegacional de acordo com a criticalidade do projeto [4]. Ter um documento de referência para consultar a forma de modelagem [5]**

[4][Aqui é sugerida novamente a questão da modelagem parcial, focando somente as características mais críticas]

PROCESSO DE MODELAGEM: critérios para seleção das características mais relevantes a serem modeladas

[5][Novamente surge a questão de um guia de referência para os modelos]

TREINAMENTO NO MODELO: guia de referência

8)

Diagrama de classes: Mais **treino, para ganhar mais experiência na construção do modelo [1]. Fazer exercícios e/ou estudar exemplos sobre mais problemas/domínios diferentes, com diferentes aspectos, e diferentes soluções para problemas similares [2].** As apresentações dos trabalhos da presente disciplina foi definitivamente uma ótima maneira de trocar experiência.

[1][Sugere um treinamento no modelo como forma de obter experiência na sua construção. Isso pode ser feito mostrando como situações reais de projeto podem ser tratadas]

[2][Sugere o uso de exemplos como forma de treinamento no modelo.]

TREINAMENTO NO MODELO: exemplos que simulem situações reais de projeto, exercícios que simulem situações reais de projeto, uso de exemplos

Diagrama de seqüência: **Dar um enfoque de nível mais alto [3]** (no caso do nosso projeto).

[3][A sugestão parece estar relacionada com a questão da modelagem parcial, onde o nível de detalhamento é o critério de corte sugerido.]

PROCESSO DE MODELAGEM: critérios para definição do detalhamento dos modelos

Modelo navegacional: **Ajudaria ter uma ferramenta específica para a criação do modelo [4].**

[4][O participante sugere que a adequação da ferramenta o modelo que está sendo construído é importante]

FERRAMENTA CASE: adequação sintática aos modelos

9)

Diagrama de classes: **Maior aprendizado da ferramenta [1] e mais exercícios [2]**

[1][A sugestão é que o treinamento englobe não só a questão do modelo e do paradigma, mas também a ferramenta a ser utilizada]

[2][A sugestão está relacionada ao treinamento no modelo envolvendo a efetiva prática na construção dos modelos. Como a sugestão está genérica, podemos englobar a questão do treinamento OO também]

TREINAMENTO NO MODELO: uso da ferramenta CASE, exploração de exercícios

Diagrama de seqüência: Maior aprendizado da ferramenta e mais exercícios

Diagrama de estado: mais exercícios

Diagrama de atividades: mais exercícios

Mapa de atores mais : exercícios

Modelo navegacional: **Material escrito em português [3],** mais exercícios.

[3][O participante sugere que ter o material de consulta na língua nativa facilitaria a construção]

TREINAMENTO NO MODELO: material na língua nativa

10)

Diagrama de classes: **Este diagrama é válido como ponto de partida do projeto [1].** Porém, algumas classes, métodos ou atributos podem passar despercebidos em um primeiro momento. Considero o esforço de sua criação o maior dentre os modelos citados. **Para diminuir as inconsistências no modelo e torná-lo completo, este deve ser revisado após cada diagrama que tenha sido feito tomando este como base [2]** (seqüência, por exemplo). Isto ajuda a capturar os métodos ou atributos não visualizados inicialmente.

[1] [Essa sugestão está diretamente relacionada a ordem de construção dos modelos]

PROCESSO DE MODELAGEM: ordem de construção

[2] [O participante sugere revisar o diagrama em busca de inconsistência. Dependendo do contexto, essa garantia de consistência poderia ser (parcialmente) feita pela ferramenta CASE.]

GARANTIA DA QUALIDADE: processos de inspeção para detectar inconsistências nos modelos

Diagrama de seqüência: Maior detalhamento do fluxo principal e alternativos dos UCs.

Diagrama de estado: Um texto bem explicado com as mudanças dos estados dos objetos, para que os estados dos objetos capturados durante a leitura dos UCs possam ser verificadas [3].

[3][Esse participante parece sugerir que os conceitos capturados durante a especificação tenham seus estados bem definidos (não necessariamente através de uma máquina de estados)]

DOCUMENTO DE REQUISITOS: detalhar estados dos conceitos capturados

Diagrama de atividades: Um texto bem explicado com os processos da organização, para que as atividades capturadas durante a leitura dos UCs possam ser verificadas [4].

[4][Verificar se os UCs estão de acordo com os processos da organização é uma atividade a ser executada *antes* da modelagem. Podemos inferir que a descrição desses processos e o rastreamento dos UCs nesses processos podem facilitar o entendimento dos UCs]

DOCUMENTO DE REQUISITOS: facilitar o entendimento dos UCS através do mapeamento destes nos processos de negócio

Modelo navegacional: Poderiam ser dados maiores exemplos deste modelo, e exercícios práticos [5].

[5][Treinamento baseado em exemplos e exercícios]

TREINAMENTO NO MODELO: exploração de exemplos e exercícios

11)

Diagrama de classes: Maior discussão prévia sobre o diagrama [1]

[1][Parece que o participante propõe treinamento quando fala em “discussão”. Esse treinamento pode abranger tanto a questão do paradigma quanto o modelo propriamente dito]

TREINAMENTO NO MODELO: enfatizar os aspectos que os modelos procuram capturar e representar

TREINAMENTO NO PARADIGMA: apresentar as principais características do paradigma

Diagrama de seqüência: Criar somente os diagramas mais relevantes [2].

Diagrama de estado: Eliminar diagramas muito simples [2]

Diagrama de atividades: Eliminar diagramas muito simples [2]

[2][Essas sugestões indicam possibilidade de corte na modelagem parcial, usando como critério a relevância do que deve ser modelado]

PROCESSO DE MODELAGEM: critérios para seleção dos modelos mais relevantes, critérios para seleção das características mais relevantes

Mapa de atores: Melhor discussão prévia [3]

[3][Parece que o participante propõe treinamento no modelo quando fala em “discussão”]

TREINAMENTO NO MODELO: enfatizar os aspectos que os modelos procuram capturar e representar

Modelo navegacional: Uma ferramenta que suporte os paradigmas do modelo, como a criação de views das classes [4].

[4][O participante propõe que uma das características da ferramenta CASE seja a sua adequação sintática e semântica aos elementos do modelo construído]

FERRAMENTA CASE: adequação sintática aos modelos

12)

Diagrama de classes: Reutilizar classes diagrama de aplicações de domínio parecidos. Ex, todo sistema WEB tem um modelo de autenticação [1].

[1][Remete ao reuso de modelos prévios bem como a adoção de padrões bem consolidados]

REUSO DE MODELOS: permitir o reuso de modelos e padrões adotados em outros projetos

Diagrama de seqüência: Não vejo necessidade de gerar este diagrama em projetos de pequeno porte [2]. Isto pode ser por que nunca encarei um projeto realmente grande. Pelo que vi em algumas apresentações a troca de mensagens entre as classes eram “abstratas” demais, e em minha opinião, não agregam tanta informação útil ao sistema.

[2][Essa sugestão esta associada à modelagem parcial]

PROCESSO DE MODELAGEM: critérios para seleção dos modelos mais relevantes

Mapa de atores: Com os requisitos já consistentes e levantados acredito que gerar o mapa de ator seja simples. Acredito que não precise ser nada feito para diminuir o esforço dessa tarefa.

Modelo navegacional: Não gerei esse modelo e não vi muita utilidade no contexto dessa aplicação [3]. Pelo que vi em algumas apresentações este modelo de nada agregou ao projeto, se resumindo simplesmente a mostrar os casos de uso com representações de “caixinhas” ao invés de “elipses”.

[3] [Parece estar associado a modelagem parcial, onde o tamanho do projeto influencia na seleção dos modelos]

PROCESSO DE MODELAGEM: critérios para seleção dos modelos mais relevantes

MODELO: percepção da utilidade

13)

Diagrama de classes: Fazer mais exercícios para a construção do diagrama, tanto individuais como em sala [1]. Ter um conjunto de exemplos de diagramas corretos e diagramas errados, para que possam ser feitas comparações durante a aprendizagem, principalmente no que diz respeito à utilização de especializações nas associações entre as classes.

[1][Essa sugestão esta associada a treinamento, que pode ser tanto no modelo quanto em OO]

TREINAMENTO NO MODELO: exploração de exercícios

Diagrama de seqüência: **Construir modelos em sala**, mais de um, pois este diagrama é um pouco mais complexo que o de classes, e **há uma tendência forte de quem não está acostumado a transformá-lo num fluxograma [2]**. A mesma idéia do “banco de dados” com modelos corretos e errados, para comparações.

[2][Treinamento em modelagem]

TREINAMENTO NO MODELO: exploração de exercícios

Diagrama de estado: **Definir um método que possa explicitar claramente como estabelecer nas descrições dos casos de uso os estados do sistema [3]** (por exemplo, sublinhar substantivos, verbos, etc.). Também a idéia da base de comparação, para que se possa aprender também como não se deve construir o diagrama.

[3][O que o participante sugere é que seja estabelecido um critério para geração do diagrama de estado, nos mesmos moldes do diagrama de classes. Ou seja, um método para, pelo menos iniciar a modelagem desse diagrama. Em linhas mais gerais podemos pensar que a sugestão remete a um conjunto claro e objetivo de instruções para construção dos modelos]

DOCUMENTO DE REQUISITOS: explicitar a associação entre estados e casos de uso

PROCESSO DE MODELAGEM: estabelecer heurísticas para geração dos modelos iniciais a partir dos requisitos

Diagrama de atividades : **Mais exemplos e exercícios em sala [4]**, pois este diagrama, apesar de ter uma idéia intuitiva, nem sempre é uma tarefa simples passá-lo para o papel (ou ferramenta case), **por possuir inúmeras possibilidades de representação e depender muito da interpretação do sistema por parte dos engenheiros de software envolvidos no projeto [5]**. Uma base de comparação com exemplos certos e errados também seria interessante neste caso.

[4][Treinamento no modelo]

TREINAMENTO NO MODELO: exploração de exercícios, uso de exemplos

[5][Esse comentário parece sugerir algo relacionado à necessidade de definir claramente em que situação os modelos devem ser usados e para capturar que informações. Situar a criação do modelo no contexto do desenvolvimento e definir instruções para sua construção]

MODELO: percepção dos aspectos que os modelos capturam, percepção da utilidade dos modelos no contexto de desenvolvimento

PROCESSO DE MODELAGEM: definição do momento em que um modelo poderá ser construído/refinado

Mapa de atores : **O mapa de atores, para este sistema pequeno, relativamente simples e com poucos atores, não apresentou muitas dificuldades. No entanto, num projeto mais complexo e que envolva uma quantidade muito maior de atores, seria preciso mostrar como proceder para capturar as relações existentes entre os atores, de dependência, herança, responsabilidades [6]**, etc. Uma base contendo diversos mapas, em diversos contextos, certos e errados, é uma boa base de comparação.

[6][*Parece sugerir que a quantidade de atores e/ou de relacionamentos entre atores pode ser um critério para decisão sobre a construção desse modelo*]

PROCESSO DE MODELAGEM: critérios para seleção dos modelos

Modelo navegacional: A documentação deste modelo poderia ser revista, pois atualmente está muito confusa, e também muito fragmentada, pois não há um exemplo que envolva em um só modelo todos os componentes que o OOWS propõe que o modelo navegacional possua, mas sim diversos exemplos de modelos pequenos [7], mostrando os componentes conforme vão sendo apresentados. A junção de todos os componentes em um modelo navegacional final para um sistema fica prejudicada por não haver a possibilidade de visualização de um modelo final completo na documentação. Mais aulas e exercícios [8], tanto em sala como individuais e disponibilização de mais material e exemplos a respeito.

[7][*Exemplos que reflitam cenários de trabalho completos e não somente fragmentos destes*]

TREINAMENTO NO MODELO: estruturar exemplos completos que simulem uma situação real

[8][*Treinamento no modelo com uso de exercícios*]

TREINAMENTO NO MODELO: exploração de exercícios

14)

Diagrama de classes: A utilização de uma ferramenta case mais intuitiva e melhor documentada [1].

[1][*Ferramenta CASE fácil de usar e intuitiva, ou seja, que esteja em sintonia com os conceitos usados nos modelos*]

FERRAMENTA CASE: adequação sintática e semântica aos modelos

Modelo navegacional: Melhor documentação dos modelos, principalmente em relação à integração das AIUs no modelo completo [2]. Desenvolvimento de exemplos completos em sala de aula [3].

[2][*Comentário está relacionada à necessidade de um documentação que facilite o entendimento do uso dos componentes do diagrama, tanto do ponto de vista sintático quanto semântico*]

TREINAMENTO NO MODELO: guia de referência

[3][*Treinamento no modelo com exemplos completos*]

TREINAMENTO NO MODELO: exercícios que simulem situações reais de projeto

15)

Diagrama de classes: Talvez possa ser interessante colocar outros exemplos na apostila [1]. Acho que em muitas aplicações Web compartilham de classes bem semelhantes. Isto poderia ter ajudado a clarear melhor as idéias na hora da criação deste diagrama.

[1][*Exploração de exemplos como forma de treinamento*]

TREINAMENTO NO MODELO: uso de exemplos

Diagrama de seqüência: Acho que este diagrama foi bem explicado no conjunto apostila-aula.

Diagrama de estado: A parte mais difícil deste diagrama é identificar o que pode ter estados [2]. Talvez seja o mesmo caso do diagrama de classes. Acho que exemplos podem ajudar bastante.

[2][A dificuldade relatada está associada ao entendimento dos aspectos que o modelo captura]

MODELO: percepção dos aspectos que os modelos capturam

Diagrama de atividades : A dificuldade neste diagrama foi em identificar a quantidade de diagramas que deviam ser feitos [3]. Não consigo pensar em maneiras para melhorar este aspecto a não ser a experiência.

[3][Esta sugestão está relacionada à necessidade de selecionar os modelos realmente relevantes no contexto de um projeto específico]

PROCESSO DE MODELAGEM: critérios para seleção das características mais relevantes

Mapa de atores: Acho que este diagrama foi bem explicado no conjunto apostila-aula.

Modelo navegacional: Acredito que se o material de OOWS tivesse traduzido, teria menos esforço [4]. Mas seu conteúdo mais aula são suficientes.

[4][A sugestão está relacionada ao idioma da documentação. O uso da língua pátria pode diminuir o esforço de aprendizado]

TREINAMENTO NO MODELO: material em língua nativa

16)

Diagrama de classes: Poderíamos ter feito os diagramas de seqüência um pouco mais consistentes com esse modelo [1], o que nos teria proporcionado um melhor entendimento do mesmo.

