UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

OBASCID: UMA ABORDAGEM ONTOLOGICAMENTE FUNDAMENTADA PARA EROA

PAULO AFONSO PARREIRA JÚNIOR

ORIENTADORA: PROFª. DRª. ROSÂNGELA APARECIDA DELLOSSO PENTEADO

São Carlos - SP
Novembro/2015

UNIVERSIDADE FEDERAL DE SÃO CARLOS

CENTRO DE CIÊNCIAS EXATAS E DE TECNOLOGIA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

OBASCID: UMA ABORDAGEM ONTOLOGICAMENTE FUNDAMENTADA PARA EROA

PAULO AFONSO PARREIRA JÚNIOR

Tese apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Doutor em Ciência da Computação, área de concentração: Engenharia de Software.
Orientadora: Profª. Drª. Rosângela A. D. Penteado.

São Carlos - SP
NOVEMBRO/2015

Ficha catalográfica elaborada pelo DePT da Biblioteca Comunitária/UFSCar

P259oa

Parreira Júnior, Paulo Afonso.
ObasCId : uma abordagem ontologicamente fundamentada para EROA / Paulo Afonso Parreira Júnior. -- São Carlos : UFSCar, 2015.
191 f.

Tese (Doutorado) -- Universidade Federal de São Carlos, 2015.

1. Engenharia de software. 2. Identificação de interesses. 3. Classificação de interesses. 4. Interesses transversais. 5. Engenharia de requisitos orientada a aspectos. 6. Ontologias de domínio. I. Título.

CDD: 005.1 (20ª)

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e de Tecnologia
Programa de Pós-Graduação em Ciência da Computação

Folha de Aprovação

Assinaturas dos membros da comissão examinadora que avaliou e aprovou a Defesa de Tese de Doutorado do candidato Paulo Afonso Parreira Junior, realizada em 17/11/2015:

Profa. Dra. Rosangela Aparecida Dellosso Penteado
UFSCar

Prof. Dr. Antonio Francisco do Prado
UFSCar

Prof. Dr. Valter Vieira de Camargo
UFSCar

Prof. Dr. Alessandro Fabricio Garcia
PUC/RJ

Prof. Dr. Eduardo Magno Lages Figueiredo
UFMG

À minha esposa Flávia, ao meu filho Alexandre,
aos meus pais Paulo e Eliana e ao meu irmão Ricardo.

AGRADECIMENTO

Agradeço a Deus pela vida, pelas oportunidades que Ele me oferece, pela saúde, pelo sustento financeiro e emocional e pelas pessoas que colocou em meu caminho para me apoiarem ao longo desta jornada. Dentre essas pessoas, gostaria de destacar as seguintes.

Minha esposa Flávia, pelo apoio incondicional dedicado a mim ao longo desses dez anos de relacionamento, tendo muitas vezes postergado seus sonhos em prol dos meus. Meu filho Alexandre, que alegra nossos dias com sua presença.

Meus pais Paulo e Eliana, pela dedicação, paciência, horas de sono perdidas em meu favor, ensinamentos, dentre outras coisas que contribuíram para meu desenvolvimento como homem, esposo, pai e profissional. Meu irmão Ricardo, pelo companheirismo ao longo de mais de vinte e cinco anos de convívio.

Minha amiga e orientadora Profª. Drª. Rosângela Aparecida Dellosso Penteado, pela amizade e pela importância dada aos sonhos que eu tinha para minha vida profissional e pessoal. Muitos deles só se realizaram devido ao seu imenso apoio.

Meus colegas do GDMS (Grupo de Desenvolvimento e Manutenção de Software), pelas conversas divertidas que faziam a convivência no grupo ser muito agradável. Em especial, meu amigo Matheus Viana, pela ótima companhia e pelas várias vezes em que me hospedou em seu apartamento.

Meus (ex-)alunos e colegas professores com os quais trabalhei e tenho trabalhado na Universidade Federal de Goiás/Regional Jataí, pelas conversas produtivas e descontraídas, que muito contribuíram para meu desenvolvimento profissional e pessoal.

A vida é muito curta para ser pequena.
Benjamin Disraeli

RESUMO

Um interesse de software (*concern*) consiste em um conjunto de requisitos relacionados a um mesmo propósito. Quando um interesse possui requisitos que se encontram entrelaçados com requisitos de outros interesses, trata-se de um "Interesse Transversal" e a modularização inadequada desse tipo de interesse pode dificultar o desenvolvimento e a evolução do software. A área de Engenharia de Requisitos Orientada a Aspectos (EROA) oferece estratégias mais adequadas para identificação, representação e composição de Interesses Transversais (ITs). Um problema comumente relatado nos estudos experimentais sobre abordagens para EROA é a baixa efetividade proporcionada por elas, com relação à identificação e classificação de interesses do software. Isso ocorre por duas causas principais: (i) a falta de uma compreensão mais ampla a respeito do domínio de interesses de software; e (ii) a escassez de recursos apropriados para apoiar engenheiros de software durante a identificação e classificação dos interesses do software. O objetivo deste doutorado é mitigar o problema relatado, atacando suas principais causas por meio de: (i) uma ontologia de referência para o domínio de interesses de software, denominada *O4C* (*Ontology for Concerns*), cujo intuito é tornar clara e precisa a descrição dos elementos desse domínio; (ii) uma abordagem para EROA fundamentada nos conceitos da ontologia *O4C*, denominada *ObasCId* (*Ontologically-based Concern Identification*); e (iii) uma ferramenta computacional para automatização de algumas atividades da abordagem *ObasCId*, denominada *OBasCId-Tool*. A avaliação da abordagem proposta, bem como de seu apoio computacional, foi realizada por meio de estudos experimentais, visando a verificar a efetividade e eficiência desses produtos de pesquisa com relação à identificação e classificação de interesses de software. Os resultados indicaram que a abordagem *ObasCId* pode contribuir positivamente para a cobertura de interesses de software, sem prejudicar a precisão e o tempo de execução dessa abordagem. Quanto à ferramenta *ObasCId-Tool*, a mesma foi considerada satisfatória por seus usuários, com relação às suas características de utilidade e facilidade de uso.

Palavras-chave: Identificação e Classificação de Interesses, Interesses Transversais, *Early-Aspects*, Engenharia de Requisitos Orientada a Aspectos e Ontologias de Domínio.

ABSTRACT

A software concern may be defined as a set of requirements related to the same purpose. A CrossCutting Concern (CCC) (or *Early-Aspect*) is a concern whose requirements cut-across other concern requirements and the inadequate modularization of this type of concern may hinder the software development and evolution. The Aspect-Oriented Requirements Engineering (AORE) area provides more appropriate strategies for CCC identification, representation and composition. A commonly reported issue on AORE approaches is the low effectiveness provided by them, regarding to the concern identification and classification. This is due to two main causes: (i) the lack of a broader understanding about software concerns domain; and (ii) the lack of appropriate resources to support software engineers during the identification and classification of software concerns. This work aims to mitigate the reported issue by dealing with its main causes. To do this, we present: (i) a reference ontology for software concerns domain, called *O4C* (*Ontology for Concerns*), that aims to make clear and precise the description of the elements of this domain; (ii) an ontologically-based AORE approach, called *ObasCId* (*Ontologically-based Concern Identification*); and (iii) a computational tool, called *OBasCId-Tool*, that automates some activities of the *ObasCId* approach. The evaluation of the proposed approach and its computational support was carried out through experimental studies, in order to verify the effectiveness and efficiency of these research products regarding to concern identification and classification. The results indicate the *ObasCId* approach may contribute positively to the recall of software concerns without harming the precision and the execution time of the approach. Regarding to the *ObasCId-Tool*, it has been considered approved by its users, regarding to its usefulness and ease-of-use features.

Keywords: Concern Identification and Classification, Crosscutting Concerns, Early-Aspects, Aspect-Oriented Requirements Engineering and Domain Ontologies.

LISTA DE FIGURAS

Figura 1.1. Exemplo do entrelaçamento entre interesses e o seu impacto no projeto e na codificação orientados a objeto. 15

Figura 2.1. Exemplo de uma matriz de contribuição entre os interesses “Mobilidade” e “Recuperação de Informação” (adaptado de Moreira *et al*., 2005).....27

Figura 2.2. Exemplo de uma visualização de ações (Baniassad e Clarke, 2004). 29

Figura 2.3. Composição entre os temas “*Logging*” e “*BankAccount*”. 30

Figura 2.4. Composição de interesses na ferramenta ARCADE (Chitchyan *et al*., 2006). 31

Figura 2.5. Ontologia para o domínio de requisitos de software (Kaiya e Saeki’s, 2006). 39

Figura 2.6. Ontologia para tomadas de decisão (Bargui *et al*., 2011). 40

Figura 2.7. Elementos para utilização da abordagem GOORE (Shibaoka *et al*., 2007). 41

Figura 2.8. Ontologia para a especificação de requisitos por meio de grafos de objetivos (López *et al*., 2008). 42

Figura 2.10. Visão geral da abordagem *SABiO* (adaptado de Falbo, 2011). 47

Figura 2.11. Exemplo de uso do perfil *OntoUML* (Benevides, 2010).....55

Figura 3.1. Representação gráfica da ontologia para interesses de software *O4C*...60

Figura 4.1. Representação gráfica da abordagem *ObasCId*. 76

Figura 4.2. Atividades da fase “Preparação do Catálogo de Interesses de Software”. 78

Figura 4.3. Exemplo simples de um catálogo de interesses de software. 79

Figura 4.4. Catálogos para interesses não funcionais, desenvolvido a partir de dados históricos do software *Health Watcher*. 80

Figura 4.5. Catálogo para interesses funcionais, desenvolvido a partir de dados históricos do software *Health Watcher*. 81

Figura 4.6. Catálogo de interesses de software desenvolvido a partir da linguagem de padrões para Gestão de Recursos de Negócios (GRN).....81

Figura 4.7. Atividades da fase “Preparação do Documento de Requisitos”. 82

Figura 4.8. Atividades da fase “Identificação e Classificação de Interesses”. 84

Figura 4.9. Atividade da fase “Detecção de Conflitos entre Interesses”.....96

Figura 5.1. Arquitetura da ferramenta *ObasCId-Tool.* 102
Figura 5.2. Cadastro de pesquisadores. 103
Figura 5.3. Autenticação de um pesquisador. 104
Figura 5.4. Repositório público de catálogos de interesses de software. 104
Figura 5.5. Tela para gerenciamento de catálogos de interesses de software. 105
Figura 5.6. Cadastro de um catálogo de interesses de software. 106
Figura 5.7. Configurações padrão (*default*) da ferramenta. 107
Figura 5.8. Listagem de catálogos de interesses de software. 107
Figura 5.9. Visualização de mais informações sobre um catálogo de interesses de software. 108
Figura 5.10. Recursos para entendimento do significado dos ícones utilizados na ferramenta. 109
Figura 5.11. União de dois catálogos de interesses de software pré-cadastrados. 109
Figura 5.12. Tela para gerenciamento de interesses de software. 110
Figura 5.13. Cadastro de um interesse de software. 110
Figura 5.14. Listagem de interesses de software. 111
Figura 5.15. Listagem de interesses do “Catálogo de interesses não funcionais”...111
Figura 5.16. Tela para gerenciamento de palavras-chave. 112
Figura 5.17. Cadastro de palavras-chave. 112
Figura 5.18. Atualização da lista de *stopwords*. 113
Figura 5.19. Lista de palavras-chave para o interesse “Segurança”. 115
Figura 5.20. Cadastro de fontes de um interesse de software. 115
Figura 5.21. Cadastro de relacionamentos entre interesses. 116
Figura 5.22. Listagem de relacionamentos entre interesses. 117
Figura 5.23. Exemplo de relatório geral de um catálogo de interesses de software. 117
Figura 5.24. Exemplo de matrizes de relacionamentos entre interesses de um catálogo. 118
Figura 5.25. Cadastro de documentos de requisitos. 119
Figura 5.26. Cadastro de um requisito de software. 120
Figura 5.27. Listagem de requisitos do software. 120
Figura 5.28. Gerenciamento dos relacionamentos de dependência entre requisitos do software. 121

Figura 5.29. Exemplo de um relatório geral de um documento de requisitos. 122
Figura 5.30. Cadastro de unidades de identificação. 123
Figura 5.31. Listagem de unidades de identificação. 123
Figura 5.32. Interesses funcionais relacionados à ferramenta *ObasCId-Tool*. 124
Figura 5.33. Saída do processo de identificação de interesses de software com a ferramenta *ObasCId-Tool*. 124
Figura 5.34. Explicações sobre as ocorrências do processo de identificação de interesses da abordagem *ObasCId*. 126
Figura 5.35. Elementos relacionados ao interesse “Responsividade”. 127
Figura 5.36. Mensagem com uma ocorrência do tipo IV. 127
Figura 5.37. Cadastro do interesse principal de um requisito. 128
Figura 5.38. Lista de requisitos e interesses identificados atualizada após a especificação dos interesses principais de cada requisito. 128
Figura 5.39. Exemplo de uma matriz de entrelaçamentos entre interesses. 129
Figura 5.40. Exemplo de aplicação do “filtro de requisitos”. 130
Figura 5.41. Lista de requisitos e interesses identificados após a aplicação do “filtro de requisitos”. 130
Figura 5.42. Exemplo de uma matriz de contribuição. 131
Figura 5.43. Matrizes de entrelaçamentos e contribuição atualizadas após a inclusão do requisito “RNF-03”. 132
Figura 5.44. Saída do processo de identificação de interesses com os requisitos da ferramenta *ObasCId-Tool*. 133
Figura 5.45. Matriz de entrelaçamentos do processo de identificação e classificação de interesses da ferramenta *ObasCId-Tool*. 136
Figura 5.46. Matriz de contribuição do processo de identificação e classificação de interesses da ferramenta *ObasCId-Tool*. 137
Figura 6.1. Relacionamento entre o objetivo O1 e suas respectivas questões e métricas. 149
Figura 6.2. Relacionamento entre os objetivos O2 e O3 e suas respectivas questões e métricas. 149

LISTA DE QUADROS

Quadro 2.1. Abordagens para EROA descritas neste trabalho. 25

Quadro 2.2. Trecho de uma matriz de relacionamentos (adaptada de Rashid *et al*., 2003). 35