[1][Sugere algum mecanismo de consistência entre os modelos]

GARANTIA DA QUALIDADE: processo para detectar inconsistências nos modelos

Diagrama de seqüência: Poderíamos ter dividido melhor as tarefas entre os componentes do grupo [2]. Os diagramas de atividade também poderiam ter ajudado na construção desse modelo [3].

[2][Esse comentário sugere que a divisão das tarefas pode influenciar na construção dos modelos. Não fica claro o porquê dessa afirmação]

[3][Esse comentário sugere que a ordem em que os modelos são construídos pode estar diretamente relacionado ao esforço de construção de um determinado modelo]

PROCESSO DE MODELAGEM: definição de uma ordem de construção

Mapa de atores: Não apresentou um grau de dificuldade/esforço suficientemente alto para que algo pudesse ser feito para diminuí-los

Modelo navegacional: Poderíamos ter nos reunido e discutido como fazer esse modelo, tomando como base outros modelos, já que ele ficou um pouco inconsistente devido a ter sido feito paralelamente com os outros modelos [4].

[4][A ordem de construção adotada trouxe dificuldades para geração do modelo]

PROCESSO DE MODELAGEM: definição de uma ordem de construção

17)

Diagrama de classes: Reunião presencial da equipe para sua definição parcial, com o comum acordo de todos [1].

[1][Esse comentário sugere que é importante que os participantes tenham uma visão única dos conceitos envolvidos e como eles se relacionam]

PROCESSO DE MODELAGEM: estabelecer mecanismos que permitam uma visão única dos modelos e das decisões de projeto

Diagrama de seqüência: Ter definido o máximo possível o Diagrama de classes: e o mapa de atores antes de começar sua confecção [2]

[2][Esse comentário sugere uma ordem de construção e afirma que ter a estrutura dos conceitos bem definida facilitou a definição dos comportamentos associados]

PROCESSO DE MODELAGEM: ordem de construção

Diagrama de estado: Ler os requisitos consistentemente para ter um conhecimento sólido do sistema.

Mapa de atores: Experiência mínima na confecção deste mapa [3] e ter a primeira versão do Diagrama de classes: definida antes de sua criação [4].

[3][Sugere que a experiência prévia é importante. Essa experiência pode vir de projetos reais ou de exercícios práticos. Além disso, exemplos também podem ajudar aqui]

TREINAMENTO NO MODELO: exercícios que simulam situações reais de projeto

[4][Sugere uma ordem de construção]

PROCESSO DE MODELAGEM: ordem de construção

Modelo navegacional: Tempo disponível tanto para sua confecção [5] quanto para o entendimento do problema.

[5][Sugere que deve haver “tempo disponível”, o que é bastante subjetivo. Pode ser que o tempo não tenha sido suficiente porque o participante subestimou o esforço para construção do modelo. Por outro lado, a falta de conhecimento pode ter demandado um tempo excessivo para construção desse modelo. Pode ser que a ferramenta não tenha apoiado adequadamente essa tarefa. Não vou associar nenhum coding aqui]

18) NÃO RESPONDEU

19)

Diagrama de classes: **Leitura da descrição do problema utilizando a técnica de retirada de substantivos e verbos visando formar classes, atributos e métodos [1].**

[1][A sugestão reforça a necessidade de haver um conjunto de instruções para construção dos modelos]

PROCESSO DE MODELAGEM: definição de heurísticas para geração dos modelos

Diagrama de seqüência: **Criação dos diagramas de atividades [2]**

[2][Sugere uma ordem para facilitar a criação do modelo]

PROCESSO DE MODELAGEM: ordem de construção

Diagrama de estado: **Análise do documento de requisitos [3]**

[3][Como o documento de requisitos usado era um documento com a descrição dos UCs, podemos inferir que o participante indiretamente sugere que o diagrama de estado pode ser construído (ou pelo menos inicialmente gerado) a partir das descrições dos UCs]

PROCESSO DE MODELAGEM: definição de heurísticas para geração dos modelos

Diagrama de atividades: **Leitura da especificação dos casos de uso e análise do diálogo entre usuário e sistema [4].**

[4][O participante sugere uma ordem para construção dos modelos e que o insumo para esse modelo são os UCs]

PROCESSO DE MODELAGEM: definição de heurísticas para geração dos modelos

Mapa de atores : Análise do diagrama de casos de uso para verificar quais casos de uso que cada ator participa.

Modelo navegacional: Sei que o normal é a ordem inversa, mas no caso do projeto que participei a elaboração de protótipos ajudou bastante no desenvolvimento desse diagrama [5].

[5][Talvez por não saber como construir o modelo ou por não enxergar nenhum desdobramento a partir desse modelo, a construção do protótipo foi realizada. A afirmação acima parece trazer uma indicação importante: a de que existe uma relação direta entre o modelo de navegação proposto e o protótipo do sistema. Fica a pergunta: se o protótipo “ajudou” na geração do modelo, será que o modelo não apoiaria a construção do protótipo? Que outros artefatos poderiam ser gerados?]

GERAÇÃO (SEMI) AUTOMÁTICA: geração de artefatos a partir de modelos

20)

Diagrama de classes: **Um maior trabalho em equipe. Reuniões presenciais para discussão do escopo do projeto [1], além de um maior número de revisões por parte de todos os integrantes da equipe [2].**

[1][*Esse comentário sugere que é importante que os participantes tenham uma visão única dos conceitos envolvidos e como eles se relacionam*]

PROCESSO DE MODELAGEM: estabelecer mecanismos que permitam uma visão única dos modelos e das decisões de projeto

[2][*Revisar o modelo gerado*]

GARANTIA DA QUALIDADE: processo de inspeção para detectar inconsistências nos modelos

Diagrama de seqüência: Mapear o projeto e validá-lo com o cliente, a fim de que se possa realmente acreditar que aquela é a seqüência de ações que devem ser tomadas pelo sistema [3].

[3][*Validar o modelo de seqüência, no contexto do que foi proposto, é semelhante a validar o UC , visto que em um nível alto de abstração, os diagramas de seqüência representam diretamente os UCs. Não faz sentido validar a troca de mensagens em si com o cliente.]*

GARANTIA DA QUALIDADE: processo de inspeção para garantir a qualidade dos requisitos

Diagrama de estado: Uma análise mais profunda por parte do engenheiro de software e do cliente para saber quais atributos do sistema possuem seu estado mutável

Diagrama de atividades: Assim como o diagrama de classes, uma maior compreensão do escopo do projeto [4] e um maior número de revisões. Possuir o mapa navegacional também seria de grande ajuda [5].

[4][*Sugere que o entendimento do escopo do projeto traz facilidade para construção dos modelos. Ou seja, o documento de requisitos tem que, de alguma forma, facilitar o entendimento desse escopo*]

DOCUMENTO DE REQUISITOS: definir e detalhar o escopo do projeto

[5][*Sugere que o mapa navegacional pode apoiar a construção do diagrama de atividades. Se o diagrama de atividades fosse usado para representar algum fluxo de trabalho poderíamos imaginar seu uso na definição do modelo de navegação, mas ao contrário não entendo*]

Mapa de atores: Entender de forma mais objetiva quem vai usar o sistema e quais atribuições cada um deve possuir [6].

[6][*Isso está relacionado ao levantamento de requisitos e não à modelagem em si*]

Modelo navegacional: Reuniões com o cliente para que ele desenhe, juntamente com um engenheiro de software, como ele quer que o caminho do sistema seja desempenhado [7].

[7][*Podemos inferir que a sugestão é que o modelo de navegação seja validado com o cliente*]

GARANTIA DA QUALIDADE: mecanismos para validar o modelo de navegação com o cliente

21)

Diagrama de classes: Todos os integrantes da equipe deveriam estar presencialmente juntos na hora em que fossem desenvolvidos os diagramas [1] de seqüência (na

minha opinião deveria sempre ser feito junto com modelo de classes [2]), abstraindo assim um idéia totalmente uniforme da perspectiva do projeto*.

*Considerando que foi o 1º projeto feito pela equipe.

Para projetos futuros, talvez, a integração se daria mais facilmente sem necessidade tão grande de sincronismo nas atividades.

[1][Novamente aqui surge a idéia de que os membros da equipe devem compartilhar de uma mesma visão dos conceitos e seus relacionamentos]

PROCESSO DE MODELAGEM: estabelecer mecanismos que permitam uma visão única dos modelos e das decisões de projeto

[2][É sugerido também que as visões estrutural e comportamental dos conceitos sejam geradas em paralelo ou, pelos menos, de uma forma interativa com ciclos curtos]

PROCESSO DE MODELAGEM: ordem de construção, tratamento complementar das visões estrutural e comportamental

Diagrama de seqüência: Idem ao modelo de classes.

Diagrama de estado: Adoção de um modelo, definido previamente, para manter padrões de desenvolvimento.

Diagrama de atividades: NA (não construí, apenas revisei). Não observei maiores dificuldades no desenvolvimento deste diagrama. Acho que o modelo de requisitos bem definido ajudou bastante nesta etapa [3].

[3] [A sugestão é clara no sentido de apontar a necessidade de ter os requisitos “bem” definidos como base para construção dos modelos. Entretanto, é preciso especificar o que seriam requisitos “bem” definidos]

DOCUMENTO DE REQUISITOS: nível de detalhamento adequado

GARANTIA DA QUALIDADE: processo para garantia da qualidade dos requisitos

Mapa de atores: Não apresentamos um modelo de mapa de atores oficial, apenas apresentamos o nosso ponto de vista frente a esse modelo. Como foram poucos atores não encontramos muitas dificuldades [4].

[4][Novamente o participante relaciona a quantidade de atores com a facilidade de geração. Essa quantidade e os relacionamentos entre os atores podem ser indicadores da necessidade de construção desses modelos]

Modelo navegacional: NA (não construí, apenas revisei). Mais experiência seria fundamental para a construção de um modelo com mais qualidade [5].

[5][Novamente aparece a questão da experiência prévia. Essa experiência pode ser adquirida em projetos reais ou através de treinamento no modelo que privilegie a prática desses modelos]

TREINAMENTO NO MODELO: exercícios que simulem situações reais de projeto

22)

Diagrama de classes: Definir a necessidade dos métodos a serem representados (construtores, get, set, etc)[1]

[1][Esse comentário sugere que o detalhamento dos conceitos não precisa ser feito de uma forma integral inicialmente. Em algum momento esse detalhamento poderá ser

integral (e deverá ser principalmente se esse modelo for usado como base para geração de código por exemplo), mas o foco inicial deve ser o de definição dos elementos básicos que possibilitem a geração dos demais modelos e a consistência entre eles]

PROCESSO DE MODELAGEM: definição do momento em que determinado modelo deverá ser construído/refinado

Diagrama de atividades: **Uso de ferramenta mais adequada [2], exemplos mais complexos [3]** e compatíveis com a dificuldade do exercício final

[2][Essa sugestão reforça a idéia de que a ferramenta CASE é um fator importante na construção dos modelos , principalmente no que diz respeito à adequação sintática e semântica desta aos modelos construídos]

FERRAMENTA CASE: adequação aos modelos

[3][Novamente é sugerida a necessidade de exemplos que facilitem o entendimento dos modelos]

TREINAMENTO NO MODELO: explorar exemplos completos que simulem situações reais de projeto

Modelo navegacional: Uso de ferramenta mais adequada, exemplos mais complexos e compatíveis com a dificuldade do exercício final

23)

Diagrama de classes: **O ideal é que esse modelo seja construído por todos os membros da equipe em reunião presencial [1] após leitura minuciosa do documento de especificação do projeto e requisitos e esta deve corresponder à primeira etapa do projeto [2].**

[1][É sugerida que a visão conceitual seja compartilhada. Aqui é sugerido que todos os membros construam esse modelo. Em outro momento foram sugeridas reuniões presenciais para atacar esse mesmo problema.]

PROCESSO DE MODELAGEM: estabelecer mecanismos que permitam uma visão única dos modelos e das decisões de projeto

[2][No que diz respeito à ordem de geração dos modelos, é sugerido que esse modelo seja o primeiro, ou seja, estruturar os conceitos primeiro]

PROCESSO DE MODELAGEM: ordem de construção

Diagrama de seqüência: **Os diagramas de seqüência devem ser construídos após a construção do diagrama de classes [3]. Também, para diminuir o esforço, esses diagramas podem ser construídos apenas para os casos de uso mais complicados [4].**

[3][Sugere uma ordem de construção: classes depois seqüência]

PROCESSO DE MODELAGEM: ordem de construção

[4][Sugere um critério para modelagem parcial no caso dos diagramas de seqüência: a complexidade do UC]

PROCESSO DE MODELAGEM: seleção dos modelos mais relevantes

Diagrama de estado: Estes não representam muita dificuldade. Apenas necessitam ser construídos após a etapa de construção do modelo conceitual, de forma a identificar as entidades que apresentam diferentes estados [5].

[5][Sugere uma ordem de construção: classe depois estado]

PROCESSO DE MODELAGEM: ordem de construção

Diagrama de atividades : O diagrama de atividades foi aquele com o qual apresentei maior dificuldade. Acredito que, para diminuir o esforço, deva-se fazer uma análise minuciosa dos casos de uso, de forma a verificar quais processos dentro desses possuem relevância para serem modelados [6] em diagramas de atividades.

[6][Também sugere um critério para modelagem parcial no caso dos diagramas de atividades: a relevância do UC (relevância aqui pode ter várias interpretações: complexidade, importância para o negócio, criticalidade, etc.)]

PROCESSO DE MODELAGEM: seleção dos modelos mais relevantes

Mapa de atores: Uma análise prévia do comportamento de cada ator no sistema ajuda a visualizar melhor um mapa de atores que possa simplificar o diagrama de casos de uso e também o mapa navegacional, propondo uma hierarquia.

Modelo navegacional: Como já mencionado antes, o mapa navegacional pode ser simplificado desde que as relações de hierarquia entre os atores tenham sido bem definidas.

24) NÃO RESPONDEU

APÊNDICE C – Manual do Usuário do UseCaseAgent

Neste apêndice é apresentado o manual do usuário do UseCaseAgent abrangendo a instalação do plug-in, as funcionalidades e restrições da ferramenta e as mensagens de erro oriundas da verificação do diagrama e da especificação do caso de uso.

C.1 Introdução

O UseCaseAgent é um *plug-in*, desenvolvido em Java, que trabalha em conjunto com a ferramenta BOUML. Esse *plug-in* foi desenvolvido para apoiar a criação e verificação sintática das especificações de casos de uso descritas segundo o metamodelo UCMoel. Tais especificações são representadas como um diagrama de atividades e o principal objetivo do UseCaseAgent é permitir que o desenvolvedor trabalhe com a descrição textual do caso de uso deixando a cargo do *plug-in* a tarefa de gerar, automaticamente, o diagrama de atividades correspondente.

Para que seja possível realizar a tradução de uma descrição textual do caso de uso para um diagrama de atividades e vice-versa, algumas condições são necessárias:

- A descrição textual deve seguir um padrão que possa ser “interpretado”. Para atender a essa condição foi criado um metamodelo, chamado UCMoel, que descreve o padrão a ser adotada na especificação de casos de uso. Esse metamodelo descreve os elementos que compõem a descrição de um caso de uso, bem como as regras que definem as combinações válidas entre esses elementos, e;
- Deve haver uma estrutura de pacotes bem definida na qual os diversos elementos que compõem o caso de uso e sua descrição possam ser gerados. Nesse caso, foram acrescentadas funcionalidades específicas para gerar a estrutura de pacotes adequada.

Assim, o UseCaseAgent conta com as seguintes funcionalidades:

- Criação de um novo caso de uso;
- Edição de um caso de uso já existente;

- Verificação sintática de diagramas de atividades que representam a especificação de casos de uso;
- Impressão da especificação de casos de uso no formato RTF, e;
- Geração da estrutura de pacotes, na ferramenta BOUML, que apóia a especificação do caso de uso.

C.2 Por que BOUML ?

A escolha do BOUML como ferramenta base para o desenvolvimento do UseCaseAgent se deve aos seguintes fatores:

- Open source;
- Roda em diversas plataformas (Windows[®], Linux e Mac OS[®]);
- Implementa praticamente toda a definição da UML 2;
- Permite a definição e uso de perfis UML, e;
- Permite a criação de *plug-ins* em C++ ou Java para estender as suas funcionalidades.

C.3 Instalação e Configuração

Para instalação do UseCaseAgent são distribuídos 2 arquivos:

- Plugout.zip
- Template.zip

A configuração do ambiente deve ser feita em 3 passos:

- 1) Instalação do BOUML: faça o download da ferramenta em <http://bouml.free.fr>
- 2) Descompactação do *plug-in*: o arquivo plugout.zip deve ser descompactado no mesmo diretório onde foi instalado o BOUML.
- 3) Descompactação do projeto *Template.zip*: descompacte o arquivo template.zip em um diretório qualquer.

Depois desses passos, para cada projeto que desejamos utilizar o UseCaseAgent devemos configurar o acesso do BOUML ao *plug-in* e aos perfis UML necessários. Infelizmente essas duas tarefas devem ser feita para cada projeto, porque o BOUML não leva essa configuração de um projeto para o outro. Visando facilitar a utilização do UseCaseAgent é distribuído, juntamente com o *plug-in*, um projeto BOUML chamado *Template*, que já possui todas as configurações necessárias.

Dessa forma, para criar um novo projeto basta abrir o projeto *Template* e usar a opção **Save as** para criar efetivamente o projeto desejado.

Importante: como o UseCaseAgent foi desenvolvido em Java, para que a sua execução seja bem sucedida, o JRE versão 1.6 ou superior deve estar instalado e disponível a partir de um diretório qualquer. Para verificar essa condição basta abrir uma janela do console e digitar “java –version”, conforme Figura A-1.

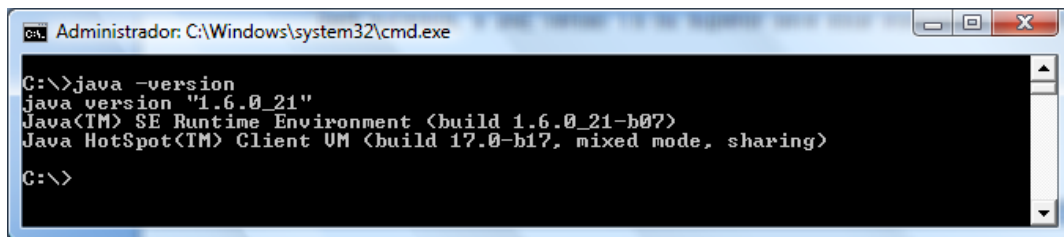


Figura A-1 – Janela do console do sistema para verificação da versão do JRE

C.4 Criação dos Atores

O UseCaseAgent não cria os atores que serão associados aos caso de uso. Para essa tarefa deve ser usado o próprio BOUML. Os atores devem ser criados, obrigatoriamente, na pasta <<actors>>, pois é nessa pasta que o *plug-in* irá buscar os atores candidatos (Figura A-2). Caso você acione o UseCaseAgent e se lembre que esqueceu de criar o ator do caso de uso, você deverá sair do UseCaseAgent, criar o ator e depois acionar o UseCaseAgent novamente.

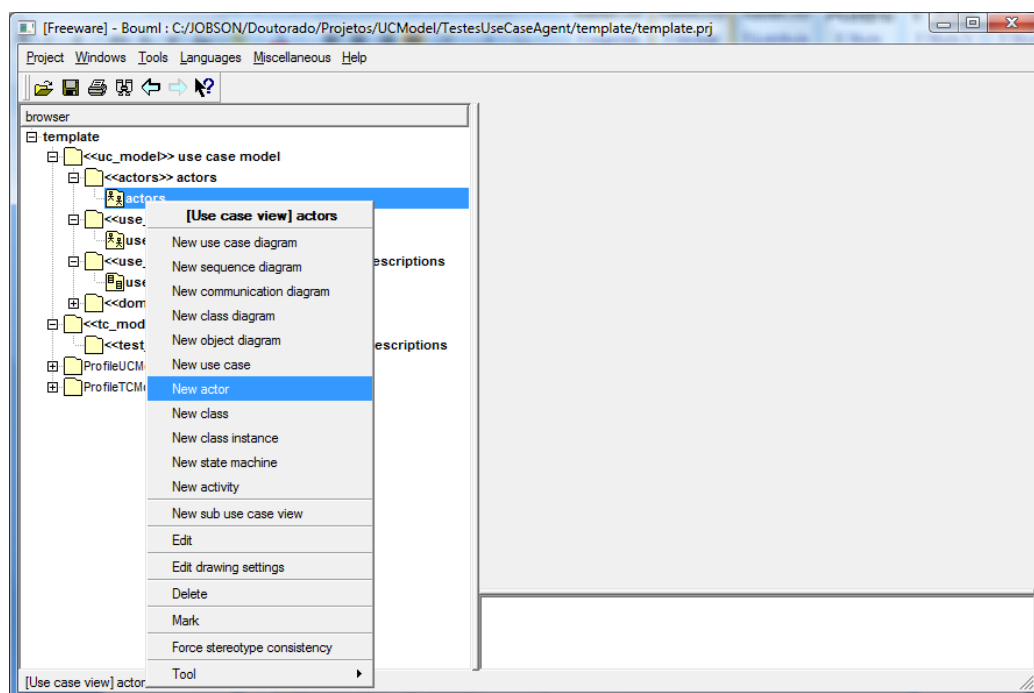


Figura A-2 – Criação de atores na ferramenta BOUML

C.5 Criação de um Caso de Uso

Após a criação do projeto (com base no projeto *Template*, conforme seção 3), basta acionar o menu de contexto no pacote `<<uc_model>>` e selecionar **Tool -> Create Use Case** (Figura A-3).

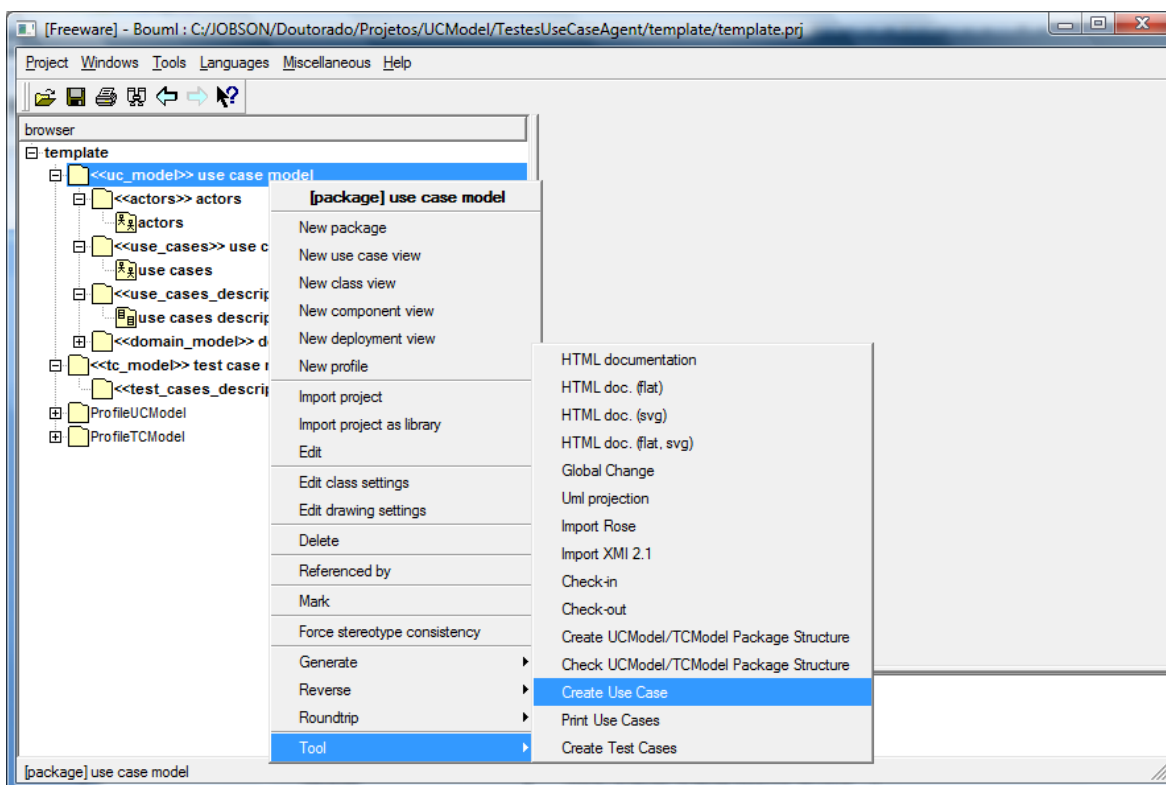


Figura A-3 – Ativação do UseCaseAgent a partir do BOUML para criação de um novo caso de uso

Em seguida será aberta a janela de console do BOUML e a janela do UseCaseAgent com a tela para definição dos dados básicos do caso de uso (Figura A-4)

Importante: a janela do console do BOUML só é usada caso não seja possível iniciar o *plug-in* ou ocorra algum erro fatal durante a execução do BOUML.

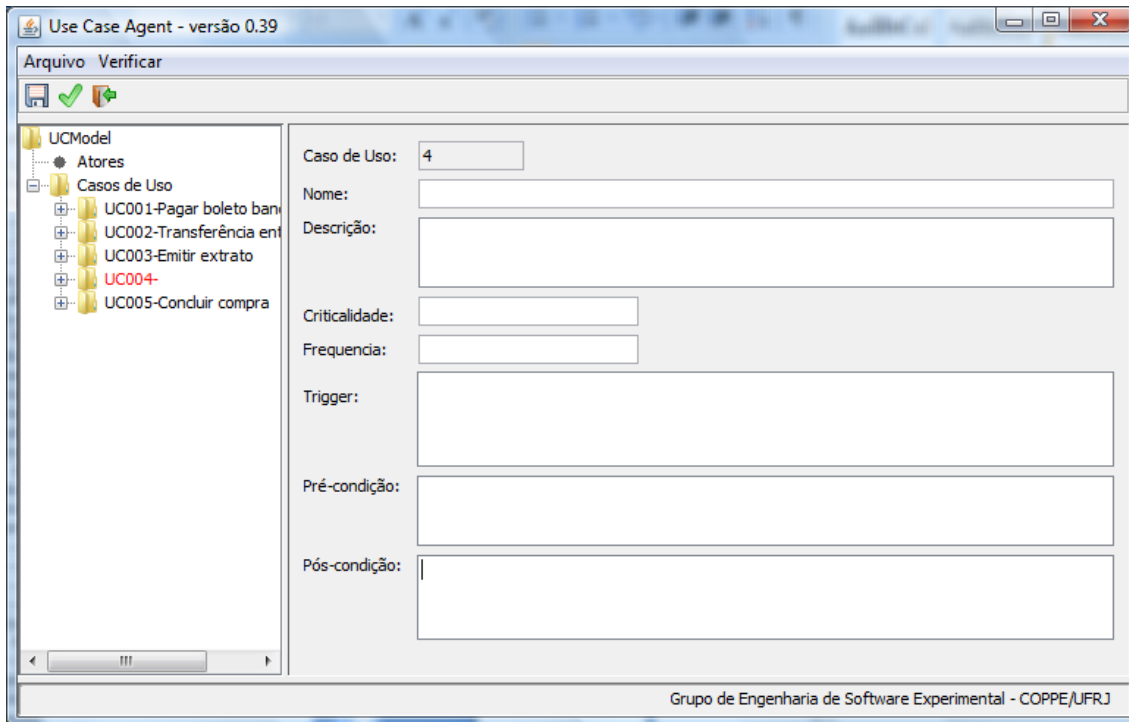


Figura A-4 – Tela inicial do UseCaseAgent

Nessa tela é possível observar 3 opções na *toolbar*:

- **Salvar:** verifica a sintaxe da descrição e gera o diagrama de atividades correspondente ao caso de uso, caso não haja nenhum erro de sintaxe no modelo.
- **Verificar:** verifica a sintaxe do modelo.
- **Sair:** sai do UseCaseAgent sem salvar a descrição do caso de uso.

No menu existe uma quarta opção:

- **Salvar como novo:** verifica a sintaxe da descrição e cria um novo caso de uso, caso não haja nenhum erro. Essa opção só está disponível se o caso de uso estiver sendo editado.

Nessa tela também temos a árvore, no painel à esquerda, que é usada para navegar entre os elementos do caso de uso. Ao selecionar os elementos dessa árvore são apresentadas telas específicas no painel à direita da tela:

- **Atores:** mostra a lista de atores cadastrados e os casos de uso que os referenciam
- **Lista de Casos de Uso:** lista com todos os casos de uso existentes. O caso de uso destacado em vermelho é aquele que está sendo

criado/editado. Os demais casos de uso não podem ser editados, mas podem ser referenciados nos *includes* ou *extends*.

- **Atores do Caso de Uso:** mostra a lista com os atores existentes e permite selecionar quais atores estão associados ao caso de uso corrente.
- **Fluxo principal:** mostra a lista de ações do fluxo principal, permitindo a sua edição.
- **Fluxos alternativos:** lista com os fluxos alternativos definidos para o caso de uso corrente. Para editar um fluxo alternativo basta clicar sobre ele.
- **Fluxos de exceção:** lista com os fluxos de exceção definidos para o caso de uso corrente. Para editar um fluxo de exceção basta clicar sobre ele.
- **Regras:** mostra a lista com as regras associadas ao caso de uso, permitindo a sua edição.
- **Erros:** mostra a lista de erros após acionar a verificação sintática do caso de uso.