Quadro 2.3. Exemplo de regra de inferência (Kaiya e Saeki's, 2006). 39

Quadro 2.4. Exemplo de consulta para detecção de conflito entre requisitos (Verma e Kass, 2008). 44

Quadro 2.5. Exemplos de axiomas. 49

Quadro 3.1. Predicados utilizados nos axiomas da ontologia *O4C*. 70

Quadro 3.2. Trabalhos relacionados aos conceitos da ontologia *O4C*. 73

Quadro 4.1. Ícones do meta-modelo *SPEM* utilizados para representação gráfica da abordagem *ObasCId*. 75

Quadro 4.2. Listagem dos artefatos consumidos e gerados/atualizados pelas fases da abordagem *ObasCId*. 76

Quadro 4.3. Modelo de documento de requisitos adotado pela abordagem *ObasCId*. 83

Quadro 4.4. Trecho do documento de requisitos do software *Health Watcher* (2015). 83

Quadro 4.5. Trecho do documento de requisitos de um software para gerenciamento de cursos (adaptado de Baniassad e Clarke, 2004). 83

Quadro 4.6. Exemplo de uma lista de requisitos e interesses identificados. 85

Quadro 4.7. Lista atualizada de requisitos e interesses identificados. 87

Quadro 4.8. Exemplo de uma lista de ocorrências. 88

Quadro 4.9. Descrição de uma ocorrência do tipo I. 89

Quadro 4.10. Descrição de uma ocorrência do tipo II. 89

Quadro 4.11. Descrição de uma ocorrência do tipo III. 90

Quadro 4.12. Descrição de uma ocorrência do tipo IV. 90

Quadro 4.13. Exemplo de uma lista de ocorrências relacionadas à identificação de interesses. 92

Quadro 4.14. Exemplo de uma matriz de entrelaçamentos entre interesses. 94

Quadro 4.15. Exemplo de uma matriz de contribuição entre interesses. 98

Quadro 5.1. Trecho do documento de requisitos da ferramenta *ObasCId-Tool*. 101

Quadro 5.2. Modelo de entrada para importação de requisitos.....120
Quadro 5.3. Exemplo de arquivo para importação de requisitos.....121
Quadro 5.4. Modelo de entrada para importação de relacionamentos de dependência entre requisitos. 121
Quadro 5.5. Lista de requisitos da ferramenta *ObasCId-Tool* e seus interesses principais. 135
Quadro 5.6. Abordagens para EROA e ferramentas de apoio (Parreira Júnior e Penteado, 2013).....140
Quadro 5.7. Disponibilidade e situação das ferramentas computacionais propostas para EROA.....141
Quadro 6.1. Avaliação da abordagem *ObasCId* quanto à sua efetividade e eficiência. 144
Quadro 6.2. Avaliação da ferramenta *ObasCId-Tool* quanto à facilidade de uso percebida por seus usuários. 145
Quadro 6.3. Avaliação da ferramenta *ObasCId-Tool* quanto à utilidade percebida por seus usuários. 145
Quadro 6.4. Questões para o objetivo 1.....145
Quadro 6.5. Questões para o objetivo O2.....146
Quadro 6.6. Questões para o objetivo O3.....146
Quadro 6.7. Métricas diretas para avaliação da abordagem proposta. 147
Quadro 6.8. Métricas indiretas para avaliação da abordagem proposta. 148
Quadro 6.9. Métricas para avaliação da utilidade e facilidade de uso da ferramenta proposta. 149
Quadro 6.10. Interpretação das métricas do modelo de avaliação proposto. 150
Quadro 6.11. Hipóteses para o Estudo Experimental I. 152
Quadro 6.12. Projeto do estudo experimental I.....153
Quadro 6.13. Teste de hipóteses para o experimento com o software *LocaDVD*...160
Quadro 6.14. Hipóteses para o estudo experimental II. 163
Quadro A.8.1. Documento de requisitos da ferramenta *ObasCId-Tool*. 182

LISTA DE ABREVIATURAS E SIGLAS

EA-Miner	*Early-Aspects Mining*
ER	Engenharia de Requisitos
EROA	Engenharia de Requisitos Orientada a Aspectos
EROA/Arcade	*AORE with Arcade*
EROA/XML	*An XML-Based Language for Specification and Composition of Aspectual Concerns*
GQM	*Goal-Question-Metric*
IT	Interesse Transversal
ITs	Interesses Transversais
MDSoC	*Multi-Dimensional Separation of Concerns*
MS	Mapeamento Sistemático
RF	Requisito funcional
RNF	Requisito não funcional
SoC	*Separation of Concerns*
TAM	*Technology Acceptance Model*
Theme	Abordagem *Theme*

CONVENÇÕES ADOTADAS

As seguintes convecções de nomenclatura e formatação foram adotadas ao longo do texto desta tese:

- O nome dos interesses de software e conceitos da ontologia de fundamentação *UFO*, bem como da ontologia *O4C*, são apresentados entre aspas duplas. Caso o nome desse elemento seja uma palavra estrangeira, a mesma é colocada em itálico. Exemplos: “Desempenho”, “Segurança”, “*Concern*” e “*Logging*”;
- Nomes de estereótipos do perfil *OntoUML*, das classes, dos atributos, bem como o texto das listagens apresentadas nesta tese são formatados com fonte `Consolas`, tamanho 11pt;
- Expressões e termos que merecem destaque aparecem em **negrito**, caso estejam no início de uma enumeração de itens, ou em *itálico*, caso estejam no interior do texto de um parágrafo;
- As siglas e a descrição dos requisitos de software, quando estiverem no interior de um parágrafo, são colocadas entre aspas duplas;
- As siglas das abordagens apresentadas nesta tese são destacadas em *itálico* (*e. g. EA-Miner, SABiO*, *UFO*, *ObasCId*, entre outras). Porém, siglas para conceitos e expressões referentes ao tema deste doutorado, tais como EROA, IT, ER, são mantidas sem itálico; e
- Os conceitos “Quadro” e “Tabela” são utilizados com base nas definições dadas pela Associação Brasileira de Normas Técnicas (ABNT). Tabela: forma não discursiva de apresentar informações das quais o dado numérico se destaca como informação central. Quadro: uma das categorias de ilustrações. Com relação à formatação, nas tabelas, ao contrário dos quadros, as bordas laterais não podem ser fechadas.

SUMÁRIO

CAPÍTULO 1 - INTRODUÇÃO.....13
1.1 Contextualização.....13
1.2 Motivação.....16
1.3 Objetivos.....18
1.4 Método para Desenvolvimento do Trabalho.....20
1.5 Organização do Trabalho.....21
CAPÍTULO 2 - FUNDAMENTAÇÃO TEÓRICA.....23
2.1 Considerações Iniciais.....23
2.2 Engenharia de Requisitos Orientada a Aspectos - EROA.....24
2.2.1 *Separação Multidimensional de Interesses na Engenharia de Requisitos* (*MDSoC - Multi-Dimensional Separation of Concerns in Requirements Engineering*) 25
2.2.2 Abordagem *Theme*.....28
2.2.3 Abordagem *EA-Miner*.....30
2.2.4 EROA/XML - Uma Abordagem baseada em XML para Especificação e Composição de Interesses Aspectuais.....32
2.2.5 EROA/Arcade - Engenharia de Requisitos Orientada a Aspectos com Arcade34
2.2.6 Considerações finais sobre as abordagens para EROA.....36
2.3 Ontologias de Domínio no Contexto da Engenharia de Requisitos.....37
2.3.1 Uso de ontologias de domínio para verificação de requisitos.....38
2.3.2 Extração de conhecimento a partir de documentos de requisitos com base em ontologias de domínio.....39
2.3.3 Uso de ontologias de domínio para guiar a elicitação e análise de requisitos..40
2.3.4 Uso de ontologias como forma de estabelecer uma conceituação básica acerca do domínio de requisitos.....42
2.3.5 Uso de ontologias de domínio para detecção e análise de conflitos entre requisitos.....43
2.3.6 Uso de ontologias de domínio para promover a rastreabilidade entre requisitos e outros tipos de abstrações de software.....44

2.3.7 Considerações finais sobre o uso de ontologias de domínio no contexto da engenharia de requisitos 44
2.4 Desenvolvimento de Ontologias de Domínio 46
2.4.1 Abordagem *SABiO* 46
2.4.2 Ontologia de fundamentação *UFO* 49
2.4.3 O perfil *OntoUML* 52
2.5 Considerações Finais 55

CAPÍTULO 3 - *O4C*: UMA ONTOLOGIA PARA O DOMÍNIO DE INTERESSES DE SOFTWARE 57

3.1 Considerações Iniciais 57
3.2 Identificação do Propósito da Ontologia *O4C* e Elicitação dos seus Requisitos 58
3.3 Formalização Ontológica da *O4C* 59
3.3.1 *Concern*, *FuncionalConcern* e *NonFuncionalConcern* 61
3.3.2 *Keyword e Source* 62
3.3.3 *Contribution*, *Dependency* e *Composition* 63
3.3.4 *Priority* 65
3.3.5 *Requirement*, *FunctionalRequirement*, *NonFunctionalRequirement* e *RequirementsDependency* 66
3.3.6 *FunctionalCCC* e *NonFunctionalCCC* 67
3.3.7 *Indication, MainConcernIndication, FunctionalCCCIndication* e *NonFunctionalCCCIndication* 68
3.3.8 Axiomas 70
3.4 Considerações Finais 72

CAPÍTULO 4 - *OBASCID*: UMA ABORDAGEM ONTOLOGICAMENTE FUNDAMENTADA PARA EROA 74

4.1 Considerações Iniciais 74
4.2 Visão Geral da Abordagem *ObasCId* 75
4.3 Preparação do Catálogo de Interesses de Software 78
4.4 Preparação do Documento de Requisitos 82
4.5 Identificação e Classificação de Interesses 84
4.5.1 Identificar interesses a partir de palavras-chave 84
4.5.2 Identificar interesses a partir da interdependência entre requisitos de software 86

4.5.3 Especificar interesses principais 87
4.5.4 Verificar resultados da identificação de interesses 88
4.5.5 Classificar interesses 93
4.6 Detecção de Conflitos entre Interesses 95
4.7 Considerações Finais 99

CAPÍTULO 5 - *OBASCID-TOOL*: UMA FERRAMENTA DE APOIO À EROA BASEADA NA ABORDAGEM *OBASCID* 100

5.1 Considerações Iniciais 100
5.2 Trecho do documento de requisitos da Ferramenta *ObasCId-Tool* 101
5.3 Visão Geral da Ferramenta *ObasCId-Tool* 101
5.4 Módulo de Gerenciamento de Pesquisadores 103
5.5 Módulo de Consulta aos Repositórios 104
5.6 Módulo de Gerenciamento de Catálogos de Interesses de Software 105
5.6.1 Gerenciamento dos elementos básicos de uma instância 105
5.6.2 Gerenciamento dos interesses de um catálogo 109
5.6.3 Gerenciamento dos relacionamentos entre interesses 115
5.6.4 Relatórios de um catálogo de interesses de software 116
5.7 Módulo de Gerenciamento de Documentos de Requisitos 118
5.7.1 Gerenciamento de requisitos 119
5.7.2 Relatório de um documento de requisitos 122
5.8 Módulo de Identificação e Classificação de Interesses 122
5.9 Identificação e Classificação dos Interesses da ferramenta *ObasCId-Tool* 132
5.10 Características Técnicas e Mecanismos de Extensão 137
5.11 Considerações Finais 139

CAPÍTULO 6 - ESTUDOS EXPERIMENTAIS 143

6.1 Considerações Iniciais 143
6.2 Modelo GQM dos Estudos Experimentais 144
6.2.1 Nível conceitual: definição dos objetivos de avaliação 144
6.2.2 Nível operacional: definição das questões 145
6.2.3 Nível quantitativo: definição das métricas 147
6.3 Estudo Experimental I – *ObasCId* vs. *Theme/Doc* 150
6.3.1 Planejamento 151
6.3.2 Análise dos Resultados 154

6.3.3 Teste de Hipóteses 158
6.3.4 Ameaças à Validade do Estudo 161
6.4 Estudo Experimental II – Facilidade de uso e utilidade da ferramenta *ObasCId-Tool* 162
6.4.1 Planejamento 162
6.4.2 Análise dos Resultados 163
6.4.3 Teste de Hipóteses 166
6.4.4 Ameaças à Validade do Estudo 168
6.5 Considerações Finais 169
CAPÍTULO 7 - CONSIDERAÇÕES FINAIS 170
7.1 Contribuições e Limitações 172
7.2 Trabalhos Futuros 174
REFERÊNCIAS 176
CAPÍTULO 8 - REQUISITOS DA FERRAMENTA *OBASCID-TOOL* 182
A.1 Requisitos da Ferramenta *ObasCId-Tool* 182
CAPÍTULO 9 - MATERIAIS UTILIZADOS NOS ESTUDOS EXPERIMENTAIS 185
B.1 Materiais utilizados no Estudo Experimental I 185
B.2 Materiais utilizados no Estudo Experimental II 189

Capítulo 1

INTRODUÇÃO

1.1 Contextualização

Um requisito de software define uma propriedade ou capacidade que deve estar disponível no software com o objetivo de atender às regras de negócio para as quais esse requisito foi concebido (Sommerville, 2011; Chitchyan *et al*., 2005).

Tradicionalmente, requisitos de software são classificados como *funcionais* ou *não funcionais*. Requisitos funcionais definem a funcionalidade de um software ou de seus componentes, como por exemplo, “acionar a sirene quando o sensor de presença for ativado”, “permitir o processamento de pedidos de passaporte”, entre outros. Requisitos não funcionais estão relacionados a restrições impostas (Sommerville, 2011): (i) *à funcionalidade do software*, como por exemplo, “as senhas dos usuários devem ser armazenadas de forma criptografada” e “o tempo de resposta para processamento de um pedido não deve ser superior a 5 (cinco) segundos”; (ii) *ao processo de desenvolvimento do software*, como por exemplo, “o *deadline* de entrega do software não pode ultrapassar o período de 2 (dois) anos”; entre outros.