C.6 Dados Gerais do Caso de Uso

Ao clicar no caso de uso destacado em vermelho (caso de uso que está sendo criado/editado) é apresentada a tela da Figura A-5.

Nessa tela devem ser informados os dados básicos do caso de uso. Repare que, na criação, o número do caso de uso é gerado automaticamente pelo *plug-in*.

The screenshot shows the 'Use Case Agent - versão 0.39' window. On the left, a tree view shows a project structure with 'UCModel' containing 'Atores' and 'Casos de Uso'. Under 'Casos de Uso', five items are listed: 'UC001-Pagar boleto bancário', 'UC002-Transferência em dinheiro', 'UC003-Emitir extrato', 'UC004-Autenticar usuário' (highlighted in red), and 'UC005-Concluir compra'. The main area on the right is a form for editing the selected case use. The fields are: 'Caso de Uso:' with the value '4'; 'Nome:' with the text 'Autenticar usuário'; 'Descrição:' with the text 'Esse caso de uso permite que o usuário faça a sua autenticação no sistema'; 'Criticalidade:' with the value 'Alta'; 'Frequência:' with the value 'Alta'; 'Trigger:' with the text 'O usuário acessa a tela inicial do sistema'; 'Pré-condição:' with the text 'O usuário não encontra-se autenticado'; and 'Pós-condição:' with the text 'O menu principal da aplicação é apresentado de acordo com o perfil do usuário'. The bottom of the window displays the text 'Grupo de Engenharia de Software Experimental - COPPE/UFRJ'.

Figura A-5 – Tela do UseCaseAgent para definição dos dados básicos do caso de uso

C.7 Atores do Caso de Uso

Ao clicar na opção **Atores do Caso de Uso** é apresentada a tela da Figura A-6:

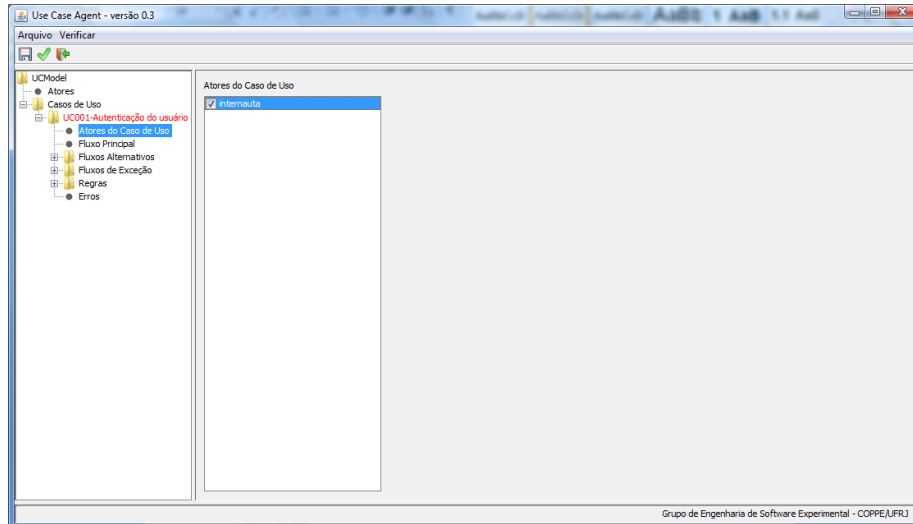


Figura A-6 - Tela do UseCaseAgent para associação dos atores com o caso de uso

Nessa tela são listados todos os atores cadastrados na pasta <<actors>>. Para associar ou dissociar um ou mais atores ao caso de uso basta marcá-los ou desmarcá-los nessa lista.

C.8 Fluxo Principal

Ao clicar em **Fluxo Principal** é apresentada a tela da Figura A-7 para edição dos passos do caso de uso:

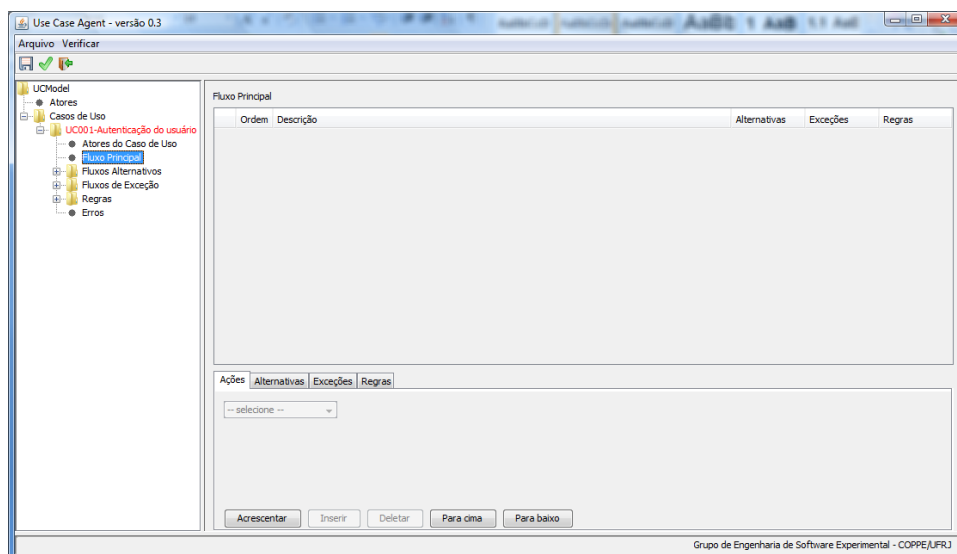


Figura A-7 – Tela do UseCaseAgent para criação das ações do fluxo principal

Nessa tela é possível:

- Acrescentar uma ação ao final do fluxo;
- Inserir uma ação na posição selecionada;
- Deletar uma ação;
- Mover a ação selecionada para cima ou para baixo;
- Incluir ou excluir alternativas e exceções, e;
- Associar regras a um determinado passo.

Cada ação pode ser de um dos seguintes tipos:

- 1) **Ação do ator:** representa uma ação do ator, que normalmente fornece ao sistema alguma informação solicitada.
- 2) **Ação do Sistema:** representa uma ação do sistema que não tem um resultado observável pelo ator. É normalmente usada para explicitar o tratamento da informação fornecida pelo ator.
- 3) **Resposta do sistema:** representa uma ação do sistema que tem um resultado observável pelo ator, chamada de resposta do sistema. Essa resposta pode usar uma interface HC, um email, um sinal que é enviado a um sensor ou qualquer outro meio através do qual o sistema possa se comunicar com o mundo externo.
- 4) **Vai para:** representa um desvio na execução e só pode ser criada para fluxos alternativos ou de exceção. O destino de uma ação “vai para” pode ser uma ação no mesmo fluxo ou qualquer ação nos fluxos da sua cadeia de ativação.
- 5) **Encerrar:** encerra o caso de uso.
- 6) **Include:** representa a inclusão de outro caso de uso.
- 7) **Extend:** representa a extensão de outro caso de uso.

Para as ações do tipo 1, 2, 3, 6 e 7 é possível definir:

- Uma descrição para a ação
- Um complemento para a descrição

Exemplo:

Ação	Resposta do sistema
Descrição	Apresenta o formulário de cadastramento
Complemento:	O formulário deve possuir os seguintes campos: - nome - endereço - email

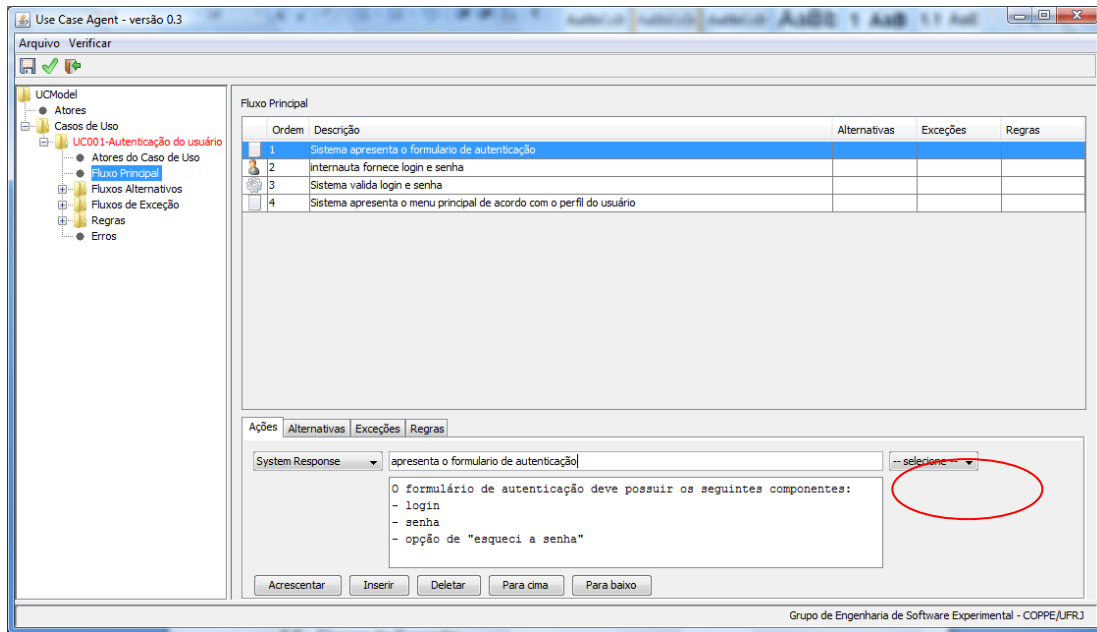


Figura A-8 – Tela do UseCaseAgent mostrando a associação de uma ação com um sub-diagrama de atividades

A tela apresentada na Figura A-8 destacada em vermelho a opção usada para criar/associar um sub-diagrama de atividades à uma ação. O objetivo é especificar o comportamento da ação ou resposta do sistema usando um outro diagrama de atividades. Repare que o UseCaseAgent não apóia a criação desses sub-diagramas. Essa opção permite apenas criar um link entre uma ação/resposta do sistema e um diagrama de atividades que especifica detalhadamente o comportamento do sistema para essa ação específica. Esses sub-diagramas devem ser criados na pasta <<behavior_view>>. Caso o sub-diagrama ainda não exista, o desenvolvedor pode selecionar a opção NOVO.

C.9 Fluxos Alternativos

Para criar um fluxo alternativo, basta selecionar a ação que será substituída pela alternativa e, na aba **Alternativas**, clicar em **Novo** ou selecionar, na árvore à esquerda, uma alternativa previamente criada. O UseCaseAgent sempre cria um novo fluxo alternativo (A1, A2, A3 e assim por diante) com uma condição indefinida (representada por “???”). Para editar esse fluxo, basta clicar sobre ele na árvore à esquerda. As Figuras A-9 e A-10 mostram esse procedimento.

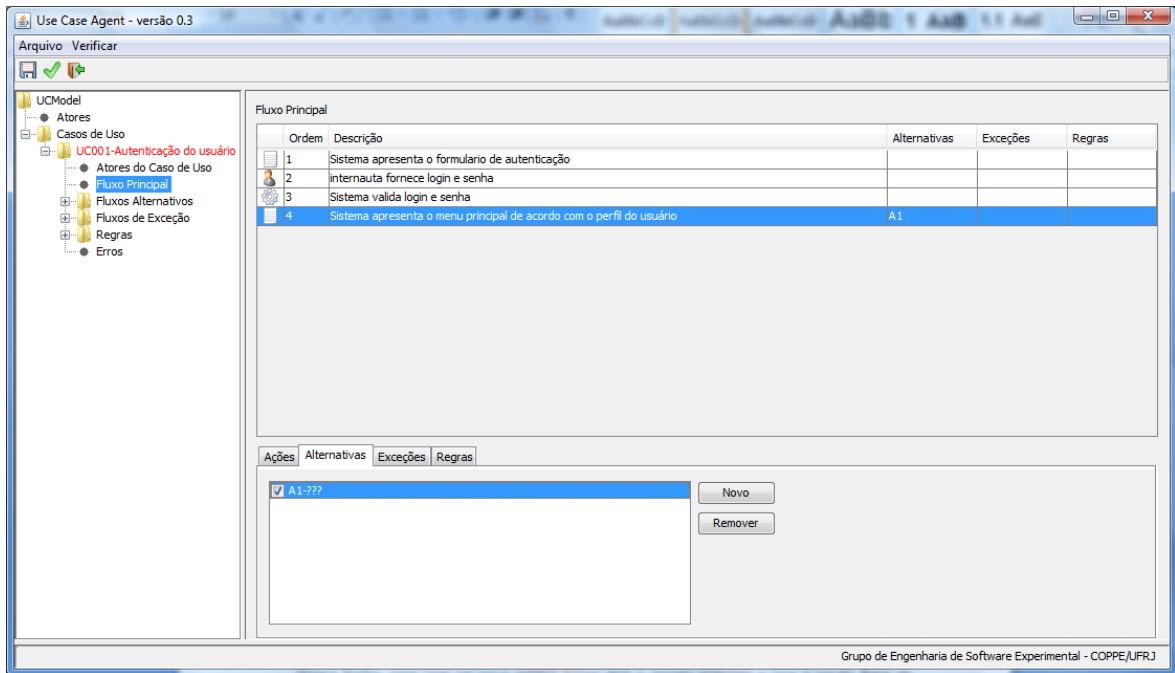


Figura A-9 – Tela do UseCaseAgent para criação de fluxo alternativo associado a uma ação

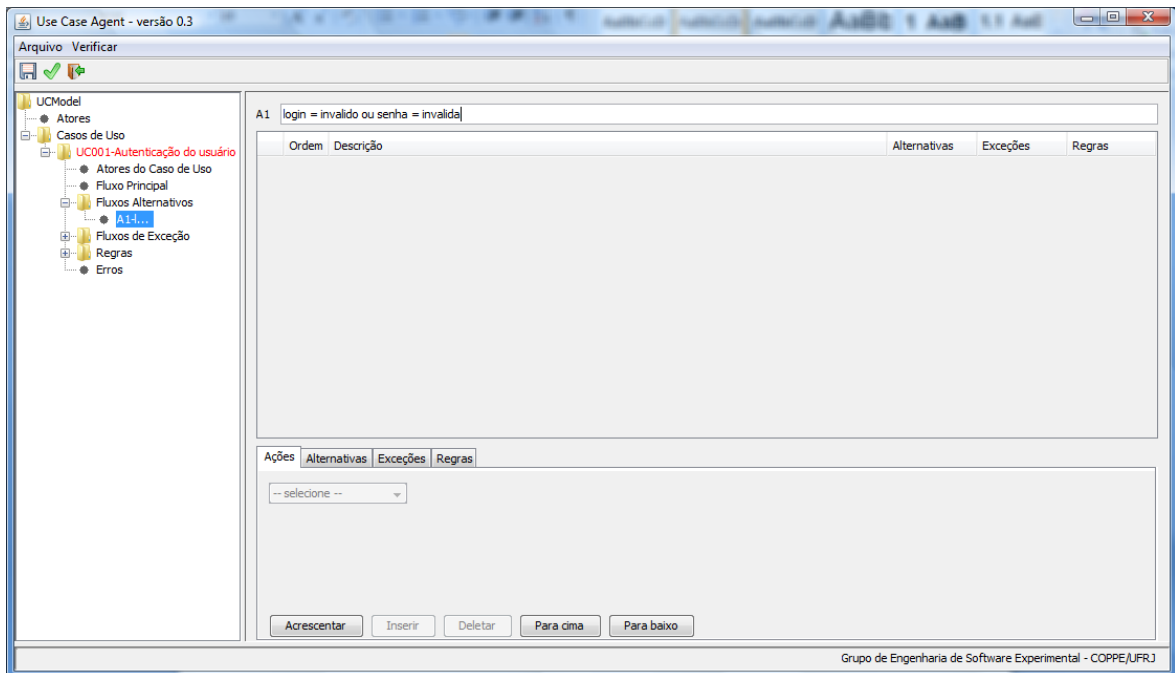


Figura A-10 – Tela do UseCaseAgent para edição das ações do fluxo alternativo

A criação das ações para os fluxos alternativos segue o mesmo padrão definido para o fluxo principal. A única diferença é que, para fluxos alternativos, existe sempre uma condição que define se esse fluxo será ou não executado nos pontos nos quais é referenciado.

C.10 Fluxos de Exceção

A criação de fluxos de exceção e suas respectivas ações ocorre de forma idêntica a dos fluxos alternativos, apenas usando a aba **Exceções** (Figura A-9).

C.11 Regras

Ao clicar na opção **Regras** na árvore à esquerda é mostrada a lista de regras associadas ao caso de uso (Figura A-11). Atualmente, não existe nenhum apoio para compartilhamento de regras entre casos de uso.

Cada regra criada deve ser categorizada como:

- 1) **Regra Conceitual:** define uma restrição estrutural ou comportamento a ser observado pelo sistema em termos dos conceitos do domínio.
- 2) **Regras de Apresentação:** define restrições relacionadas à interface entre o ator e o sistema (seja ela uma interface gráfica ou de qualquer outra natureza).
- 3) **Regras de Navegação:** define restrições no acesso às informações ou serviços disponibilizados pelo sistema.

Após a criação das regras é possível associá-las aos passos do caso de uso usando a aba **Regras** durante a edição dos fluxos principal, alternativo ou de exceção.

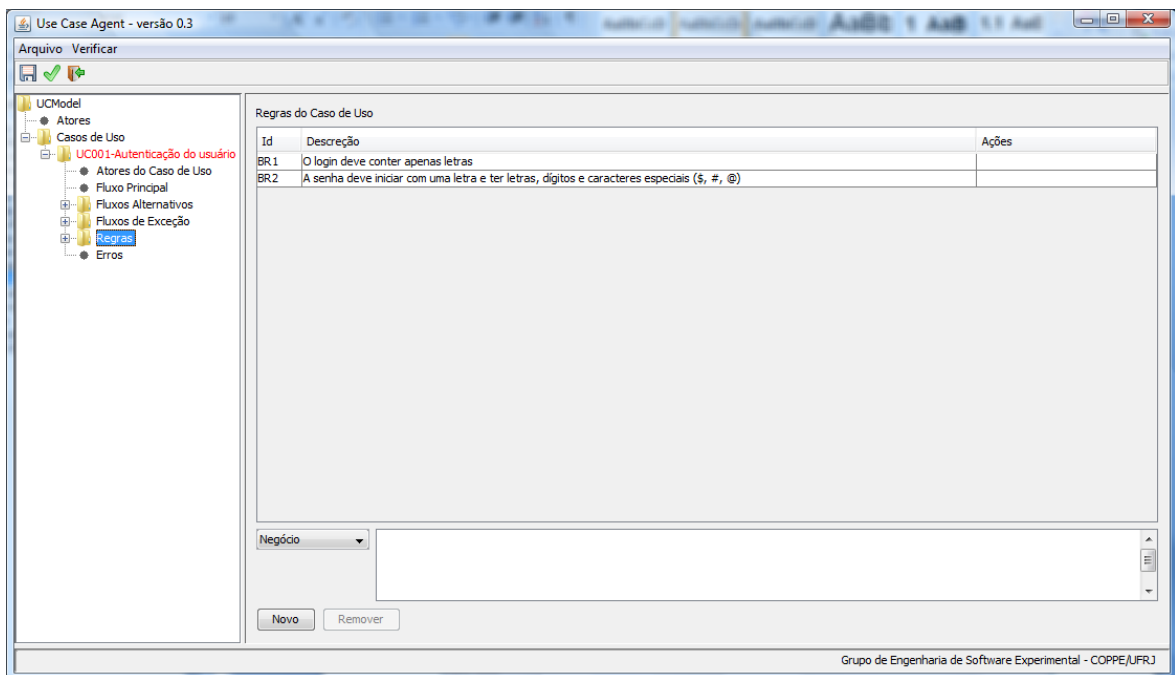


Figura A-11 – Tela do UseCaseAgent para criação de regras

Atualmente a categorização das regras em regras conceitual, regras de apresentação e regras de navegação não tem nenhum desdobramento ou crítica em termos de descrição do caso de uso. Entretanto, essa classificação pode auxiliar em fases posteriores do desenvolvimento.

C.12 Verificação Sintática

A qualquer instante o desenvolvedor poderá acionar a opção **Verificar** para checar a sintaxe da especificação do caso de uso. Caso algum erro seja detectado, o mesmo poderá ser visualizado acionando a opção **Erros** da árvore à esquerda (Figura A-12). Para cada erro o UseCaseAgent apresenta a sua localização a fim de facilitar a correção. É importante ressaltar que a verificação é automaticamente chamada quando a opção **Salvar** é acionada e que, caso algum erro seja detectado, a operação de salvar será cancelada.

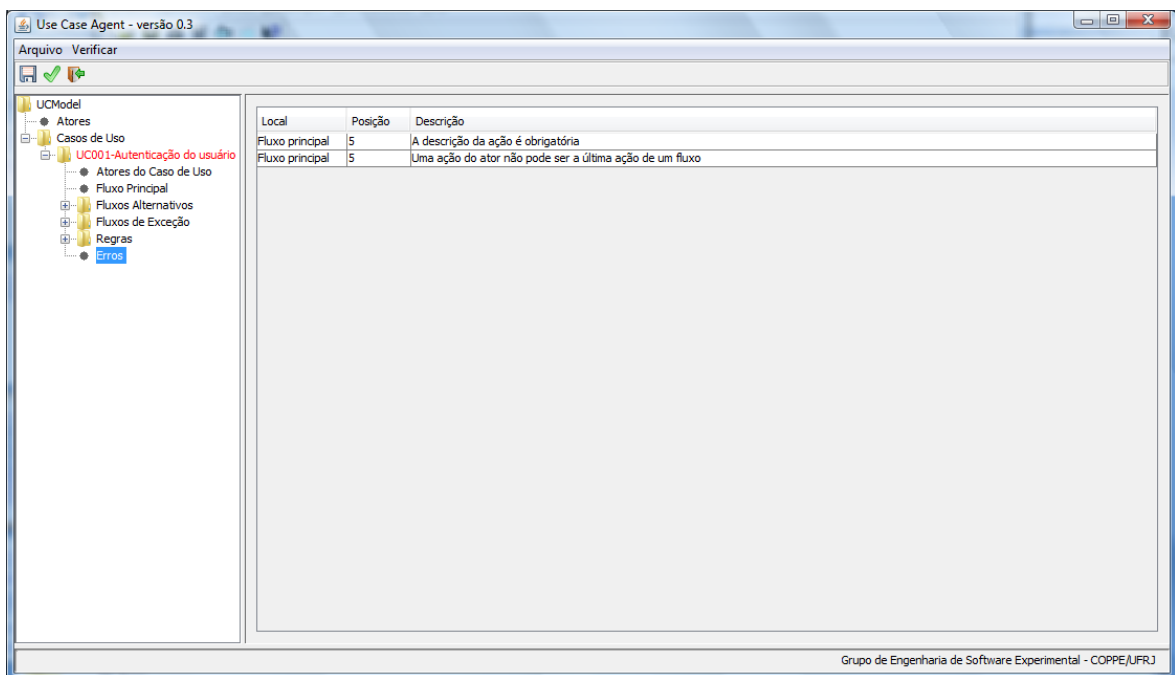


Figura A-12 – Tela do UseCaseAgent com a lista de erros sintáticos e suas localizações

C.13 Edição de um Caso de Uso

Para editar um caso de uso no BOUML basta acionar o menu de contexto sobre a atividade que especifica o caso de uso e acionar a opção **Tool -> Edit Use Case**, conforme a Figura A-13.

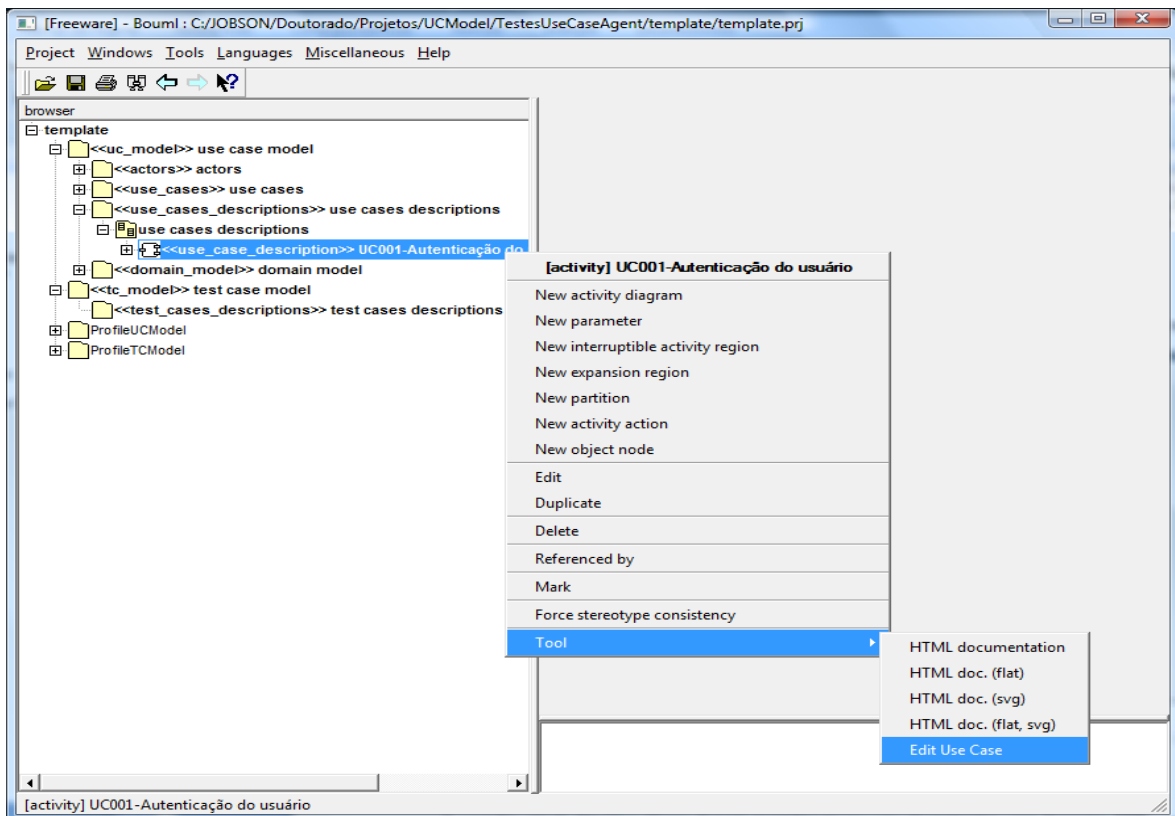


Figura A-13 – Ativação do UseCaseAgent a partir do BOUML para edição de um caso de uso

As mesmas funcionalidades usadas na criação de um novo caso de uso se aplicam à sua edição.

C.14 Impressão dos Casos de Uso

A versão atual do UseCaseAgent permite a geração de um documento, em formato RTF, com a descrição textual dos casos de uso existentes. Para tal, basta acionar, no BOUML, o menu de contexto no pacote `<<uc_model>>` e selecionar **Tool - > Print Use Cases**, conforme a Figura A-14

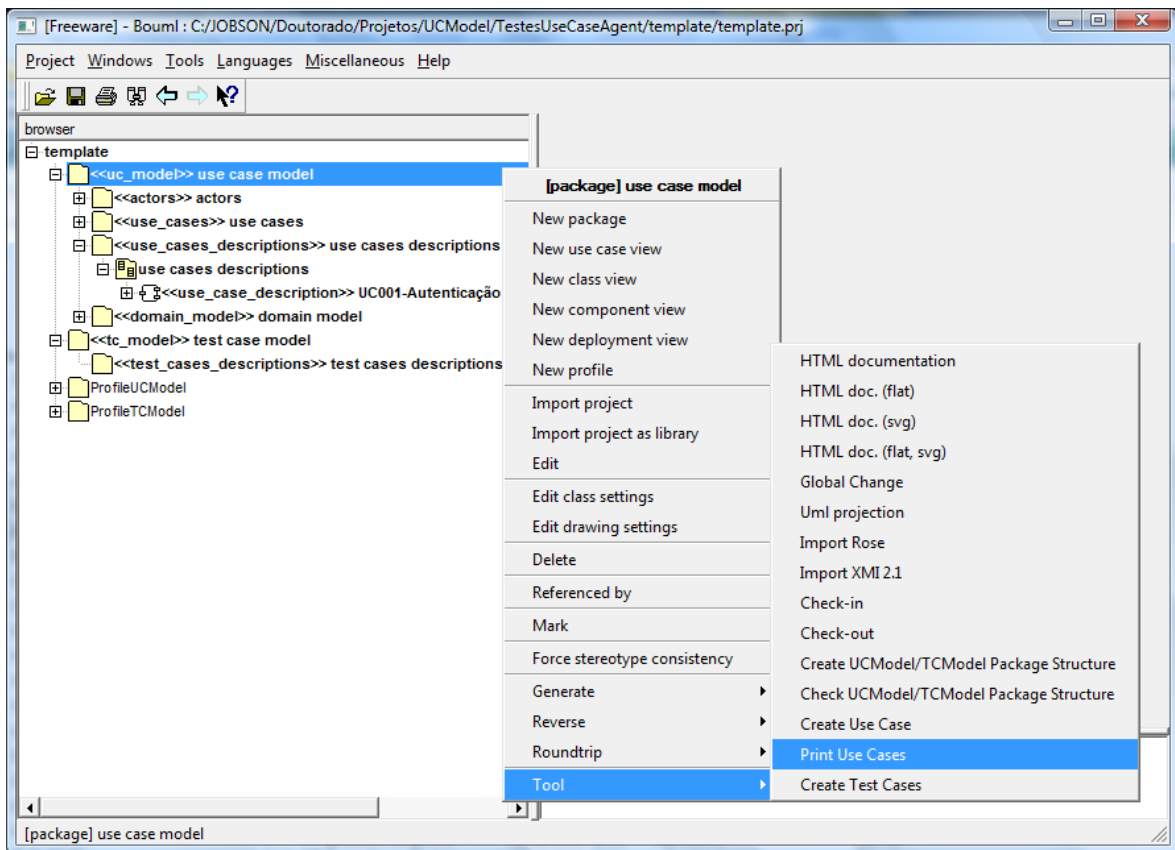


Figura A-14 – Ativação do UseCaseAgent a partir do BOUML para impressão dos casos de uso

Na tela apresentada na Figura A-15 basta selecionar os casos de uso a serem impressos e acionar a opção **Verificar**. Os casos de uso cujas especificações estiverem sintaticamente corretas poderão ser impressos no arquivo RTF desejado (Figura A-16).