Um conjunto de requisitos de software relacionados a um mesmo propósito é definido como um *interesse* (*concern*) (Herrera *et al*., 2012; Sampaio *et al*., 2007). Por exemplo, o interesse “Segurança” pode contemplar diversos requisitos relacionados ao propósito “garantir que o software seja seguro”. Interesses também podem ser classificados

como *funcionais* ou *não funcionais*, caso eles contemplem[1] requisitos funcionais ou não funcionais, respectivamente.

Ao longo do ciclo de desenvolvimento do software, seus interesses vão sendo transformados em artefatos de mais baixo nível de abstração, até serem completamente implementados em alguma linguagem de programação. Com base no princípio da Separação de Interesses (*Separation of Concerns - SoC*) (Dijkstra, 1976), idealmente, cada interesse deveria estar alocado em um único módulo do software, que possuísse a responsabilidade de satisfazer somente aos requisitos relacionados a ele. Quando essa alocação é possível, o software resultante é dito bem modularizado (Sampaio *et al*., 2007).

Para alguns interesses, essa clara alocação em módulos não é possível apenas utilizando as abstrações usuais da engenharia de software, tais como, casos de uso, pontos de vista, objetivos, cenários, classes e objetos, entre outros (Rashid *et al*., 2003). A esses interesses, dá-se o nome de "Interesses Transversais (ITs)"[2] ou "*Early-Aspects*". Um interesse é dito transversal quando há requisitos desse interesse que estão entrelaçados (ou entrecortam – *cut-across*) com requisitos de outros interesses do software (Herrera *et al*., 2012; Sampaio *et al*., 2007; Rashid *et al*., 2003). Alguns exemplos de ITs bem conhecidos pela comunidade científica são "Segurança", "Persistência", "Desempenho", "Concorrência", "Distribuição", "*Logging*", entre outros (Baniassad *et al*., 2006).

Para exemplificar o que foi comentado anteriormente, considere os requisitos "estudantes podem se matricular em um curso" e "quando um estudante se matricular em um curso, um registro (*log*) desse evento deve ser persistido no software", ambos extraídos do documento de requisitos de um software para gerenciamento de cursos apresentado por Baniassad e Clarke (2004). Segundo os autores, o propósito do primeiro requisito é explicitar um comportamento funcional do software, relacionado ao interesse "Gerenciamento de Matrículas", e o do segundo requisito é especificar um comportamento não funcional, referente ao interesse conhecido como "*Logging*". É possível notar que o primeiro requisito depende da ação "persistir um registro de *log*", do segundo requisito. Nesse caso, diz-se que "*Logging*" atua como um IT, pois há requisitos desse interesse que se encontram entrelaçados com requisitos de outros interesses do software.

Tarr *et al.* (1999) afirmam que cada método de modelagem e programação de software impõe uma maneira de decomposição e modularização de interesses, como por

[1] O termo "contemplar" significa que os interesses em questão estão relacionados a requisitos criados especificamente para atender a esses interesses, o que é diferente do conceito de "entrelaçamento", como é explicado mais a frente neste texto.

[2] Daqui em diante, esse tipo de interesse será referenciado pela expressão "Interesse Transversal - IT" ou "Interesses Transversais - ITs".

exemplo, classes e objetos, dados e procedimentos, funções, entre outros. Essa imposição é conhecida como “tirania da decomposição dominante” e é por meio dela que o entrelaçamento existente entre os interesses supracitados pode vir a gerar representações entrelaçadas e espalhadas ao longo do ciclo de desenvolvimento do software. Por exemplo, na Figura 1.1 é apresentado um possível impacto que o entrelaçamento entre interesses pode causar no projeto e na codificação orientados a objetos. Como pode ser visto, a implementação dos interesses “Gerenciamento de Matrículas” e “*Logging*”, em uma linguagem orientada a objetos, levaria a trocas de mensagens explícitas entre os módulos (classes, pacotes, entre outros) que os representam, aumentando o acoplamento e reduzindo a coesão entre esses módulos.

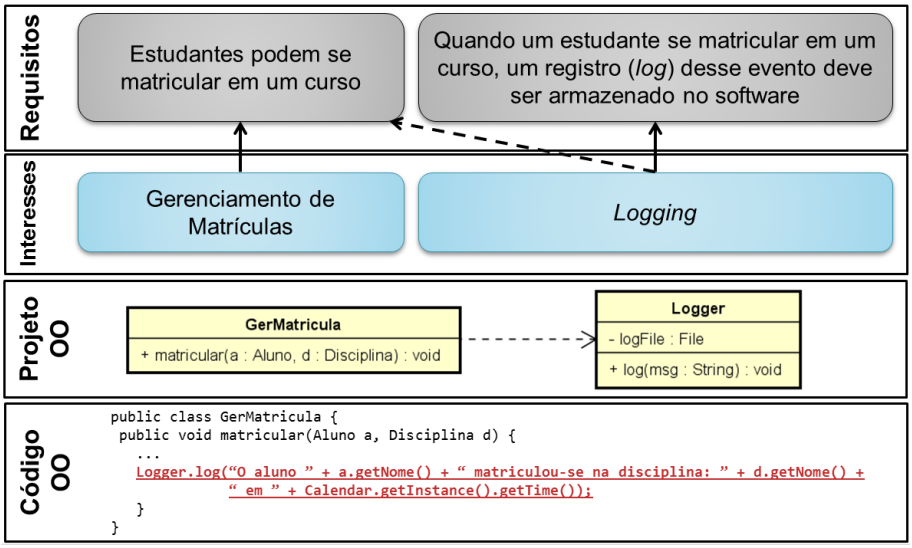


Figura 1.1. Exemplo do entrelaçamento entre interesses e o seu impacto no projeto e na codificação orientados a objeto.

A modularização inadequada dos interesses de um software interfere no raciocínio do engenheiro de software quanto aos efeitos provocados pela inclusão, remoção ou alteração de algum requisito sobre os demais, dificultando a rastreabilidade, o entendimento, a manutenção e a evolução do software (Tarr *et al.*, 1999; Chitchyan *et al*., 2005). Por exemplo, caso o primeiro requisito do exemplo anterior fosse alterado de tal forma que não fosse mais o aluno quem efetuasse sua matrícula, mas sim a secretária do curso, então além do interesse “Gerenciamento de Matrículas”, aqueles interesses cujos requisitos estão entrelaçados a ele também deveriam ser observados/modificados pelo engenheiro de software.

Para minimizar as consequências trazidas por esse problema, são necessários métodos, técnicas e ferramentas que possam tratar adequadamente dos ITs do software (Tarr *et al.*, 1999). Assim, várias pesquisas têm sido conduzidas no contexto da Engenharia de Requisitos Orientada a Aspectos (EROA), em inglês, *Aspect-Oriented Requirements Engineering* (AORE) (Araújo *et al*., 2004; Baniassad e Clarke, 2004; Rashid *et al*., 2003;

Grundy, 1999) para promover melhorias quanto à separação de interesses durante as fases iniciais do desenvolvimento do software. Para isso, a EROA visa a oferecer estratégias mais adequadas para identificação, representação e composição de ITs.

1.2 Motivação

Nos últimos anos, várias abordagens para EROA foram desenvolvidas (Chitchyan *et al*., 2006; Soeiro *et al*., 2006; Clarke e Baniassad, 2005; Moreira *et al*., 2005; Sampaio *et al*., 2005; Baniassad e Clarke, 2004; Rashid *et al*., 2003; Rashid *et al*., 2002)[3]. Em geral, tais abordagens contemplam uma ou mais das seguintes atividades:

i) **Identificação e Classificação de Interesses**: consiste em descobrir os interesses existentes no software, classificando-os como base[4] ou transversais;

ii) **Representação de Interesses**: consiste em representar os interesses identificados por meio de algum tipo de notação textual e/ou gráfica, especificando como os ITs afetam outros interesses do software e em quais requisitos isso ocorre;

iii) **Composição de Interesses**: refere-se à especificação de como os interesses relacionados (transversais e base) devem ser combinados para se atingir o objetivo proposto para o software; e

iv) **Detecção e Análise de Conflitos**: consiste em investigar a influência mútua existente entre diferentes interesses, a fim de se detectar possíveis conflitos entre eles. Por exemplo, a criptografia necessária para atender o interesse “Segurança” pode fazer com que esse interesse contribua negativamente para o interesse “Desempenho”.

A importância da EROA, em particular da atividade “Identificação e Classificação de Interesses”, baseia-se no princípio de que “quanto mais cedo ocorrer a identificação dos ITs no ciclo de desenvolvimento do software, menor será o esforço necessário para modularizá-los adequadamente” (Resende, 2007). Contudo, estudos experimentais realizados com

[3] A lista completa de abordagens para EROA, identificadas por meio de um Mapeamento Sistemático sobre esse assunto, pode ser encontrada nos trabalhos de Parreira Júnior e Penteado (2014) e Parreira Júnior e Penteado (2015e).

[4] Interesses base (ou não transversais), no contexto da EROA, são aqueles cujos requisitos não estão entrelaçados com requisitos de outros interesses do software (Herrera *et al*., 2012; Sampaio *et al*., 2007).

algumas das principais abordagens para EROA têm destacado alguns problemas com relação à efetividade dessa atividade, são eles (Sampaio *et al*., 2007; Herrera *et al*. 2012; Parreira Júnior e Penteado, 2015b, Parreira Júnior e Penteado, 2015c):

- **A identificação de interesses base geralmente é melhor do que a de interesses transversais**. Em Sampaio *et al*. (2007), a cobertura média de interesses base provida pelas abordagens avaliadas foi de aproximadamente 89%, enquanto que para ITs foi de 52%. No trabalho de Herrera *et al*. (2012), a situação se repete, uma vez que, em média, 77% dos interesses base foram identificados, contra 66% referentes aos ITs; e
- **A precisão na identificação dos interesses transversais é aceitável, entretanto, a cobertura é baixa**. Isto quer dizer que, de todos os interesses identificados e classificados como transversais por meio dessas abordagens, a maior parte é realmente transversal; porém, nem todos os interesses transversais existentes no software foram corretamente identificados. Sampaio *et al*. (2007) e Herrera *et al*. (2012) citam que a precisão média oferecida pelas abordagens para EROA analisadas variou entre 71% e 90%, enquanto que a cobertura média ficou entre 52% e 66%. Nos trabalhos de Parreira Júnior e Penteado (2015b, 2015c), a precisão média das abordagens avaliadas ficou em 91%, enquanto que a cobertura média variou entre 41% (abordagem *Theme/Doc* – apresentada no Capítulo 2) e 74% (abordagem *ObasCId*, proposta nesta tese e apresentada no Capítulo 4).

É possível elencar, pelo menos, duas principais causas para os problemas citados anteriormente.

Causa 1: falta de compreensão mais ampla a respeito do domínio dos interesses de software. Há escassez de estudos cujo intuito seja oferecer uma compreensão clara e consensual sobre o que vem a ser um interesse de software, quais são suas principais características, como ele se relaciona com outros interesses e com os requisitos de um software, entre outros. Assim, o conhecimento sobre o domínio de interesses de software é representado de forma divergente nas diversas abordagens existentes para EROA, o que dificulta o entendimento e a utilização de tais abordagens; e

Causa 2: escassez de recursos apropriados para apoiar engenheiros de software durante a identificação e classificação dos interesses do software. Muitas abordagens para EROA baseiam-se apenas na experiência dos engenheiros de software para a correta identificação dos interesses do software, o que pode levar à diminuição da precisão e da cobertura dessas abordagens.

Na Seção 2.2.6 são apresentados mais detalhes sobre essas causas, analisando-as com base nos trabalhos relacionados ao tema deste doutorado.

1.3 Objetivos

Em resumo, o problema de pesquisa originário desta tese consiste na baixa efetividade proporcionada pelas abordagens para EROA, com relação à identificação e classificação de interesses a partir de requisitos de software. Conforme comentado, as principais causas desse problema são a falta de uma compreensão mais ampla sobre o domínio dos interesses de software e a escassez de recursos adequados para apoiar os engenheiros de software durante a identificação e classificação de interesses. Assim, o objetivo deste doutorado é mitigar o problema relatado, atacando suas principais causas por meio de:

- **Uma ontologia de domínio para interesses de software, denominada *O4C* (*Ontology for Concerns*)**, que consiste em um modelo conceitual responsável por representar conceitos e relacionamentos bem conhecidos e já documentados na literatura sobre o domínio dos interesses de software. Além disso, a partir da instanciação dos conceitos e relacionamentos dessa ontologia, pode-se confeccionar catálogos para tipos específicos de interesses e utilizá-los como apoio para a identificação e classificação de interesses a partir de requisitos de software;
- **Uma abordagem ontologicamente fundamentada para EROA, denominada *ObasCId (Ontologically-based Concern Identification)***, que descreve um conjunto de atividades necessárias para identificação e classificação de interesses a partir de requisitos de software, bem como para detecção e análise de conflitos entre os interesses identificados. A expressão ontologicamente fundamentada refere-se ao fato de que a abordagem *ObasCId* foi desenvolvida com base nos conceitos e relacionamentos descritos na ontologia *O4C* e que utiliza instâncias desses elementos para seu funcionamento; e
- **Uma ferramenta computacional denominada *OBasCId-Tool***, que automatiza algumas das atividades da abordagem *ObasCId*.

A hipótese sobre a qual o objetivo deste trabalho se baseia considera que o uso de uma ontologia para o domínio de interesses de software pode mitigar as causas do problema relatado anteriormente. Como justificativa, tem-se a própria definição de *ontologia de domínio*, isto é, “um tipo especial de especificação conceitual, cujo objetivo é tornar clara e precisa a descrição dos elementos de um domínio para o propósito de comunicação, aprendizado e resolução de problemas” (Guizzardi, 2005; Falbo, 2011). Essa definição vai

ao encontro da primeira causa para o problema, isto é, a falta de compreensão sobre o domínio de interesses de software.

Além disso, a base de conhecimento representada pelas instâncias dos conceitos e relacionamentos da ontologia proposta, juntamente com a descrição de um conjunto de passos para utilização das mesmas, pode aprimorar a efetividade da atividade de identificação e classificação de interesses a partir de requisitos de software, mitigando os efeitos da segunda causa para o problema. Por fim, acredita-se ainda que o entendimento mais amplo sobre o domínio dos interesses de software, em especial dos interesses transversais, pode permitir a construção de métodos, técnicas e ferramentas de apoio à EROA que sejam amplamente utilizáveis e compatíveis, por terem como base, definições compartilhadas sobre um mesmo domínio.