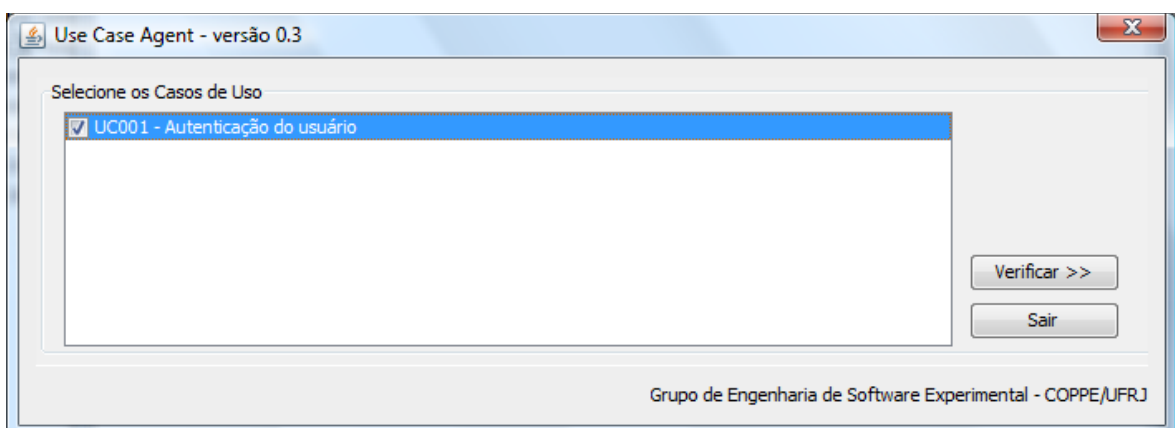


Figura A-15 – Tela do UseCaseAgent para seleção dos casos de uso a serem impressos

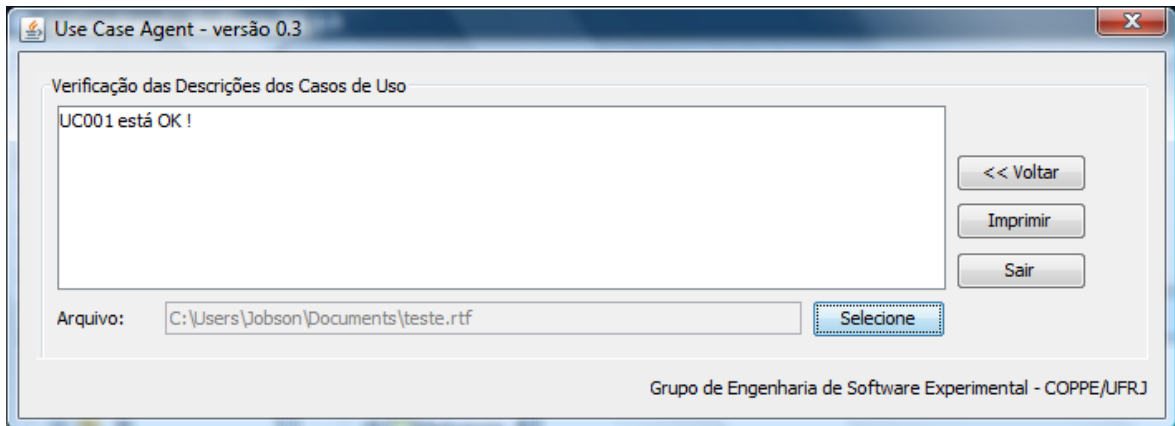


Figura A-16 – Tela do UseCaseAgent para seleção do documento onde os casos de uso sintaticamente corretos serão serem impressos

C.15 Verificação Sintática do Diagrama de Atividades

O UseCaseAgent não impossibilita que o desenvolvedor crie/altere a especificação do caso de uso editando diretamente o diagrama de atividades. Entretanto, essa opção deve ser usada com cuidado, pois, caso o diagrama criado/editado não possua os requisitos mínimos para ser convertido para o metamodelo UCMModel a sua edição via UseCaseAgent não será possível.

Para verificar se um diagrama alterado no BOUML está ou não aderente ao metamodelo UCMModel basta acionar o menu de contexto sobre a atividade que especifica o caso de uso e acionar a opção **Tool -> Check Use Case**, conforme a Figura A-17. Caso alguma incompatibilidade seja detectada, a mensagem de erro correspondente será apresentada no console do BOUML. Nesse caso, o diagrama deve ser corrigido pelo desenvolvedor usando o editor do BOUML.

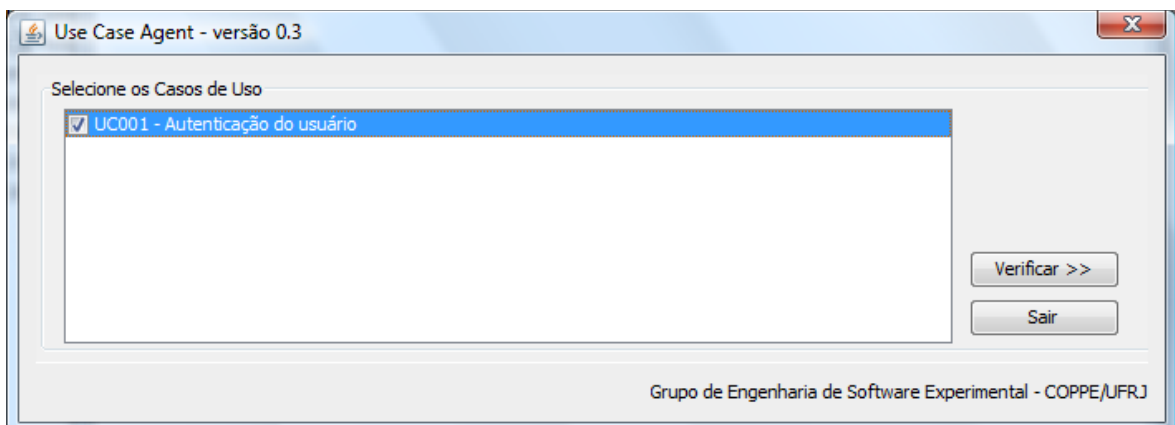


Figura A-17 – Tela do BOUML

C.16 Restrições do UseCaseAgent

Ao editar um caso de uso, o UseCaseAgent sempre exclui a sua especificação anterior (diagrama de atividades) e gera uma nova especificação. Portanto, se foram acrescentados elementos ao diagrama que não fazem parte do metamodelo (por exemplo, comentários associados às ações), estes serão perdidos.

Na versão atual, o UseCaseAgent não permite a edição de informações que fluem entre as ações do caso de uso (fluxo de dados). Assim, se forem criados fluxos de dados entre as ações do caso de uso, esses não serão visíveis no UseCaseAgent, mas serão preservados se o caso de uso for editado.

Como o BOUML não possui nenhuma API para geração automática de diagramas, o UseCaseAgent sempre cria um diagrama em branco. Para popular o diagrama com os elementos desejados basta arrastá-los a partir do *project browser* (painel à esquerda) do BOUML para dentro do diagrama. Devido a essa restrição é recomendado que o desenvolvedor não perca tempo formatando o diagrama enquanto o caso de uso não estiver concluído, pois, caso seja necessário editar o caso de uso, o diagrama criado anteriormente será perdido quando a nova especificação for salva. Dica: para alterar as configurações de desenho do diagrama ou de qualquer um de seus elementos basta acionar, no BOUML, o menu de contexto sobre o diagrama ou no elemento desejado e clicar na opção **Edit drawing settings**.

C.17 Combinação de Ações na Especificação do Caso de Uso

A seqüência básica de ações de um caso de uso, preconizada pelo metamodelo UCMoel, segue os conceitos definidos em Jacobson (1992), conforme a Figura A-18.

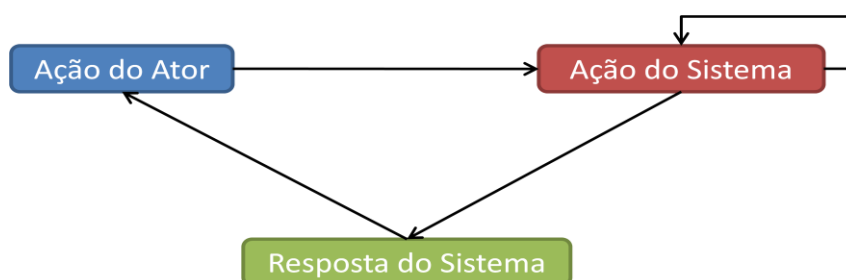


Figura A-18 – Seqüência básica de ações previstas no metamodelo UCMoel

De acordo com a Figura A-18 temos:

Uma **ação do ator**,

é seguida de uma ou mais **ações do sistema**, que são seguidas de **respostas do sistema** que retornam à uma outra ação do ator reiniciando o ciclo.

- **Ação do ator:** representa uma interação do ator com o sistema na qual este faz uma requisição ao sistema informando os dados necessários.
- **Ação do Sistema:** Representa uma ação do sistema cujos resultados gerados não são diretamente observados pelo ator. É usada para explicitar o tratamento dado à requisição do ator. Esta ação está, normalmente, associada a recuperação de informações, alteração do estado interno do sistema e geração de resultados que serão posteriormente apresentados.
- **Resposta do sistema:** Representa uma ação do sistema cujos resultados são direta ou indiretamente observados pelo ator. Essa ação pode apresentar resultados gerados anteriormente ou solicitar outros dados ao ator. Essa resposta pode usar uma interface HC, um email, um sinal que é enviado a um sensor, ou outros meios.

A descrição do Caso de Uso pode ser iniciada com qualquer uma dessas três ações e o seu término deve ser, obrigatoriamente, com uma ação ou resposta do sistema.

De acordo com o tipo da ação que leva a um ponto de decisão, esta pode ser interpretada de diversas formas:

1. **Uma decisão após uma ação do ator:** é interpretada como uma decisão sobre as informações fornecidas pelo ator, ou seja, o ator pode realizar uma escolha explícita que leva a uma ou outra ação do sistema ou a natureza da informação fornecida pode levar a uma ou outra ação do sistema. O estado SOS sistema também pode influenciar nessa decisão.

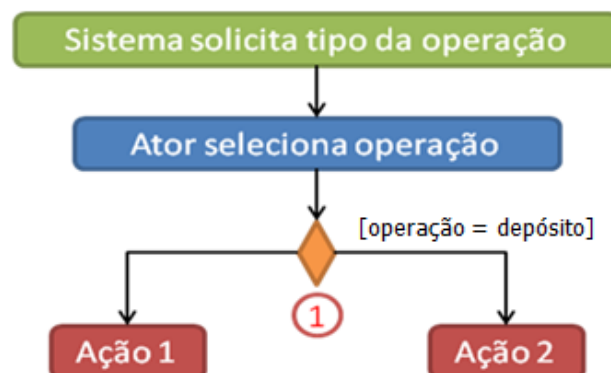


Figura A-19 – Interpretação de uma decisão após uma ação do ator

2. **Uma decisão após uma resposta do sistema:** como a resposta do sistema apresenta o resultado de um processamento e/ou a solicitação de uma nova informação, a decisão é interpretada como uma atitude do ator diferente daquela esperada/solicitada pelo sistema. Nesse caso, a condição define a ação tomada pelo ator.

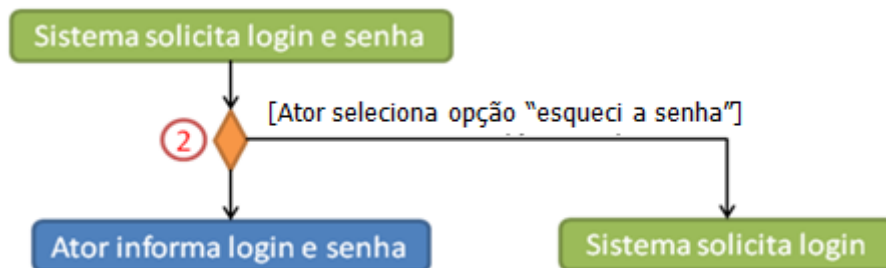


Figura A-20 – Interpretação de uma decisão após uma resposta do sistema

3. **Uma decisão após uma ação do sistema:** representa alternativas de resposta do sistema ou de processamento em função do resultado do processamento de uma informação. Nesse caso, a condição normalmente verifica o resultado do processamento realizado ou o estado do sistema.

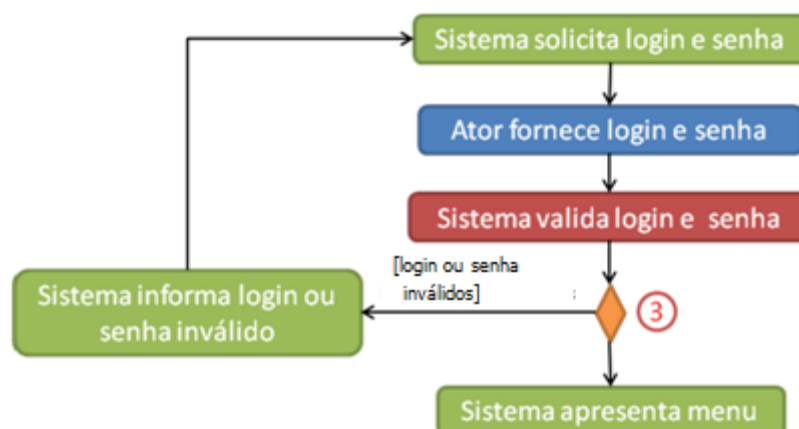


Figura A-21 – Interpretação de uma decisão após uma ação do sistema

C.18 Orientações para Descrição de Casos de Uso

Um Caso de Uso deve:

- Atingir um único, significativo e bem definido resultado de interesse de um ator;

- Ser escrito no vocabulário do domínio;
- Modelar um ou mais requisitos funcionais do sistema;
- Fornecer uma visão externa do sistema – importante para os *stakeholders* validarem;
- Descrever **o quê** o sistema faz, mas não detalha ou especifica **como** é feito, e;
- Ser livre de tecnologia.

Segundo Jacobson [Jacobson 92] um caso de uso é uma seqüência de transações realizada por um sistema para produzir um resultado de valor observável para um determinado ator ou conjunto de atores.

Uma transação consiste em um conjunto de ações realizado por um sistema e é disparada por um ator ou por um evento qualquer dentro do sistema.

De acordo com Jacobson, uma transação consiste em 4 passos:

- 1) Um ator primário envia uma solicitação e os dados (se existirem) para o sistema.
- 2) O sistema valida a solicitação e os dados, se pertinente.
- 3) O sistema altera seu estado interno como resultado do processamento da solicitação.
- 4) O sistema retorna o resultado ao ator.

Para aprimorarmos as descrições alguns conceitos são importantes:

- **Fluxo Principal** – descreve a seqüência de ações que serão executadas considerando que nada anormal acontecerá durante essa execução, ou seja, descreve o que normalmente acontece quando o caso de uso é realizado na situação ideal ou mais comum.
- **Fluxos Alternativos** – descrevem o que acontece quando o ator faz uma escolha alternativa ou quando o resultado de um processamento demanda alguma tomada de decisão.
- **Fluxos de Exceção** – correspondem à descrição das situações de exceção, ou seja, descrevem o que acontece quando algo inesperado ocorre na interação entre o ator e o sistema.

- **Trigger** – evento ou condição que dispara a execução do caso de uso
- **Pré-condição** – define que hipóteses são assumidas como verdadeiras para que o caso de uso tenha início. Deve ser usada em casos de uso cuja realização não faz sentido em qualquer momento, mas somente quando o sistema está em um determinado estado ou com certas propriedades.
- **Pós-condição** – estado que o sistema alcança após o caso de uso ter sido realizado com sucesso.
- **Regras** - são políticas, condições ou restrições que devem ser consideradas na execução dos processos existentes em uma organização ou em um sistema em particular.

Dicas de Estilo

- Casos de uso devem ter nomes que indiquem o objetivo do ator, de preferência no formato “verbo no infinitivo + substantivo”;
- Descreva o caso de uso do ponto de vista do ator primário;
- Use sempre termos relacionados ao domínio do problema, e;
- Não use termos que direcionem para uma solução ou tecnologia específica;
- Use links para outros documentos ou referências para outras seções do documento (fórmulas de cálculos, layouts de relatórios, telas, arquivos de interface com outros sistemas dentre outros) ao invés de descrever toda a complexidade da estrutura ou do comportamento dentro do próprio caso de uso;
- Use diagramas de atividades para explicitar comportamento complexos ao invés de defini-los usando linguagem natural;
- Na descrição das ações:
 - Indique sempre quem faz o quê;
 - Inicie sempre indicando quem realiza a ação: “Sistema...” ou “Ator...”;
 - Use sempre verbos no presente e na voz ativa;
 - Não use comandos de controle de fluxo do tipo “**se..então**” ou “**enquanto..faça**”. Refatore esses desvios como fluxos alternativos ou de exceção, e;

- Não junte diversas ações do sistema em um único passo. Exemplo:
- 1. O Sistema calcula o valor da compra e o frete e os apresenta ao cliente

deve ser descrito como:

1. O Sistema calcula o valor da compra e o frete
2. O Sistema apresenta o valor da compra e o frete ao cliente

Por quê ?

- Regras diferentes podem estar associadas a uma ou outra ação.
- Alternativas ou exceções diferentes podem estar associados a uma ou outra ação
- Dados complementares das ações podem ser diferentes.
- A natureza das duas ações pode ser diferente. Nesse exemplo, uma representa um comportamento interno do sistema e outra representa uma interface HC com o ator.

C.19 Mensagens de Erro

A.19.1 Mensagens de Erro na Verificação de Diagramas

Essa verificação é realizada quando o mesmo é editado ou transformado e tem o objetivo de garantir que o diagrama possui os requisitos mínimos para que possa ser traduzido para o metamodelo UCMModel. Caso algum desses erros seja encontrado a edição/transformação do modelo é abortada.

- **ID do Caso de Uso é nulo ou não é um inteiro positivo. Verifique a tagged-value ID da descrição do Caso de Uso.**

As atividades estereotipadas com *<<use case description>>* possuem uma *tagged-value* chamada ID. Esse ID identifica o caso de uso de forma única dentre os demais e é gerado automaticamente pelo UseCaseAgent. Caso algum caso de uso seja criado sem a ajuda do UseCaseAgent, o desenvolvedor é responsável por fornecer esse ID e garantir que ele é único.

- **Nó inicial não encontrado**

Um diagrama de descrição de UC deve ter um, e somente um, nó inicial.

- **Nó final não encontrado**

Um diagrama de descrição de UC deve ter um, e somente um, nó final.

- **O nó inicial deve ter um, e somente um, fluxo de controle sem nenhuma condição de guarda**

Apenas um fluxo de controle sem nenhuma condição de guarda é permitido a partir de um nó inicial.

- **Uma ação deve ter um, e somente um, fluxo de controle sem nenhuma ação de guarda e zero ou mais interrupções (fluxos de controle com <<interrupt>>) para tratamento de exceção**

Apenas um fluxo de controle sem nenhuma condição de guarda é permitido a partir de uma ação. Se houverem outros fluxos esses representam tratamentos de exceções e devem ser obrigatoriamente estereotipados com <<interrupt>>. Nesse caso, a condição de guarda é obrigatória e descreve a exceção a ser tratada.

- **Uma ação deve estar estereotipada como <<actor_action>>, <<system_action>> ou <<system_response>>**

Cada ação do diagrama de descrição do UC deve ser obrigatoriamente estereotipada com um desses três estereótipos.

- **Um nó de decisão deve ter um, e somente um, controle de fluxo sem nenhuma condição de guarda e um ou mais fluxos de controle com condições de guarda**

Apenas um fluxo de controle sem nenhuma condição de guarda é permitido a partir de um nó de decisão. Se houverem outros fluxos esses representam fluxos alternativos e, nesse caso, a condição de guarda é obrigatória e descreve a condição que dispara o fluxo alternativo.

- **Um diagrama de descrição de Caso de Uso pode ter somente um nó inicial, um nó final, ações (opacas ou call behavior) e nós de decisão**

Um diagrama de descrição de UC utiliza somente um subconjunto dos elementos possíveis em um diagrama de atividades da UML 2: nós inicial e final, ações e nós de decisão.

A.19.2 Mensagens de Erro na Especificação do Caso de Uso

Essa verificação é realizada dentro do próprio Use Case Agent e tem o objetivo de garantir que a descrição gerada obedece a todas as restrições do metamodelo UCModel. Caso algum desses erros seja encontrado a geração do diagrama de descrição do UC é abortada. Alguns desses erros são auto-explicativos.

- **Nome do Caso de Uso é obrigatório**
- **Resumo do Caso de Uso é obrigatório**
- **Trigger do Caso de Uso é obrigatória**
- **Pré-condição do Caso de Uso é obrigatória**
- **Pós-condição do Caso de Uso é obrigatória**
- **O destino do goto deve ser definido**
- **A descrição da ação é obrigatória**
- **Regra X tem que ter uma descrição**
- **A regra X não é referenciada por nenhum passo do Caso de Uso**
- **O destino de uma ação goto não pode ser outra ação goto**
- **A inclusão (<<include>>) deve referenciar um caso de uso existente**
- **A extensão (<<extend>>) deve referenciar um caso de uso existente**
- **O Fluxo Alternativo X não é referenciado por nenhum passo do Caso de Uso**
- **O Fluxo de Exceção X não é referenciado por nenhum passo do Caso de Uso**
- **O Caso de Uso tem que ter pelo menos um ator**

Você deve selecionar ao menos um ator para o Caso de Uso. Use a opção “Atores do Caso de Uso” (seção 5.2).

- **Um fluxo tem que ter pelo menos uma ação**

O fluxo principal, alternativo ou de exceção deve ter pelo menos uma ação, ou seja, você não pode criar um fluxo e não defini-lo.

- **Um fluxo alternativo/exceção não pode ter somente uma ação goto**

Fluxos alternativos/exceção devem possuir pelo menos uma ação concreta (ação do ator, ação do sistema ou resposta do sistema). Não são permitidos fluxos alternativos/exceção apenas com um desvio para outro ponto do Caso de Uso.

- **Um fluxo alternativo/exceção tem que ter uma condição**

Um fluxo alternativo/exceção deve sempre definir a condição que o dispara.

- **Uma ação do ator não pode ter exceção**

Exceções são condições inesperadas que ocorrem durante uma ação ou resposta do sistema. Assim, não são permitidas exceções em ações relacionadas aos atores do caso de uso.

- **Uma ação do ator não pode ser a última ação de um fluxo**

A semântica de uma ação do ator indica que, nessa ação, o ator envia informações para o sistema ou fornece as informações solicitadas pelo mesmo. Logo, não faz sentido que essa ação seja a última ação de um fluxo, seja ele principal, alternativo ou exceção, pois não há nenhuma ação posterior para processar a informação enviada.

- **Uma ação do ator deve ser sucedida por uma ação do sistema, goto, include ou extend**

- **Uma ação do sistema deve ser sucedida por outra ação do sistema, uma resposta do sistema, goto, include ou extend**

- **Uma resposta do sistema deve ser sucedida por uma ação do ator, goto, include ou extend**

Esses 3 últimos erros informam que o seqüenciamento das ações não segue o padrão definido no metamodelo.

- **Um goto tem que ser a última ação em um fluxo alternativo/exceção**

Como um goto representa um desvio incondicional, não faz sentido existirem outras ações após o mesmo.

- **Se um fluxo alternativo/exceção é disparado por uma ação do ator, então sua primeira ação tem que ser uma ação do sistema**

- **Se um fluxo alternativo é disparado por uma ação do sistema, então sua primeira ação tem que ser uma resposta do sistema**

- **Se um fluxo de exceção é disparado por uma ação do sistema, então sua primeira ação tem que ser uma resposta do sistema ou outra ação do sistema**
- **Se um fluxo alternativo/exceção é disparado por uma resposta do sistema, então sua primeira ação tem que ser uma resposta do sistema ou ação do sistema**

Esses 4 últimos erros indicam que há um problema entre a ação que dispara o fluxo alternativo/exceção e a primeira ação desses fluxos. As combinações, nesse caso, também devem seguir o metamodelo do UCModel.

- **Um goto não pode ter fluxos alternativos ou de exceção**

Como um goto representa um desvio incondicional, não faz sentido existirem alternativas ou exceções para o mesmo.

- **Um goto após uma ação do ator deve ter como destino uma ação do sistema**
- **Um goto após uma ação do sistema deve ter como destino outra ação do sistema ou uma resposta do sistema**
- **Um goto após uma resposta do sistema deve ter como destino outra resposta do sistema ou uma ação do ator**

Esses 3 últimos erros indicam que há um problema entre a última ação de fluxo alternativo/exceção e a ação destino de um goto. As combinações, nesse caso, também devem seguir o metamodelo do UCModel.

- **Destino do goto X não pertence à cadeia de ativação do fluxo**

O destino de uma ação goto deve ser, obrigatoriamente, uma das ações em um dos fluxos que ativou o fluxo do goto. Por exemplo, se o fluxo principal, ativa o fluxo alternativo A1 e esse, por sua vez, ativa o fluxo alternativo A2, uma ação goto em A2 deve ter como destino uma ação de A1 ou do fluxo principal (que fazem parte da cadeia de ativação de A2).

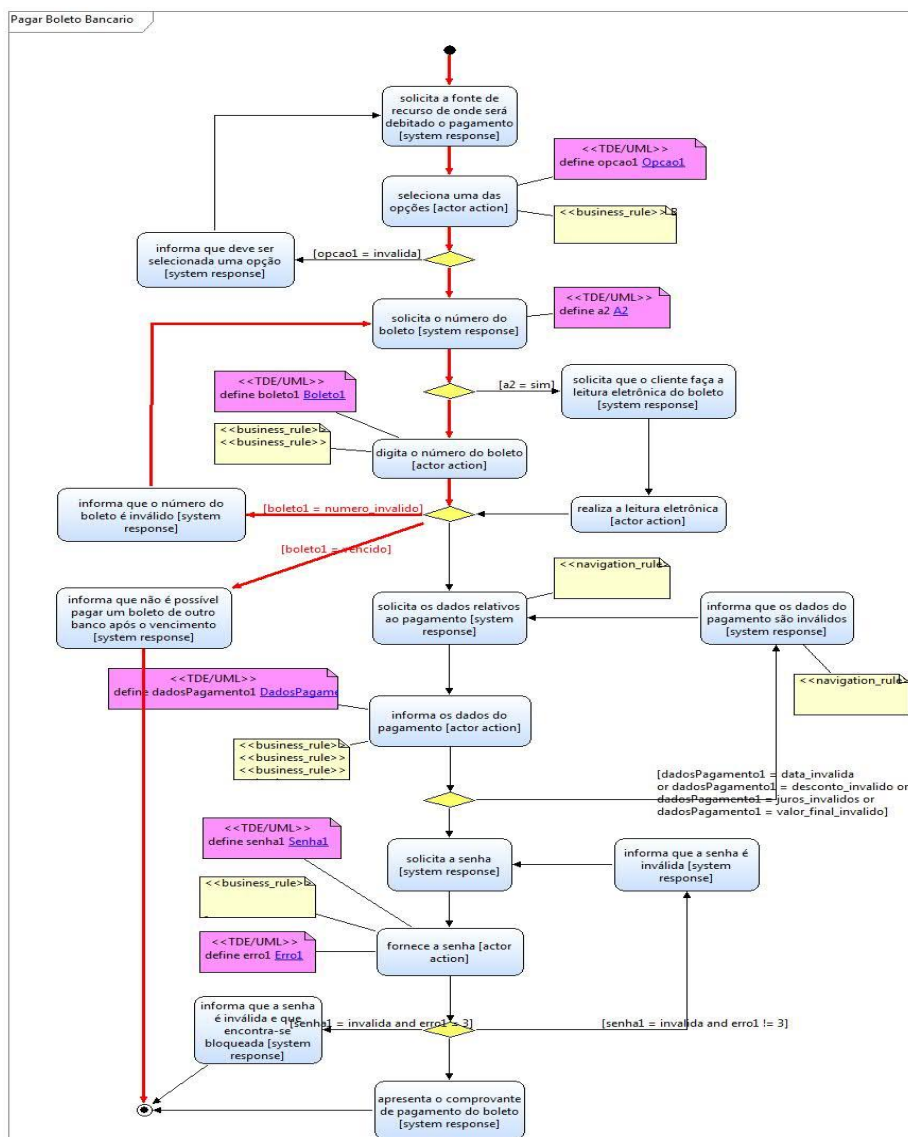
APÊNDICE D – Exemplo de Casos e Procedimentos de Teste Funcional

Este apêndice apresenta o plano de testes funcionais gerado pela ferramenta TDE/UML[®] referente ao caso de uso “Pagar boleto bancário” apresentado como exemplo na seção 6.3, página 146.