A linha de pesquisa sobre ontologias de domínio tem sido intensamente estudada e aplicada na solução de problemas de diversas áreas, como Inteligência Artificial, Processamento de Linguagem Natural, Recuperação da Informação, Banco de Dados e Engenharia de Software (Guizzardi, 2005). Na área de Engenharia de Requisitos, especificamente, o uso de ontologias de domínio também tem sido bastante explorado (Parreira Júnior e Penteado, 2015a; Dermeval *et al*., 2015), porém, não foram encontrados estudos específicos sobre o uso de ontologias para caracterização do domínio de interesses de software no contexto da EROA, nem sobre o uso de instâncias dos conceitos dessas ontologias como apoio para a identificação e classificação de interesses a partir de requisitos de software.

A decisão de se utilizar ontologias de domínio para especificação conceitual do domínio de interesses de software ao invés de outros tipos de modelagem conceitual foi tomada com base na maturidade e no grau de formalização apresentado/requerido por essa área de pesquisa. Além disso, trata-se de uma área para a qual há um vasto arcabouço de métodos, técnicas e ferramentas disponível na literatura para confecção, validação e uso desse tipo de modelo conceitual (Parreira Júnior e Penteado, 2015a).

Como objetivos específicos deste doutorado, têm-se:

- Estabelecer o estado da arte referente à EROA e ao uso de ontologias no contexto da Engenharia de Requisitos;
- Prover uma definição conceitual clara a respeito do domínio de interesses de software, destacando os principais conceitos e relacionamentos envolvidos com esse domínio;
- Prover recursos apropriados aos engenheiros de software para que eles possam desempenhar a atividade de identificação e classificação de interesses em requisitos de software de forma mais efetiva; e

- Oferecer apoio computacional adequado para geração e compartilhamento do conhecimento a respeito dos interesses de software, bem como para identificação e classificação de interesses a partir de requisitos de software.

1.4 Método para Desenvolvimento do Trabalho

O método utilizado para desenvolvimento desta tese consistiu na realização de estudos bibliográficos, por meio de Mapeamentos Sistemáticos da Literatura, com o objetivo de estabelecer o estado da arte referente aos assuntos relacionados a essa pesquisa, a saber, “Engenharia de Requisitos Orientada a Aspectos” (Parreira Júnior e Penteado, 2015d; Parreira Júnior e Penteado, 2014) e “Uso de Ontologias no Contexto da Engenharia de Requisitos” (Parreira Júnior e Penteado, 2015a).

Uma vez tendo concluído a etapa de levantamento bibliográfico, iniciou-se a leitura mais profunda dos principais trabalhos relacionados ao tema deste doutorado, a fim de se identificar o problema a ser atacado nesta pesquisa. Após a definição do problema de pesquisa e da solução proposta para o mesmo, iniciou-se a etapa de definição da ontologia para o domínio de interesses de software, denominada *O4C* (*Ontology for Concerns*) e da abordagem para EROA, *ObasCId* (*Ontologically-based Concern Identification*). À medida que a ontologia e a abordagem foram sendo desenvolvidas/refinadas, estudos experimentais foram conduzidos para avaliação das mesmas (Parreira Júnior e Penteado, 2015b; Parreira Júnior e Penteado, 2015c).

Após a realização de estudos experimentais preliminares a respeito da ontologia de domínio e da abordagem proposta, desenvolveu-se a ferramenta computacional necessária para automatização de algumas atividades da abordagem *ObasCId*, denominada *ObasCId-Tool*. O modelo de processo utilizado para desenvolvimento dessa ferramenta foi o iterativo e incremental, no qual pequenas porções da ferramenta eram desenvolvidas e testadas antes de serem incorporadas à versão final da mesma. Por fim, novos estudos experimentais foram conduzidos a fim de avaliar a ferramenta desenvolvida (Capítulo 6).

Como principais resultados desta pesquisa, têm-se:

- O desenvolvimento da ontologia *O4C* permitiu que alguns conceitos sobre o domínio de interesses de software, não contemplados pelas abordagens para EROA existentes na literatura, pudessem ser descobertos, entendidos e utilizados no processo de identificação e classificação de interesses. Além disso, permitiu-se que conceitos comuns, mas representados de formas distintas entre

as abordagens para EROA, fossem consolidados em um único modelo conceitual;

- Verificou-se que o uso da abordagem *ObasCId* pode aprimorar a cobertura de interesses em requisitos de software, sem impactar na precisão e no tempo de execução da abordagem (Parreira Júnior e Penteado, 2015b; Parreira Júnior e Penteado, 2015c); e
- A ferramenta computacional *ObasCId-Tool* foi considerada um importante recurso de apoio à EROA por parte de seus usuários. Notou-se ainda que, com um breve treinamento de 40 (quarenta) minutos, a maioria dos usuários da ferramenta conseguiu elaborar e gerenciar os recursos necessários para identificação e classificação de interesses a partir de requisitos de software.

1.5 Organização do Trabalho

Esta tese está organizada em 7 (sete) capítulos e 2 (dois) apêndices. No Capítulo 1 foi contextualizado o problema, bem como discutidas a motivação e as justificativas deste doutorado. Além disso, foram apresentados os objetivos do trabalho, o método utilizado para alcançá-los, bem como os principais resultados obtidos.

No Capítulo 2 são apresentados os principais conceitos referentes à EROA e ao uso de ontologias no contexto da ER, bem como a descrição de alguns trabalhos relacionados a esses assuntos. Além disso, são apresentadas a abordagem para desenvolvimento de ontologias *SABiO* (*Systematic Approach for Building Ontologies*) (Falbo, 2011), a ontologia de fundamentação *UFO* (*Unified Foundation Ontology*) e o perfil *OntoUML* (Guizzardi, 2005), que compõem a base teórica para desenvolvimento da ontologia proposta neste trabalho.

Nos Capítulos 3, 4 e 5 são apresentadas as principais contribuições deste doutorado. No Capítulo 3 é apresentada a ontologia para o domínio de interesses de software, denominada *O4C*, bem como o processo para desenvolvimento da mesma. No Capítulo 4 é apresentada a abordagem para EROA fundamentada nos conceitos e relacionamentos da ontologia *O4C*, denominada *ObasCId*. Por fim, a ferramenta computacional de apoio à abordagem *ObasCId*, denominada *ObasCId-Tool*, é apresentada no Capítulo 5.

A apresentação dos estudos experimentais conduzidos sobre a abordagem *ObasCId* e sobre a ferramenta *ObasCId-Tool*, bem como do planejamento desses estudos com base no modelo GQM (*Goal-Question-Metrics*) (Basili e Rombach, 1994) estão contidos no Capítulo 6. Finalmente, no Capítulo 7 são discutidas as principais contribuições e limitações deste trabalho, destacando-se também as propostas para sua continuidade.

No Apêndice A é apresentado o documento de requisitos da ferramenta *ObasCId-Tool* e no Apêndice B estão os materiais de apoio (questionários, roteiro de atividades, documentos de requisitos, entre outros) utilizados para condução dos estudos experimentais descritos no Capítulo 6.

Capítulo 2

FUNDAMENTAÇÃO TEÓRICA

2.1 Considerações Iniciais

Os principais temas relacionados a esta pesquisa são “Engenharia de Requisitos Orientada a Aspectos” e “Ontologias de Domínio”, uma vez que foram utilizados conceitos de ontologias para tornar a atividade de identificação e classificação de interesses em requisitos de software mais efetiva.

A revisão da literatura realizada foi conduzida por meio de Mapeamentos Sistemáticos (MS) da literatura (Petersen *et al*., 2008). O primeiro MS (Parreira Júnior e Penteado, 2014; Parreira Júnior e Penteado, 2015d) permitiu a identificação e a catalogação das principais abordagens para EROA disponíveis na literatura. O objetivo da realização deste MS foi prover subsídios teóricos para a definição do problema de pesquisa deste doutorado, bem como da proposta de solução para o mesmo. O segundo MS (Parreira Júnior e Penteado, 2015a), por sua vez, permitiu conhecer as principais formas de uso das ontologias no contexto da ER, bem as abordagens relacionadas a elas.

Neste capítulo são apresentados os principais conceitos relacionados à EROA (Seção 2.2), bem como a descrição de algumas abordagens para EORA que contribuíram para a construção da solução proposta neste doutorado. Além disso, alguns dos principais conceitos relacionados às ontologias de domínio, bem como à utilização das mesmas no contexto da Engenharia de Requisitos (Seção 2.3) são apresentados. Na Seção 2.4, o desenvolvimento de ontologias de domínio é discutido, com o intuito de prover a fundamentação teórica necessária para entendimento do Capítulo 3, que refere-se à ontologia *O4C*. Por fim, na Seção 2.5 são apresentadas as considerações finais deste capítulo.

2.2 Engenharia de Requisitos Orientada a Aspectos - EROA

A Engenharia de Requisitos (ER) engloba atividades relacionadas à coleta e à estruturação de informações, também conhecidas como *Elicitação* e *Análise de Requisitos*. Cada subatividade (ou tarefa) realizada durante a obtenção de requisitos resultará em um documento textual com a descrição de todos os requisitos que o software deve contemplar (Chitchyan *et al*., 2006). Esse documento é então analisado durante a atividade de *Análise de Requisitos* e os requisitos são estruturados em unidades individuais como pontos de vista, objetivos, casos de uso, cenários, entre outros, dependendo da abordagem para ER utilizada. Isso é realizado a fim de se promover a identificação e a modularização dos interesses do software.

Abordagens tradicionais da ER como pontos de vista, objetivos, casos de uso, cenários, foram criadas com base no princípio da separação de interesses, porém, certos interesses de escopo amplo, também conhecidos como interesses transversais, não são fáceis de serem modularizados e mantidos separadamente durante o ciclo de desenvolvimento do software. Nesse sentido, Moreira *et al*. (2005) afirmam que uma abordagem efetiva para ER deve conciliar a necessidade de separar os interesses com a necessidade de atender às restrições dos interesses transversais.

O fato de existirem interesses que são difíceis de serem isolados em módulos individuais motivou a realização de pesquisas relacionadas à Engenharia de Requisitos Orientada a Aspectos (EROA). Nesta seção são apresentadas com mais detalhes algumas das principais abordagens para EROA, com a descrição de suas respectivas atividades, pontos fortes e limitações. Além disso, essas abordagens são discutidas com base nas quatro principais atividades da EROA apresentadas no Capítulo 1 desta tese, são elas: “Identificação e Classificação de Interesses”, “Representação de Interesses”, “Composição de Interesses” e “Detecção e Análise de Conflitos”.

Para facilitar a identificação dessas abordagens no restante do texto desta tese, seus nomes foram transformados em siglas, conforme apresentado no Quadro 2.1. Ainda nesse quadro, são apresentadas as referências para as publicações nas quais as abordagens foram propostas, bem como o número das seções em que tais abordagens são apresentadas com mais detalhes neste texto.

As abordagens do Quadro 2.1 são apresentadas neste trabalho, pois: (i) contemplam algum tipo de conceito utilizado para construção da ontologia de domínio *O4C* (Capítulo 3), bem como da abordagem *ObasCId* e da ferramenta *ObasCId-Tool* (Capítulos 4 e 5); e (ii) têm se mostrado robustas e maduras, tendo sido analisadas em

recentes estudos comparativos (Parreira Júnior e Penteado, 2013; Herrera *et al*., 2012; Singh e Gill, 2011; Bombonatti e Melnikoff, 2010; Rashid e Chitchyan, 2008; Sampaio *et al*., 2007; Bakker *et al*., 2005; Chitchyan *et al*., 2005), inclusive por meio de estudos experimentais com softwares reais (Herrera *et al*., 2012; Sampaio *et al*., 2007).

Quadro 2.1. Abordagens para EROA descritas neste trabalho.

Sigla	Descrição	Seção
MDSoC	Separação Multidimensional de Interesses na Engenharia de Requisitos (*Multi-Dimensional Separation of Concerns in Requirements Engineering*) Moreira *et al*., 2005	2.2.1
Theme	Abordagem *Theme* *Clarke e Baniassad, 2005; Baniassad e Clarke, 2004*	2.2.2
EA-Miner	Abordagem *EA-Miner* Chitchyan *et al*., 2006; Sampaio *et al*., 2005	2.2.3
EROA/XML	Uma Abordagem baseada em XML para Especificação e Composição de Interesses Aspectuais Soeiro *et al*., 2006	2.2.4
EROA/Arcade	Engenharia de Requisitos Orientada a Aspectos com Arcade Rashid *et al*., 2003; Rashid *et al*., 2002	2.2.5

2.2.1 *Separação Multidimensional de Interesses na Engenharia de Requisitos* (*MDSoC - Multi-Dimensional Separation of Concerns in Requirements Engineering*)

Esta abordagem propõe que interesses devem ser decompostos de forma uniforme com relação a sua natureza funcional, não funcional ou transversal (Moreira *et al*., 2005). Tratando todos os interesses da mesma forma, pode-se então escolher qualquer conjunto de interesses como base para analisar a influência dos outros interesses sobre essa base.

Identificação, Classificação e Representação de Interesses. Esta atividade leva em consideração que certos interesses, como por exemplo, “Mobilidade”, “Recuperação de Informação”, “Persistência”, entre outros aparecem frequentemente durante o desenvolvimento de software. Assim, os autores dividiram o espaço de interesses em dois: (i) o dos metainteresses, que consiste em um conjunto abstrato de interesses típicos de serem encontrados, como os que foram mencionados acima; e (ii) o dos interesses do sistema, que contempla os interesses específicos do software sob análise.

Para se utilizar esta abordagem, os requisitos do software devem ser analisados pelo engenheiro de software e categorizados com base nos interesses existentes no espaço de metainteresses, gerando assim os interesses do sistema. Para representação dos interesses, tanto os metainteresses quanto os do sistema, arquivos XML são utilizados.

Na Listagem 2.1 – (A) encontra-se a definição do metainteresse “Mobilidade (*Mobility*)”. Essa definição inclui uma breve descrição do interesse, alguns exemplos de uso do mesmo, derivados de experiências prévias de utilização desse interesse, e os metainteresses que podem relacionar-se com ele.

```
<?xml version="1.0" ?>
<MetaConcern name="Mobility">
  <Description>The quality of moving freely </Description>
  <Examples>Wireless networks, Mobile phones</Examples>
  <Relationships> Availability, Portability, Context </Relationships>
</MetaConcern>
```

(A)

```
<?xml version="1.0" ?>
<Concern name="Mobility">
  <Requirement id="1">The system will be accessed on the move.</Requirement>
</Concern>
```

(B)

Listagem 2.1. Interesse “Mobilidade (*Mobility*)” representado no espaço dos metainteresses (A) e no espaço dos interesses do sistema (B) (adaptado de Moreira *et al*., 2005).

A Listagem 2.1 – (B) ilustra a especificação do interesse “Mobilidade (*Mobility*)”, com base no documento de requisitos de um software de guia turístico sensível ao contexto apresentado por Moreira *et al*. (2005). Neste caso, o engenheiro de software deve especificar os requisitos do software relacionados ao interesse em questão.

Composição de Interesses. Após a identificação e representação dos interesses do software, regras de composição são definidas para se especificar como um determinado interesse influencia outros requisitos ou interesses do software. As regras de composição também são especificadas por meio de arquivos XML. Na Listagem 2.2 é apresentado um exemplo de regra de composição na qual o interesse “Mobilidade (*Mobility*)” afeta todos os requisitos do interesse “Recuperação de Informação (*Information Retrieval*)” e o requisito de identificador “1” do interesse “Navegação (*Navigation*)”.

```
<?xml version="1.0" ?>
<Composition>
    <Requirement concern="Mobility" id="all">
       <Constraint action="affect" operator="on">
          ...
          <Requirement concern="Navigation" id="1" />
          <Requirement concern="InformationRetrieval" id="all" />
       </Constraint>
       ...
    </Requirement>
</Composition>
```

Listagem 2.2. Regra de composição para o interesse “Mobilidade (*Mobility*)” (adaptado de Moreira *et al*., 2005).

Detecção e Análise de Conflitos. Esta atividade é realizada a partir da observação das interações de um interesse com os outros interesses do software. Para se identificar os conflitos existentes entre dois interesses C_1 e C_2 de um software, deve-se analisar a *Interseção de Composição* $SC_1 \cap SC_2$.

$SC_1 \cap SC_2$ corresponde ao conjunto de interesses cujos requisitos estão entrelaçados com C_1 e C_2, simultaneamente. Por exemplo, na Listagem 2.2, nota-se que o interesse "Mobilidade (*Mobility*)" afeta o requisito "1" do interesse "Navegação (*Navigation*)". Supondo que o interesse "Recuperação de Informação (*Information Retrieval*)" também afete esse requisito, então a interseção de composição $SC_{\text{Recuperação de Informação}} \cap SC_{\text{Mobilidade}}$ consiste no conjunto {"Navegação"}.

A análise dos conflitos ocorre com base no tipo de contribuição que um interesse pode exercer sobre outro. Essas contribuições podem ser negativas (-), positivas (+) ou neutras. Uma matriz de contribuição é construída, de forma que cada célula apresenta o tipo da contribuição (+ ou -) dos interesses em questão com relação aos interesses do conjunto de interseções de composição localizado dentro da célula. Uma célula vazia denota a não existência de relacionamento entre os interesses.

Com relação à análise de conflitos entre os interesses de "Recuperação de Informação (*Information Retrieval*)" e "Mobilidade (*Mobility*)", tem-se o seguinte: "quanto mais o visitante se movimenta, maiores serão as dificuldades dele em recuperar informações no software". Isso significa que "Mobilidade (*Mobility*)" contribui negativamente para "Recuperação de Informação (*Information Retrieval*)" com relação à "Navegação (*Navigation*)". A contribuição na direção oposta também é negativa, uma vez que quanto mais complexa é a informação que precisa ser recuperada, menos móvel o software poderá ser, uma vez que algumas redes sem fio possuem tamanho de banda limitado. A Figura 2.1 ilustra a matriz de contribuição obtida a partir da análise do cenário anterior.

	Recuperação de Informação	Mobilidade
Recuperação de Informação		$\overline{\text{}}$ **{Navegação}**
Mobilidade	$\overline{\text{}}$ **{Navegação}**	

Figura 2.1. Exemplo de uma matriz de contribuição entre os interesses "Mobilidade" e "Recuperação de Informação" (adaptado de Moreira *et al*., 2005).

Como o efeito cumulativo da "Mobilidade" e "Recuperação de Informação" sobre "Navegação" é negativo, então há necessidade de se realizar uma análise, com o intuito de resolver o conflito existente. Os autores desta abordagem propõem a atribuição de pesos aos interesses que contribuem negativamente para outros interesses. Cada peso é um número real no intervalo [0..1] e representa a prioridade de um interesse com

relação aos outros. Esses valores são dados pelo engenheiro de software, de acordo com a importância de cada interesse em relação aos demais.

Pontos fortes e fracos. Como pontos fortes da abordagem *MDSoC*, tem-se que: (i) trata-se de uma abordagem para EROA que lida com interesses funcionais e não funcionais sob uma mesma perspectiva, o que pode favorecer a identificação de ITs relacionados a requisitos funcionais do software; (ii) propõe um arquivo XML (*template*) para documentação de interesses, destacando suas propriedades e relacionamentos com outros interesses; e (ii) lida com todas as principais atividades para EROA, apresentadas no Capítulo 1 desta tese.

Como limitações ou pontos fracos, tem-se que: (i) a criação e aplicação das regras de composição propostas parecem ser onerosas e propensas a erros; (ii) não há diretrizes que auxiliem os engenheiros de software durante a realização das atividades da abordagens, tornando o resultado dessas atividades fortemente dependente da experiência desses profissionais; e (iii) o *template* proposto pelos autores não deixa claro quais são os tipos de relacionamentos existentes entre diferentes interesses do software, como por exemplo, dependência, composição, contribuição, entre outros.

2.2.2 Abordagem *Theme*

A abordagem *Theme* (Clarke e Baniassad, 2005; Baniassad e Clarke, 2004) é dividida em dois níveis: (i) o de requisitos, denominado *Theme/Doc*, que fornece mecanismos para visualização dos interesses do software, bem como do relacionamento entre eles; e (ii) o de projeto, denominado *Theme/UML*, que permite ao engenheiro de software modelar os interesses base e transversais do software e especificar como eles podem ser combinados.

Identificação e Classificação de Interesses. Para esta atividade, o engenheiro de software dispõe do conceito de “visualização de ações”, um tipo de abstração para interesses de software proposto pelos autores da abordagem. Duas entradas são obrigatórias para se gerar uma visualização de ações: (i) uma lista de ações-chaves, isto é, verbos identificados pelo engenheiro de software ao analisar o documento de requisitos; e (ii) o conjunto de requisitos do software. Na Figura 2.2 é apresentada a visualização de ações criada a partir de um conjunto de requisitos e de uma lista de ações-chaves de um pequeno software de gerenciamento de cursos apresentado por Baniassad e Clarke (2004). As ações-chaves são representadas por losangos e os requisitos do texto, por caixas com bordas arredondadas.

Se um requisito contém uma ação-chave em sua descrição, então ele é associado a essa ação-chave por meio de uma seta da caixa com borda arredondada

para o losango correspondente à ação. A ideia é utilizar essa visualização para separar as ações e os requisitos do software em dois grupos: (i) o grupo "base", que é autocontido, ou seja, não possui requisitos que se referem a ações do outro grupo (equivale ao conceito de interesse base); e (ii) o grupo "transversal" que possui requisitos que se referem a ações do grupo base (equivale ao conceito de interesse transversal). Para atingir essa separação em grupos, o engenheiro de software deve examinar os requisitos e as ações do software.

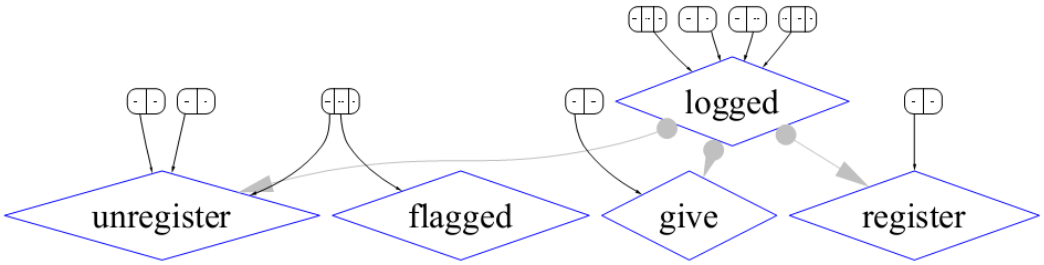


Figura 2.2. Exemplo de uma visualização de ações (Baniassad e Clarke, 2004).

Caso o engenheiro de software decida que uma ação principal entrecorta (ou está entrelaçada) as demais ações do requisito em questão, então uma seta com um ponto em uma de suas extremidades é traçada da ação que entrecorta para a ação que é entrecortada. Na Figura 2.2, denota-se que a ação "*logged*" entrecorta as ações "*unregister*", "*give*" e "*register*".

Representação e Composição de Interesses. Para essas atividades, utiliza-se o nível *Theme/UML* da abordagem *Theme*. Esse nível trabalha com o conceito de temas - elementos utilizados para representar interesses e que podem ser do tipo base ou transversal. Os temas base encapsulam as funcionalidades do domínio do problema, enquanto que os transversais encapsulam os interesses que afetam os temas base. A representação gráfica de um tema é um pacote da UML denotado com o estereótipo `<<theme>>`. Os temas transversais são representados por meio de gabaritos da UML. Um gabarito é representado graficamente por um pacote da UML com um parâmetro no canto superior direito.

Após a especificação do software em temas base e transversais, é necessário realizar a composição deles. Para isso utiliza-se o relacionamento de ligação (*bind*), que descreve para quais eventos ocorridos nos temas base o comportamento do tema transversal deve ser disparado. Na Figura 2.3 sãos apresentados um exemplo de tema base, denominado "*BankAccount"*, e um de tema transversal, "*Logging*". Além disso, a composição entre esses dois temas é representada graficamente pela linha tracejada que os vincula. Esse relacionamento de ligação especifica que a classe "*Account*", do tema base, deve ser associada à classe gabarito do tema "*Logging*", e consequentemente, ser afetada pelo comportamento transversal desse tema.

Detecção e Análise de Conflitos. Os trabalhos analisados sobre a abordagem *Theme* não apresentam detalhes sobre a realização dessa atividade.

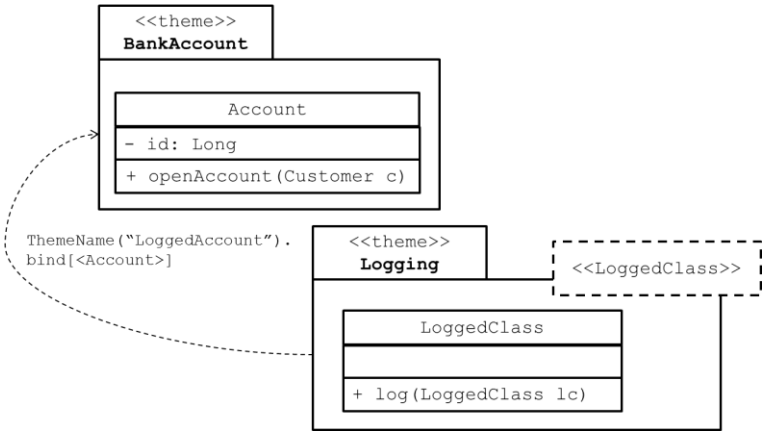


Figura 2.3. Composição entre os temas "*Logging*" e "*BankAccount*".

Pontos fortes e fracos. Como pontos positivos desta abordagem tem-se que ela: (i) provê recursos para identificação, representação e composição de interesses em nível de requisitos; e (ii) possui preocupação com a rastreabilidade dos interesses em fases distintas do desenvolvimento do software, como análise e projeto.

Como limitações, tem-se que a abordagem: (i) requer ampla intervenção do usuário em todas as suas atividades. Por exemplo, o usuário precisa analisar todos os requisitos do software e criar os elementos de modelagem apropriados, além de prover conjuntos de ações-chave para construção das visualizações de ação; (ii) depende fortemente do conjunto de palavras-chave elicitado pelo engenheiro de software para identificação e classificação dos interesses. Essa é uma tarefa onerosa, suscetível a erros e altamente dependente da experiência do engenheiro de software, o que pode influenciar diretamente a qualidade do produto gerado por esta abordagem.

2.2.3 Abordagem *EA-Miner*

A abordagem *EA-Miner* (Mineração de *Early-Aspects*) é executada a partir de uma suíte de ferramentas computacionais, que apoiam a execução das principais atividades da EROA (Chitchyan *et al*., 2006; Sampaio *et al*., 2005). Essas ferramentas exercem dois tipos de papéis: (i) o de **gerador de informações**: responsável por analisar os artefatos de entrada de uma determinada atividade e os complementarem com informações linguísticas, semânticas, estatísticas, entre outros; e (ii) o de **consumidor de informações**: que utiliza as informações adicionais atribuídas pelo gerador de informações para múltiplos tipos de análise.

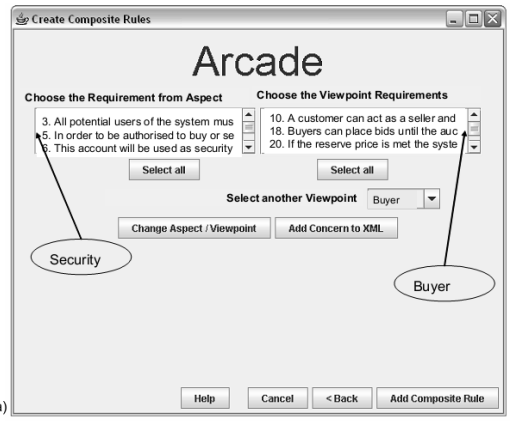
A principal ferramenta geradora de informações da abordagem *EA-Miner* é a WMATRIX (2015), uma aplicação *web* para Processamento de Linguagem Natural

(PLN). Ela é utilizada por essa abordagem para identificação de conceitos do domínio do software a partir de seus requisitos.