1. Test Cases

1.1. Test Case 'TC001_TP1'

1.1.1. Test Case Path



1.1.2. Test Case Steps

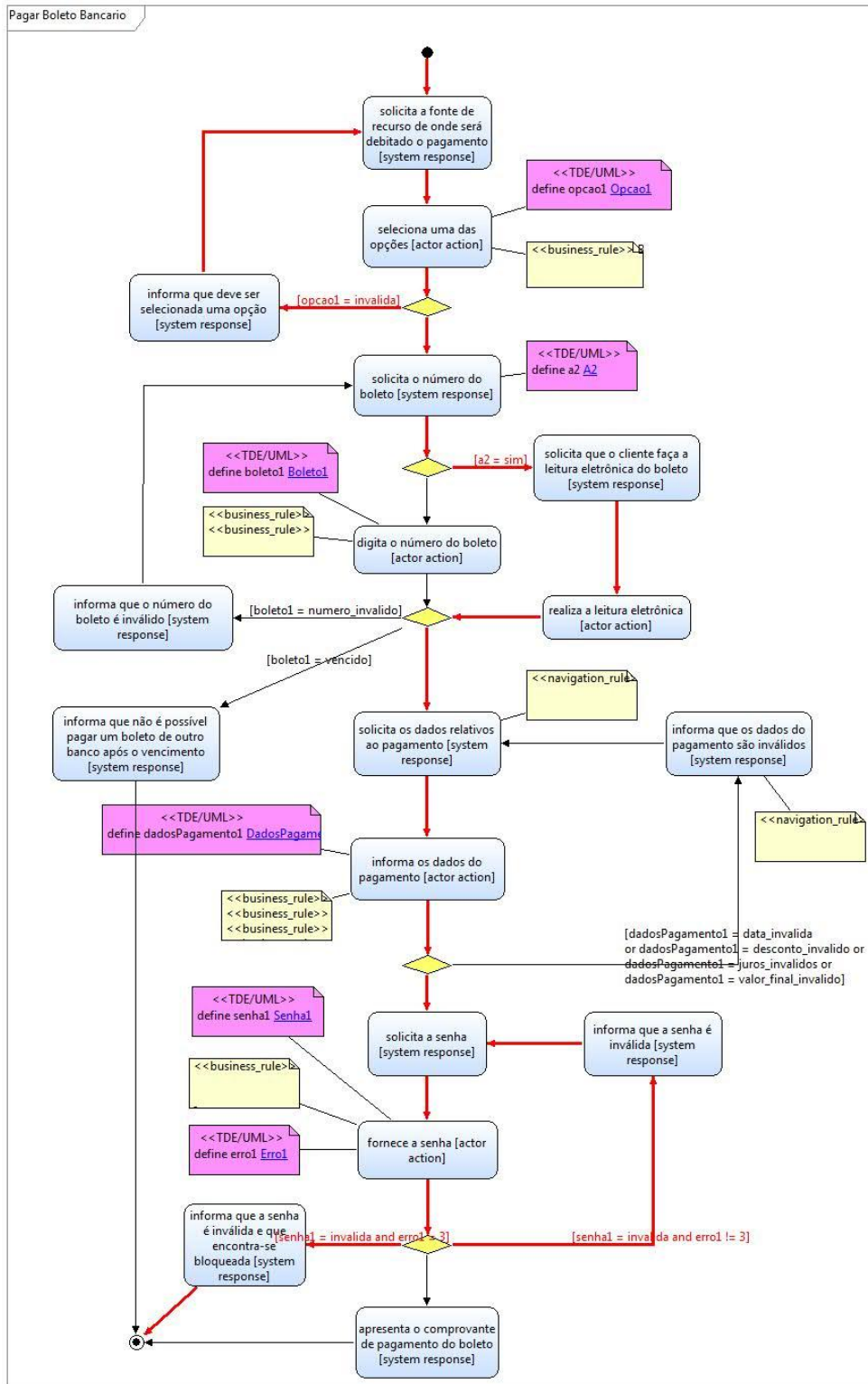
1	solicita a fonte de recurso de onde será debitado o pagamento [system response]	<no description>
2	seleciona uma das opções [actor action]	<no description>
3	solicita o número do boleto [system response]	<no description>
4	digita o número do boleto [actor action]	<no description>
5	informa que o número do boleto é inválido [system response]	<no description>
6	solicita o número do boleto [system response]	<no description>
7	digita o número do boleto [actor action]	<no description>
8	informa que não é possível pagar um boleto de outro banco após o vencimento [system response]	<no description>

1.1.3. Test Case Data Variations

	opcao1		a2		boleto1		
	valida	invalida	sim	nao	numero_invalido	vencido	valido
TC001 TP1							
Proc. 0							
Step 2							
Step 3							
Step 4							
Step 6							
Step 7							

1.2. Test Case 'TC001_TP2'

1.2.1. Test Case Path



1.2.2. Test Case Steps

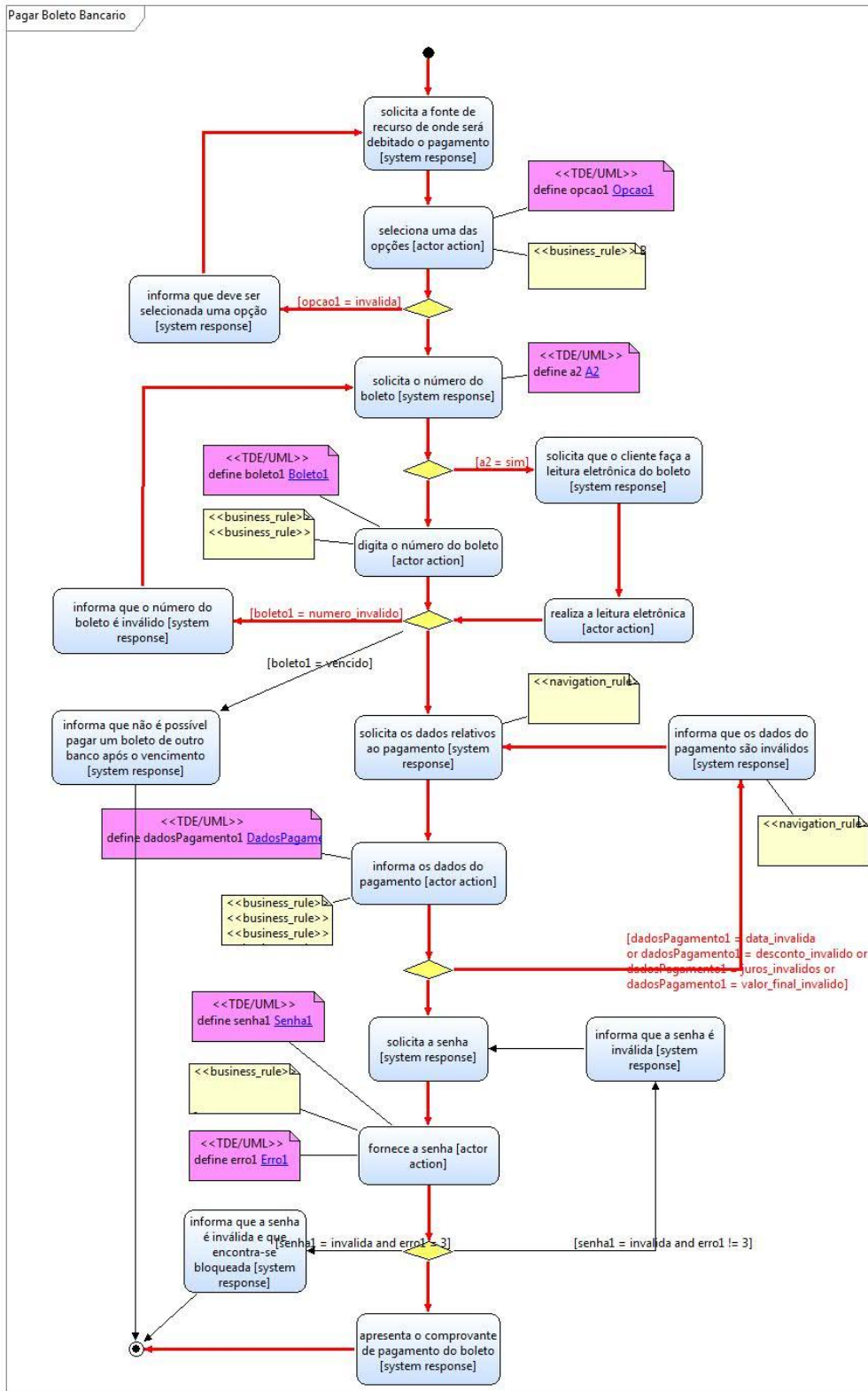
1	solicita a fonte de recurso de onde será debitado o pagamento [system response]	<no description>
2	seleciona uma das opções [actor action]	<no description>
3	informa que deve ser selecionada uma opção [system response]	<no description>
4	solicita a fonte de recurso de onde será debitado o pagamento [system response]	<no description>
5	seleciona uma das opções [actor action]	<no description>
6	solicita o número do boleto [system response]	<no description>
7	solicita que o cliente faça a leitura eletrônica do boleto [system response]	<no description>
8	realiza a leitura eletrônica [actor action]	<no description>
9	solicita os dados relativos ao pagamento [system response]	<no description>
10	informa os dados do pagamento [actor action]	<no description>
11	solicita a senha [system response]	<no description>
12	fornece a senha [actor action]	<no description>
13	informa que a senha é inválida [system response]	<no description>
14	solicita a senha [system response]	<no description>
15	fornece a senha [actor action]	<no description>
16	informa que a senha é inválida e que encontra-se bloqueada [system response]	<no description>

1.2.3. Test Case Data Variations

	opcao1		a2		dadosPagamento1					senha1		errol	
	valida	invalida	sim	nao	data_invalida	desconto_valido	juros_invalidos	valor_final_invalido	valido	valida	invalida	2	3
TC001_TP2													
Proc. 0													
Step 2													
Step 5													
Step 6													
Step 10													
Step 12													
Step 15													

1.3. Test Case 'TC001_TP3'

1.3.1. Test Case Path



1.3.2. Test Case Steps

1	solicita a fonte de recurso de onde será debitado o pagamento [system response]	<no description>
2	seleciona uma das opções [actor action]	<no description>
3	informa que deve ser selecionada uma opção [system response]	<no description>
4	solicita a fonte de recurso de onde será debitado o pagamento [system response]	<no description>
5	seleciona uma das opções [actor action]	<no description>
6	solicita o número do boleto [system response]	<no description>
7	solicita que o cliente faça a leitura eletrônica do boleto [system response]	<no description>
8	realiza a leitura eletrônica [actor action]	<no description>
9	informa que o número do boleto é inválido [system response]	<no description>
10	solicita o número do boleto [system response]	<no description>
11	digita o número do boleto [actor action]	<no description>
12	solicita os dados relativos ao pagamento [system response]	<no description>
13	informa os dados do pagamento [actor action]	<no description>
14	informa que os dados do pagamento são inválidos [system response]	<no description>
15	solicita os dados relativos ao pagamento [system response]	<no description>
16	informa os dados do pagamento [actor action]	<no description>
17	solicita a senha [system response]	<no description>
18	fornece a senha [actor action]	<no description>
19	apresenta o comprovante de pagamento do boleto [system response]	<no description>

1.3.3. Test Case Data Variations

	opcao1		a2		boleto1			dadosPagamento1				senha1		erro1		
	valida	invalida	sim	nao	numero_invalido	vencido	valido	data_invalida	desc_onto_invalido	juros_invalidos	valor_financeiro_invalido	valido	valida	invalida	2	3
TC001_TP3																
Proc. 0																
Step 2																
Step 5																
Step 6																
Step 10																
Step 11																
Step 13																
Step 16																
Step 18																
Proc. 1																
Step 2																
Step 5																
Step 6																
Step 10																
Step 11																
Step 13																
Step 16																
Step 18																
Proc. 2																

1	Choice 'sim'	<no description>
2	Choice 'nao'	<no description>

2.1.5. Category 'Boleto1'

1	Choice 'numero_invalido'	<no description>
2	Choice 'vencido'	<no description>
3	Choice 'valido'	<no description>

2.1.6. Category 'Erro1'

1	Choice '2'	<no description>
2	Choice '3'	<no description>

3. Global Test Data Matrix

	opcao1		a2		boleto1			dadosPagamento1					senha1		erro1	
	valida	invalida	sim	nao	numero_invalido	vencido	valido	data_invalida	desconto_invalido	juros_invalidos	valor_financeiro_invalido	valido	valida	invalida	2	3
TC001_TP1																
Proc. 0																
Step 2	■															
Step 3				■												
Step 4					■	■										
Step 6				■												
Step 7							■									
TC001_TP2																
Proc. 0																
Step 2		■														
Step 5	■															
Step 6			■													
Step 10							■									
Step 12		■														■
Step 15		■														■
TC001_TP3																
Proc. 0																
Step 2		■														
Step 5	■															
Step 6			■													
Step 10				■												
Step 11							■									
Step 13								■								
Step 16							■									
Step 18	■															■
Proc. 1																
Step 2		■														
Step 5	■															
Step 6			■													
Step 10				■												
Step 11							■									
Step 13								■								
Step 16							■									
Step 18	■															■
Proc. 2																
Step 2		■														
Step 5	■															
Step 6			■													
Step 10				■												
Step 11							■									
Step 13								■								
Step 16							■									
Step 18	■															■
Proc. 3																
Step 2		■														

Step 5	■															
Step 6			■													
Step 10				■												
Step 11							■									
Step 13											■	■				
Step 16							■									
Step 18	■															■