Identificação e Classificação de Interesses. É realizada pela ferramenta *EA-Miner*, que recebe o mesmo nome da abordagem apresentada nesta seção. Para identificação de interesses transversais não funcionais, *EA-Miner* utiliza o catálogo de requisitos não funcionais proposto por Chung e Leite (2000). Os interesses transversais são identificados pela equivalência semântica entre as palavras do documento de requisitos e as categorias desse catálogo. Para identificação de interesses transversais funcionais, *EA-Miner* utiliza uma estratégia semelhante à da abordagem *Theme*, detectando a ocorrência de verbos repetidos no documento de requisitos, o que pode sugerir a presença de interesses transversais funcionais.

Representação e Composição de Interesses. Para esta atividade, a ferramenta ARCADE (*Aspectual Requirements Composition And Decision*) é utilizada. Com ela, o engenheiro de software pode selecionar quais requisitos são afetados pelos interesses do software, escolher os relacionamentos existentes entre eles e, posteriormente, gerar regras de composição. ARCADE utiliza a mesma ideia de regra de composição da abordagem *MDSoC* (Moreira *et al*., 2005), apresentada na Seção 2.2.1.

A interface da ferramenta ARCADE é apresentada na Figura 2.4. Por meio dela, o engenheiro de software seleciona quais requisitos do interesse “Segurança (*Security*)” estão entrelaçados com requisitos do interesse “Comprador (*Buyer*)” (Figura 2.4 – A). Posteriormente, ele pode escolher os relacionamentos existentes entre esses requisitos e gerar regras de composição, conforme pode ser visto na Figura 2.4 – (B).



(a)

```
<Composition>
    <Requirement aspect=“Security” id=“3”>
        <Constrain action=“enforce” operator=“with”>
            <Requirement viewpoint =“Buyer” id=“10”/>
        </Constrain>
        <Outcome action=“fullfilled”/>
</Requirement>
```

(b) ....

Figura 2.4. Composição de interesses na ferramenta ARCADE (Chitchyan *et al*., 2006).

Detecção e Análise de Conflitos. ARCADE possui também um componente analisador de conflitos, o qual identifica sobreposição entre interesses com relação aos requisitos que eles afetam. O engenheiro de software é alertado sobre essa sobreposição e decide se os aspectos sobrepostos prejudicam ou favorecem um ao outro.

Pontos fortes e fracos. Como pontos fortes da abordagem *EA-Miner*, tem-se que: (i) os autores propõem uma suíte de ferramentas que apoiam o engenheiro de software durante a execução das atividades deste processo; (ii) lida com todas as principais atividades da EROA; e (iii) a maneira como as ferramentas foram combinadas (por meio de geradores e consumidores de informações) garante flexibilidade ao processo de EROA, permitindo que ele não fique dependente de um tipo de ferramenta apenas.

Como limitações, tem-se que: (i) *EA-Miner* baseia-se na WMATRIX, que é específica para processamento de textos da língua inglesa; não foi relatado pelos autores qual seria o grau de esforço necessário para utilizar essa ferramenta em outros contextos, como por exemplo, no caso de documentos de requisitos escritos em outros idiomas; e (ii) como várias ferramentas são utilizadas na suíte proposta e essas ferramentas foram desenvolvidas por diferentes autores e em diferentes momentos, há algumas inconsistências entre nomes e conceitos nas interfaces das mesmas. Por exemplo, na ferramenta *EA-Miner* o termo “Early-Aspects” é utilizado para representar interesses transversais, enquanto que na ferramenta ARCADE, utiliza-se o termo “Aspects”. Isso pode vir a confundir o usuário durante a execução da abordagem proposta.

2.2.4 EROA/XML - Uma Abordagem baseada em XML para Especificação e Composição de Interesses Aspectuais

A abordagem EROA/XML contempla as atividades descritas a seguir:

Identificação e Classificação de Interesses. Ocorre por meio da análise da descrição do software feita por parte do engenheiro de software. Os autores indicam que a identificação dos interesses pode ser auxiliada pelo uso de catálogos de requisitos não funcionais, como o proposto por Chung e Leite (2000). Para cada entrada do catálogo, deve-se decidir se o interesse em questão existe ou não no software em análise.

Representação e Composição de Interesses. Para essas atividades foram criados arquivos XML específicos, com o intuito de coletar e organizar todas as

informações a respeito dos interesses do software. A estrutura e explicação desse arquivo são apresentadas a seguir:

- *Name*: define um identificador para o interesse;
- *Description*: apresenta uma breve descrição do comportamento do interesse;
- *Classification*: indica se o interesse é funcional ou não funcional;
- *ListOfSources*: lista as fonte de informação que contribuíram para a criação do interesse. Uma lista de três valores foi criada: *stakeholders*[5], documentos de domínio e catálogos;
- *ListOfPrioritiesByStakeholder*: especifica uma lista na qual cada elemento agrega um *stakeholder* juntamente com sua prioridade para um determinado interesse. Essa prioridade reflete o grau de importância daquele interesse para o *stakeholder*. Esse atributo possui quatro valores: Muito Importante, Importante, Médio, Baixo e Muito Baixo;
- *ListOfResponsabilities*: lista a informação sobre o que o interesse deve realizar;
- *ListOfContributions*: define o relacionamento de contribuição entre o interesse em questão e outros interesses. Esta contribuição pode ser positiva (+), negativa (-) ou "*don't care*", significando que um interesse pode não ter uma contribuição explícita com outros interesses; e
- *ListOfRequiredConcerns*: define uma lista de outros interesses dos quais o interesse atual depende.

A composição dos interesses do software ocorre por meio de regras de composição, que consistem dos seguintes elementos:

- *Term*: pode ser um interesse ou outra regra de composição;
- *Operator*: define o tipo de operação de composição, que pode ser >>, [> ou ||. C1 >> C2 consiste em uma composição sequencial e significa que o comportamento de C2 inicia-se se e somente se C1 tiver terminado com sucesso. C1 [> C2 significa que C2 interrompe o comportamento de C1 quando começa a executar. C1 || C2 significa que o comportamento de C1 está sincronizado com o de C2; e
- *Outcome*: expressa o resultado das restrições impostas pelos operadores comentados anteriormente. Ela pode ser: i) *satisfied*: utilizada para declarar que outro interesse deve ser contemplado após a aplicação da regra de

[5] São todos aqueles interessados no software a ser desenvolvido, afetando ou sendo afetados por ele (Sommerville, 2011).

composição; e ii) *fulfilled*: utilizado para declarar que nada precisa ser feito após a aplicação da regra de composição.

Detecção e Análise de Conflitos. Os trabalhos analisados sobre essa abordagem não apresentam detalhes sobre a realização desta atividade.

Pontos fortes e fracos. O principal ponto positivo desta abordagem em relação às demais é que o arquivo XML (*template*) utilizado para representação dos interesses apresenta propriedades/elementos importantes dos interesses, não encontrados nas demais abordagens, tais como a lista de fontes de um interesse e a separação entre relacionamentos de contribuição e dependência entre interesses, entre outros.

Como ponto negativo da abordagem *EROA/XML*, tem-se que ela apenas propõe que sejam utilizadas técnicas convencionais para identificação de interesses de software, como por exemplo, a análise manual do documento de requisitos com o auxílio de catálogos de requisitos não funcionais, sem apresentar diretrizes para que isso seja realizado. Isso faz com que os resultados esperados dessa abordagem sejam altamente dependentes da experiência dos usuários que a aplicam.

2.2.5 EROA/Arcade - Engenharia de Requisitos Orientada a Aspectos com Arcade

Rashid *et al*. (2003) e Rashid *et al*. (2002) propuseram uma abordagem para EROA baseada em pontos de vista (*viewpoints*). Nesta abordagem, são utilizados arquivos XML para especificação dos pontos de vista, dos interesses não funcionais e das regras de composição entre pontos de vista e interesses do software. Além disso, a ferramenta ARCADE, comentada na Seção 2.2.3, também é utilizada por essa abordagem para automatizar a tarefa de representação dos conceitos descritos anteriormente.

Identificação, Classificação e Representação de Interesses. A identificação dos pontos de vista e interesses não funcionais é realizada por meio da análise dos requisitos do software pelo engenheiro de software. Uma vez identificados, eles são especificados em arquivos XML, semelhantes àqueles apresentados na abordagem *MDSoC* (Seção 2.2.1). Após a identificação dos pontos de vista e dos interesses não funcionais do software, define-se quais desses interesses são candidatos a interesses transversais. Para isso cria-se uma matriz de relacionamentos, na qual os interesses do software são colocados em suas linhas e os pontos de vista, em suas colunas (Quadro 2.2).

Cada célula dessa matriz, quando marcada, representa que um determinado interesse exerce influência sobre os requisitos do ponto de vista da coluna correspondente àquela célula. Sendo assim, é possível observar quais pontos de vista são entrecortados pelos interesses do software. Segundo os autores, quando um interesse entrecorta os requisitos de vários pontos de vista, isso pode indicar que se trata de um interesse transversal. No exemplo do Quadro 2.2, apresentado por Rashid *et al*. (2003), o interesse "Disponibilidade" está entrelaçado com requisitos dos interesses "Pagamento" e "Cadastramento", enquanto que o interesse "Tempo de Resposta" está entrelaçado apenas com "Pagamento".

Quadro 2.2. Trecho de uma matriz de relacionamentos (adaptada de Rashid *et al*., 2003).

Interesses não funcionais/ Pontos de Vista	**Pagamento**	**Cadastramento**	**...**
Tempo de resposta	X		...
Disponibilidade	X	X	...
...	...	...	...

Composição de Interesses e Detecção e Análise de Conflitos. Após a identificação dos candidatos a interesses transversais e dos pontos de vista do software ter ocorrido, os mesmos devem ser compostos por meio de regras de composição. Posteriormente, a detecção e análise de conflitos deve ser realizada. A definição das regras de composição e da atividade de detecção e resolução de conflitos segue a mesma ideia da abordagem *MDSoC* (Moreira *et al*., 2005) e por esse motivo não será comentada novamente.

Pontos fortes e fracos. O principal ponto positivo desta abordagem é a apresentação de um novo tipo de recurso para identificação de interesses transversais, a saber, a matriz de relacionamento.

Como pontos negativos, têm-se: (i) embora haja o apoio computacional ARCADE, muitas atividades onerosas continuam sem automatização, como por exemplo, a identificação de interesses a partir dos requisitos do software; (ii) além da baixa automatização, algumas das atividades desta abordagem são altamente dependentes da experiência do engenheiro de software com relação aos conceitos de interesses transversais, como por exemplo, identificação de interesses e definição da influência desses interesses sobre outros; e (iii) a matriz de relacionamentos proposta não contempla a situação em que um interesse funcional (tratado como ponto de vista) seja transversal.

2.2.6 Considerações finais sobre as abordagens para EROA

Algumas das limitações das abordagens apresentadas nesta seção podem ser as fontes para as causas da baixa efetividade proporcionada por essas abordagens quanto à identificação e classificação de interesses em requisitos de software, conforme foi comentado no Capítulo 1 desta tese. Nesta seção, a discussão sobre essas causas é retomada à luz dos trabalhos analisados neste capítulo, bem como daqueles obtidos a partir do MS sobre EROA conduzido neste doutorado (Parreira Júnior e Penteado, 2014; Parreira Júnior e Penteado, 2015d).

Causa 1: falta de compreensão a respeito do domínio dos interesses de software. Na maioria das abordagens para EROA existentes na literatura, em especial naquelas que lidam com a identificação e classificação de interesses, o conhecimento sobre o domínio dos interesses de software é documentado em arquivos XML (*templates*) específicos, desenvolvidos pelos autores dessas abordagens. Tais arquivos geralmente não são acompanhados do meta-modelo que deu origem aos mesmos, nem compartilham entre si os principais conceitos e relacionamentos existentes no domínio dos interesses de software. Por exemplo, o arquivo proposto por Moreira *et al*. (2005) não contempla como informação de um interesse, a fonte que deu origem a ele, tais como um *stakeholder*, um documento de negócio, entre outros. Contudo, essa informação pode ser encontrada em *templates* de outras abordagens para EROA (Agostinho *et al*., 2008; Whittle e Araújo, 2004; Brito e Moreira, 2003). Em alguns casos, há divergências entre os termos utilizados para representar um mesmo conceito. Por exemplo, no conjunto de ferramentas proposto para a abordagem *EA-Miner* (Chitchyan *et al*., 2006; Sampaio *et al*., 2005), em alguns casos ITs são tratados como "*Early-Aspects*" e em outros casos, como aspectos ("*Aspects*").

Causa 2: escassez de recursos apropriados para apoiar engenheiros de software durante a identificação e classificação dos interesses do software. Muitas abordagens para EROA, tais como *Theme/Doc* (Clarke e Baniassad, 2005; Baniassad e Clarke, 2004), baseiam-se apenas na experiência dos engenheiros de software para a correta identificação e classificação dos interesses do software, o que pode levar à diminuição da precisão e da cobertura dessas abordagens. Em outras abordagens, esta atividade é apoiada por meio do uso de catálogos de requisitos não funcionais (Chernak, 2012; Zheng *et al*., 2010; Alencar *et al*., 2010; Mussbacher *et al*., 2010; Agostinho *et al*., 2008; Chitchyan *et al*., 2006; Soeiro *et al*., 2006; Moreira *et al*., 2005; Whittle e Araújo, 2004; Brito e Moreira, 2003), tais como os propostos por Cysneiro (2015), Chung e Leite (2000) e Boehm e In (1996). Entretanto, essa estratégia não é

adequada para o contexto da EROA, pois esses catálogos: (i) não são preparados para o domínio dos interesses de software, deixando de considerar algumas propriedades inerentes a esse domínio; e (ii) são descritos por meio de notações específicas para requisitos de software, tais como SIG (*Softgoal Interdepedency Graphs*), o que torna difícil a leitura e o entendimento desses catálogos por parte de engenheiros de software não especialistas nestas notações. Além disso, apesar de sugerirem o uso desses catálogos durante a identificação e classificação de interesses, tais abordagens não apresentam processos ou diretrizes para que os mesmos possam ser utilizados corretamente neste processo.

2.3 Ontologias de Domínio no Contexto da Engenharia de Requisitos

"Ontologia" é uma palavra de origem latina, cujo significado pode ser entendido como "o estudo da existência". Ela tem sido utilizada na filosofia para designar tanto uma disciplina – quando escrita com "O" maiúsculo – quanto um sistema de categorias independente de linguagem, apropriado para conceituação de teorias científicas (Guizzardi, 2005).

O primeiro relato de uso deste termo na computação data de 1967 por G. H. Mealy, em seu trabalho sobre os fundamentos da modelagem de dados, na área de processamento de dados (Mealy, 1967 *apud* Guizzardi, 2005). Desde então, ontologias têm sido aplicadas em diversas áreas, como Sistemas de Informação, Engenharia de Software, Inteligência Artificial e Web Semântica, porém com diferentes significados e propósitos. Nas duas primeiras áreas, o termo ontologia é comumente utilizado em conformidade com seu significado na Filosofia, ou seja, como um sistema de categorias independente de linguagem. Em contrapartida, em outras áreas como Inteligência Artificial e Web Semântica, tal termo é utilizado para designar um artefato concreto, projetado para um propósito específico, e representado em uma linguagem específica.

A definição ontologia utilizada pela maioria dos trabalhos apresentados nesta seção é "uma especificação formal e explícita de uma conceitualização compartilhada" (Gruber, 1995). Fensel (2001) *apud* Hernandes (2009) interpreta a definição de Gruber da seguinte maneira:

- **Conceitualização**: uma ontologia é uma visão simplificada e abstrata do domínio que se deseja representar, portanto, ela contempla os conceitos para alguma área de interesse e os relacionamentos existentes entre eles;
- **Explícita**: indica que os conceitos e relacionamentos tratados por uma ontologia devem ser definidos de maneira explícita;
- **Formal**: ontologias devem ser codificadas em uma linguagem formal para evitar ambiguidades e serem passíveis de processamento automático; e
- **Compartilhada**: refere-se ao fato de que uma ontologia capta o conhecimento consensual, não se restringindo a um indivíduo em si.

Os principais componentes de uma ontologia são: (i) **indivíduos**: também conhecidos como "instâncias" ou "instâncias de classes", indivíduos representam objetos no domínio de interesse; (ii) **propriedades**: são relações binárias que interligam dois indivíduos, duas classes, um indivíduo e um valor ou uma classe e um valor; (iii) **classes**: são representações concretas de um conceito e reúnem indivíduos que satisfazem seu conjunto de propriedades (Horrige *et al*., 2011; Hernandes, 2009; Lima e Carvalho, 2005); e (iv) **axiomas**: sentenças sempre verdadeiras que especificam restrições ou definições de conceitos derivados em uma ontologia.

A partir dos resultados de um MS conduzido neste doutorado (Parreira Júnior e Penteado, 2015a), notou-se que há seis principais formas de uso de ontologias no contexto da Engenharia de Requisitos. Estas formas de uso são descritas nesta seção, juntamente com os principais trabalhos relacionados a elas.

2.3.1 Uso de ontologias de domínio para verificação de requisitos

Os trabalhos desta categoria utilizam o formalismo proporcionado pelas ontologias de domínio para identificar defeitos de omissão, ambiguidade, contradições na descrição dos requisitos de um software. Por exemplo, em Kaiya e Saeki's (2006), os autores desenvolveram uma ontologia com dois principais conceitos, "*Concept*" e "*Relationship*", que possuem, por sua vez, diversos subconceitos (Figura 2.5). Além disso, "*Relationship*" e "*Concept*" estão associados por meio de um relacionamento que conecta uma instância de um "*Relationship*" a duas instâncias de "*Concept*".

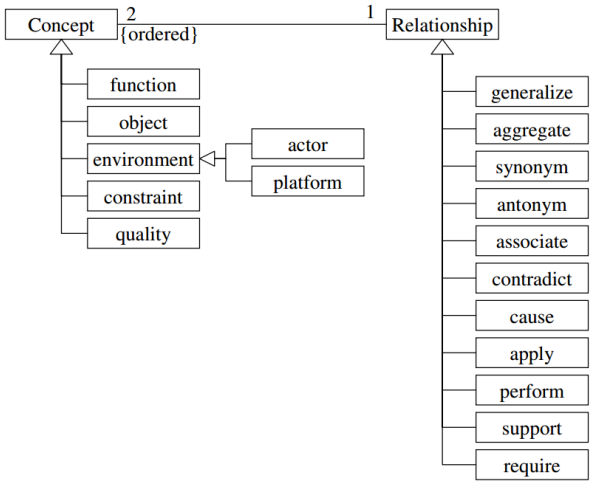


Figura 2.5. Ontologia para o domínio de requisitos de software (Kaiya e Saeki's, 2006).

Com base nesta ontologia, o engenheiro de software deve analisar os requisitos do software e mapeá-los para os conceitos e relacionamentos da ontologia proposta. A partir daí, uma série de regras de inferência são aplicadas para detectar inconsistências, incompletudes, ambiguidades e contradições no documento de requisitos. Por exemplo, a regra descrita no Quadro 2.3 especifica que se há um conceito "X" presente no documento de requisitos e esse conceito depende de outro conceito "Y", então "Y" também deve estar no documento de requisitos.

Quadro 2.3. Exemplo de regra de inferência (Kaiya e Saeki's, 2006).

$\forall s \forall x \forall y \cdot (InSpec(x,s) \wedge require(x,y) \rightarrow InSpec(y,s))$

Dzung e Ohnishi (2012) propuseram uma ideia bem parecida com a de Kaiya e Saeki's, divergindo principalmente nos tipos de conceitos e relacionamentos adotados em sua ontologia de domínio.

2.3.2 Extração de conhecimento a partir de documentos de requisitos com base em ontologias de domínio

Os trabalhos desta categoria descrevem métodos para se extrair conhecimento de domínio a partir de documentos de requisitos pré-existentes. A motivação comum para os trabalhos desta categoria consiste em obter ontologias que possam ser reutilizadas no processo de elicitação e análise de requisitos de projetos futuros.

Bargui *et al*. (2011) propuseram um conjunto de heurísticas para extrair conceitos relacionados a tomadas de decisões a partir de diagramas de processos de negócio e criar instâncias de uma ontologia de domínio para tomada de decisão.

Inicialmente, os autores propuseram uma ontologia para o domínio de tomada de decisões, conforme pode ser visualizada na Figura 2.6.

Posteriormente, os autores criaram um processo para extração de conhecimento a partir de diagramas do tipo *Business Process Model and Notation* (BPMN), que é composto de três atividades: extração, comparação e aprimoramento.

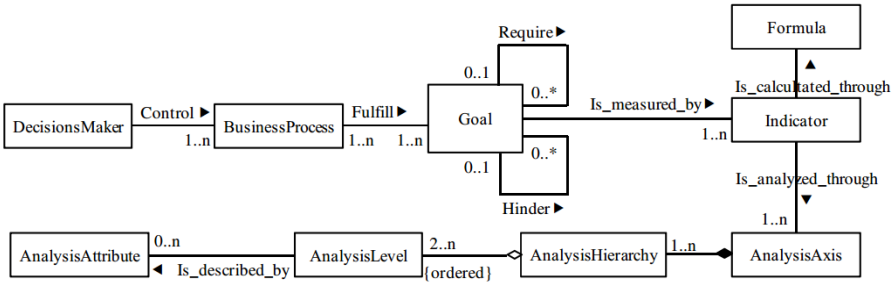


Figura 2.6. Ontologia para tomadas de decisão (Bargui *et al*., 2011).

Na primeira atividade, o engenheiro de software faz uso de um conjunto de seis heurísticas criadas pelos autores para extrair conceitos dos diagramas BPMN e classificá-los de acordo com os conceitos da ontologia proposta. Um exemplo de heurística é: “o nome do processo especificado no diagrama BPMN é identificado como um conceito do tipo *BusinessProcess* da ontologia proposta”.

Como resultado dessa atividade, tem-se uma lista de instância dos conceitos da ontologia para o domínio de tomada de decisão. Durante a atividade de comparação, esses conceitos são analisados com o intuito de identificar equivalências e problemas de ambiguidade entre os mesmos. Por fim, na atividade de aprimoramento, um conjunto de regras é aplicado para resolver os possíveis problemas encontrados na atividade anterior. Um exemplo de regra é que “se dois conceitos identificados pelo engenheiro de software são homônimos (significam ‘coisas’ diferentes, mas foram descritos com o mesmo termo), então um deles deve ser renomeado e os dois devem ser adicionados à ontologia”.

Os autores salientam que estas instâncias de conceitos identificados e refinados por meio do processo proposto servem de base para elicitação de requisitos de software em projetos futuros, relacionados com o domínio de tomada de decisão.

2.3.3 Uso de ontologias de domínio para guiar a elicitação e análise de requisitos

Os trabalhos desta categoria se concentram na ideia de que os conceitos disponíveis em uma ontologia, bem como os relacionamentos entre eles, são recursos importantes para auxiliar o engenheiro de software durante o processo de elicitação e

análise de requisitos (*e. g*. evitando o esquecimento de conceitos importantes, permitindo o melhor entendimento do relacionamento entre requisitos, agrupando requisitos com a mesma finalidade, entre outros).

Shibaoka *et al*. (2007) propõem uma extensão do trabalho de Kaiya e Saeki's (2006), na qual utiliza-se a ontologia desenvolvida por esses autores como base para um processo de elicitação de requisitos. Esse processo, denominado GOORE (*Goal-Oriented and Ontology Driven Requirements Elicitation*), é uma extensão do método para engenharia de requisitos baseada em objetivos (*Goal-Oriented Requirements Engineering - GORE*). Para utilizar o processo GOORE, inicialmente, deve-se ter: (i) uma representação dos requisitos do software por meio de um grafo de objetivos (Figura 2.7 - A); e (ii) uma instanciação da ontologia proposta em Kaiya e Saeki's (2006) para o domínio com o qual o software em análise está envolvido (Figura 2.7 - B). A partir desses dois artefatos, as palavras existentes no grafo de objetivos são mapeadas para os conceitos da instância de ontologia. Essa atividade pode ser realizada automaticamente, por meio de uma ferramenta computacional desenvolvida pelos autores da abordagem. Tal ferramenta usa técnicas de processamento de linguagem natural e um dicionário de sinônimos para realizar esse mapeamento.

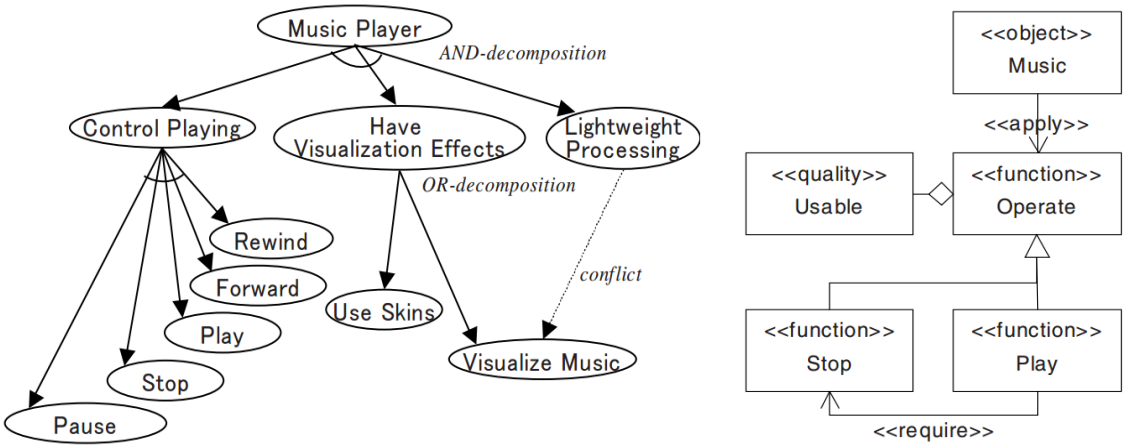


Figura 2.7. Elementos para utilização da abordagem GOORE (Shibaoka *et al*., 2007).

A partir desse mapeamento, um conjunto de regras de inferência, como aquela descrita no Quadro 2.3, são aplicadas sobre a instância de ontologia a fim de fornecer sugestões de novos objetivos a serem incorporados ao modelo inicial dos requisitos do software. Segundo os autores, o intuito é reduzir a dependência do processo de elicitação de requisitos com a abordagem orientada a objetivos da experiência dos engenheiros de software que a aplicam.

López *et al*. (2008) apresentam uma ontologia para compartilhamento e reuso de Requisitos Não Funcionais (RNF) e decisões de *design*, com base na descrição de

catálogos de RNF disponíveis na literatura. O pesquisador pode, então, criar instâncias dos conceitos dessa ontologia que contemplem os RNF e as decisões de *design* de seu interesse. Assim como no trabalho de Shibaoka *et al*. (2007), o trabalho de López *et al*. baseia-se no conceito da engenharia de requisitos orientada a objetivos. Contudo, enquanto Shibaoka *et al*. (2007) propõem o mapeamento dos termos do grafo de objetivos aos conceitos de sua ontologia de domínio, López *et al*., por sua vez, criaram uma ontologia de domínio para representar os conceitos da notação de grafos de objetivos, conforme pode ser vista na Figura 2.8.

Os autores sugerem que catálogos de RNF como os propostos por Cysneiro (2015) e Chung e Leite (2000) devem, então, ser convertidos em instâncias dessa ontologia. O intuito é propiciar um maior poder de processamento semântico sobre o conhecimento existente nos catálogos de RNF modelado em grafos de objetivos.

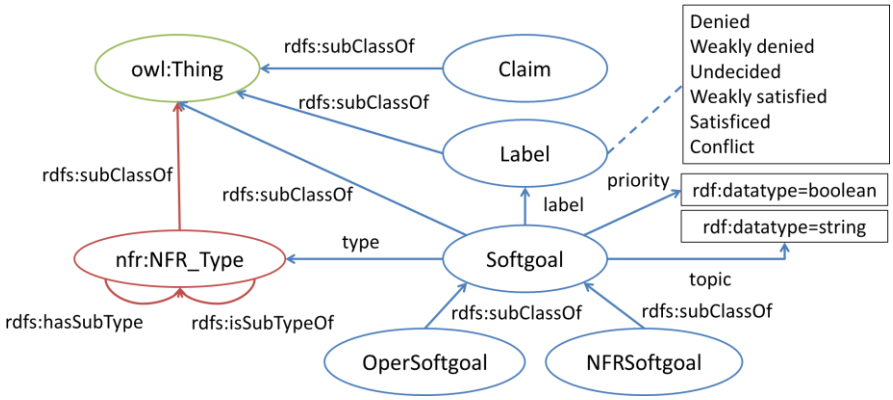


Figura 2.8. Ontologia para a especificação de requisitos por meio de grafos de objetivos (López *et al*., 2008).

2.3.4 Uso de ontologias como forma de estabelecer uma conceituação básica acerca do domínio de requisitos

Trabalhos que seguem esta linha procuram estabelecer uma compreensão clara a respeito dos conceitos e relacionamentos envolvidos no domínio de requisitos de software, tais como quais são as características de um requisito, como ele se relaciona com outros requisitos do software, dentre outras coisas. O propósito é facilitar a comunicação, o aprendizado e o uso desses conceitos em projetos futuros.

Neste sentido Nardi e Falbo (2006) propuseram uma ontologia de requisitos de software, com o intuito de facilitar a comunicação entre pessoas e agentes de software e permitir construção de ferramentas de apoio à Engenharia de Requisitos (ER) mais amplamente utilizáveis. Durante a construção dessa ontologia, várias outras ontologias acerca do domínio de engenharia de software foram analisadas e utilizadas, buscando-se reutilizar conceituações já estabelecidas.

Apesar de outros trabalhos, tais como Shibaoka *et al*. (2007) e Kaiya e Saeki's (2006) também apresentarem ontologias de domínio para requisitos de software, o enfoque destes trabalhos não era o mesmo do apresentado por Nardi e Falbo (2006), isto é, caracterizar o domínio de requisitos de software. Assim, muitos dos conceitos e relacionamentos existentes na ontologia proposta por Nardi e Falbo (2006) não são contemplados nestes trabalhos. Além disso, por ter utilizado uma abordagem para desenvolvimento de ontologias (*SABiO - Systematic Approach for Building Ontologies*), o trabalho de Nardi e Falbo (2006) apresentou uma ontologia mais bem definida, contemplando elementos importantes para uma ontologia de domínio, tais como seus requisitos e axiomas, que não aparecem nos trabalhos de Shibaoka *et al*. (2007) e Kaiya e Saeki's (2006).

2.3.5 Uso de ontologias de domínio para detecção e análise de conflitos entre requisitos

Os trabalhos desta categoria têm como enfoque o uso de ontologias para detecção e análise de conflitos entre requisitos de software. Esta categoria difere-se da categoria "Uso de ontologias para verificação de requisitos", pois dois requisitos podem estar corretos (não ambíguos, consistentes, entre outros), mas mesmo assim, conflitarem entre si.

Verma e Kass (2008) propõem uma abordagem para identificação de dependências e conflitos entre requisitos utilizando instâncias de uma ontologia de domínio. Para isso, os autores propõem uma ontologia para o domínio de requisitos de software que contempla conceitos como "tipo do requisito", os componentes de um requisito, tais como "agente", "ação", entre outros. Além disso, os principais tipos de requisitos não funcionais, como segurança, desempenho, entre outros, e a influência existente entre eles são catalogados nesta ontologia. Segundo os autores, os tipos de requisitos não funcionais e a influência existente entre eles foram obtidas a partir de um catálogo de requisitos não funcionais proposto por Boehm e In (1996).

Após a elicitação dos requisitos do software, o engenheiro de software deve mapear os mesmos para os conceitos da ontologia de domínio proposta e aplicar consultas na linguagem SPARQL (linguagem para realização de consultas lógicas sobre o conhecimento representado em ontologias) para detectar as dependências e os conflitos existentes entre eles. Por exemplo, a consulta apresentada no Quadro 2.4 visa a identificar todos os requisitos que afetam um ao outro de alguma forma.

Quadro 2.4. Exemplo de consulta para detecção de conflito entre requisitos (Verma e Kass, 2008).

```
select ?req1, ?req2 where
{?req1 hasRequirementType ?type1 . ?req2 hasRequirementType ?type2 .
Affects domain ?type1 . Affects range ?type2 .
?req2 hasAgent ?agent2 .
?req1 hasAgent ?agent1 filter( ?agent1 = ?agent2)})
```

2.3.6 Uso de ontologias de domínio para promover a rastreabilidade entre requisitos e outros tipos de abstrações de software

Esta categoria comporta trabalhos que apresentam o uso de ontologias de domínio como um recurso para promover a rastreabilidade entre requisitos e outros tipos de abstrações de um software, tais como modelos do tipo entidade-relacionamento, casos de uso, pontos de vista, entre outros. A ideia é confeccionar ontologias de domínio para diferentes tipos de abstração de software e realizar, posteriormente, o mapeamento entre os conceitos dessas ontologias. Assim, é possível rastrear elementos de um tipo de abstração em uma ou mais abstrações de outro tipo.

Neste contexto, Assawamekin (2011) aplica técnicas de processamento de linguagem natural e uma abordagem baseada em regras para rastrear requisitos em modelos do tipo entidade-relacionamento. Para isso, o autor propõe uma ontologia para o domínio de requisitos, que contempla também conceitos da abordagem entidade-relacionamento. Assim, dados um documento de requisitos e um diagrama ER, termos pré-existentes nestes artefatos devem ser inicialmente mapeados para os conceitos essa ontologia. Uma vez feito isso, um algoritmo desenvolvido pelos autores identifica tipos de relações semânticas e de rastreabilidade entre os conceitos da ontologia. Com essa ideia é possível, por exemplo, dizer se o atributo “nome de um funcionário” na descrição de um requisito equivale ao atributo “nome” de uma entidade “enfermeira” no modelo entidade-relacionamento.

2.3.7 Considerações finais sobre o uso de ontologias de domínio no contexto da engenharia de requisitos

No MS conduzido neste trabalho sobre o uso de ontologias no contexto da ER (Parreira Júnior e Penteado, 2015a), constatou-se que diversas abordagens baseadas em ontologias de domínio têm sido propostas para essa área, contudo, nenhuma delas é específica para o contexto da EROA. Assim, esses trabalhos não contemplam características específicas dos interesses de software, tais como sua classificação como

interesses não funcionais ou funcionais, o seu relacionamento com palavras-chave, sua decomposição em subinteresses, entre outros.

Outra limitação dos trabalhos analisados é que muitos deles apresentam ontologias para o domínio de requisitos sem fazer menção à abordagem utilizada para desenvolvimento dessas ontologias, nem ao modo como os trabalhos relacionados foram utilizados para obtenção dos conceitos das mesmas. Na maioria dos casos, os estudos sugerem que o conhecimento de especialistas foi utilizado para confecção dos conceitos representados nestas ontologias e que o desenvolvimento da ontologia foi realizado de forma *ad-hoc*. Essa situação acaba por gerar diversidades entre propostas similares, deixando de contemplar uma das características de uma ontologia de domínio, que é representar uma conceitualização compartilhada sobre um determinado domínio.

Por exemplo, os trabalhos de Kaiya e Saeki's (2006) e Dzung e Ohnishi (2012) propõem ontologias para um mesmo domínio, a saber, o de requisitos de software e para uma mesma finalidade, isto é, a verificação de requisitos. Contudo, é possível encontrar tipos de conceitos a mais e/ou a menos nos diferentes trabalhos; no trabalho de Kaiya e Saeki's (2006), apresenta-se alguns tipos de relacionamento entre requisitos, tais como "*support*", "*cause*", "*apply*", que não aparecem em Dzung e Ohnishi (2012).

Outro problema que pode ter ocorrido devido a não utilização de uma abordagem bem definida para desenvolvimento de ontologias é a falta de elementos importantes de uma ontologia, tais como seus requisitos e axiomas. A falta de axiomas para as ontologias propostas geralmente deixa dúvidas sobre as restrições existentes entre alguns conceitos das mesmas. Além disso, todas as ontologias construídas são do tipo operacional, isto é, ontologias desenvolvidas com enfoque em processamento automatizado. Há relatos na literatura sobre a importância da construção de ontologias de referência antes da construção de ontologias operacionais (Falbo, 2011). Além disso, o uso de ontologias de fundamentação (refere-se a uma ontologia cujos conceitos são independentes de domínio e é utilizada para articular conceituações dos diversos domínios – mais detalhes sobre ontologias de fundamentação são apresentados na Seção 2.4.2) nesse processo tem sido preconizado pela comunidade de engenharia de ontologias como uma boa prática para o desenvolvimento de ontologias de melhor qualidade.

2.4 Desenvolvimento de Ontologias de Domínio

O desenvolvimento de uma ontologia não é tarefa fácil. Assim, como qualquer outra atividade complexa do processo de desenvolvimento de um software, para se construir ontologias de qualidade, são necessários métodos, técnicas e ferramentas apropriadas para seu desenvolvimento (Falbo, 2011).

A literatura apresenta diversos métodos para o desenvolvimento de ontologias. Gómez-Pérez *et al*. (2003) afirmam que, desde 1990, vários pesquisadores publicaram estudos importantes para orientar a criação de ontologia, mas foi em 1995 que surgiram as primeiras propostas concretas, baseadas na experiência acumulada por pesquisadores no desenvolvimento da *Enterprise Ontology* (Uschold e King, 1995) e da TOVE (*TOronto Virtual Enterprise*) (Grüninger e Fox, 1995), propostas essas que foram refinadas nos anos seguintes.

Como resultado do MS realizado sobre o uso de ontologias de domínio no contexto da ER, notou-se que grande parte dos estudos que propõe a construção de ontologias específicas para um determinado domínio não informam o método utilizado para construção das mesmas ou informam ter utilizado uma abordagem *ad-hoc*. Uma das poucas abordagens citadas como apoio para desenvolvimento das ontologias propostas é a *SABiO* (*Systematic Approach for Building Ontology*) (Falbo, 2011), a qual é apresentada em mais detalhes na Seção 2.4.1.

2.4.1 Abordagem *SABiO*

A primeira versão da abordagem *SABiO* (*Systematic Approach for Building Ontology*) (versão 1.0) foi proposta com base na abordagem de Uschold e King (1995), adicionando-se algumas melhorias a essa abordagem, tais como o uso de uma linguagem gráfica para modelagem de ontologias, a classificação dos axiomas e o uso de questões de competência. Em sua versão mais atual (versão 2.0 – Falbo, 2011), algumas boas práticas reconhecidas pela comunidade de engenharia de ontologia foram incorporadas a essa abordagem, como por exemplo: (i) a inclusão das atividades de projeto, implementação e teste para ontologias operacionais; (ii) o reconhecimento da importância do uso de ontologias de fundamentação para o desenvolvimento de ontologias de domínio; dentre outras coisas.

A Figura 2.9 apresenta uma visão geral do processo de desenvolvimento de ontologias proposto pela abordagem *SABiO*. Esse processo é composto por cinco fases

principais, são elas: (i) identificação do propósito e elicitação dos requisitos da ontologia; (ii) captura e formalização ontológica; (iii) projeto da ontologia; (iv) implementação da ontologia; e (v) teste da ontologia.

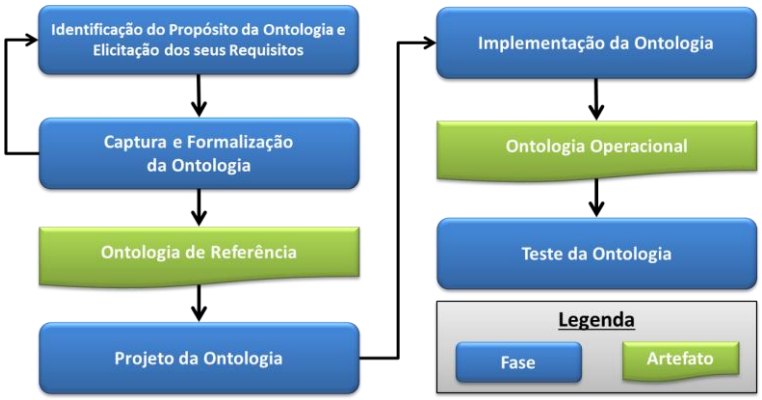


Figura 2.9. Visão geral da abordagem *SABiO* (adaptado de Falbo, 2011).

SABiO contempla tanto o desenvolvimento de ontologias de referência quanto o de ontologias operacionais. Uma ontologia de referência é um tipo especial de modelo conceitual, cujo objetivo é tornar clara e precisa a descrição dos elementos de um determinado domínio, facilitando a comunicação, o aprendizado e a resolução de problemas neste domínio. Uma ontologia operacional, por sua vez, é uma versão de implementação de uma ontologia de referência. Seu enfoque principal está em garantir que a ontologia possa ser computacionalmente processada.

Falbo (2011) afirma que, se há o interesse no desenvolvimento de uma ontologia de referência, então apenas as duas primeiras fases do processo da Figura 2.9 são necessárias. Caso uma ontologia operacional seja desejada, então todo o processo deve ser contemplado. Uma vez que o objetivo desta tese é apresentar uma ontologia de referência para o domínio de interesses de software, apenas as duas primeiras fases do processo da Figura 2.9 são descritas nesta seção.

Identificação do Propósito da Ontologia e Elicitação dos seus Requisitos. Inicialmente, deve-se identificar o propósito da ontologia e seus potenciais usos. Uma vez definido esse propósito, deve-se então, elicitar seus requisitos. Os requisitos da ontologia devem levar em consideração os potenciais usos da mesma e podem ser expressos por meio de questões de competência, isto é, questões que a ontologia deve ser capaz de responder (Grüninger e Fox, 1995).

Estabelecer as questões de competência é uma maneira de determinar o que é e o que não é relevante para uma determinada ontologia, ou seja, trata-se do seu escopo. Além disso, as questões de competência podem ser interpretadas como justificativas para a construção da ontologia, bem como uma forma de avaliá-la durante a fase de teste da ontologia. Exemplos de questões de competência para o domínio de Gerência

de Configuração (GC) são (Calhau e Falbo, 2012): (i) quais são as atividade do processo de GC?; (ii) quais são as entradas e saídas de cada atividade?; (iii) quem é responsável por realizar essas atividades?; entre outras.

Captura e Formalização Ontológica. Segundo Falbo (2011), esta é a fase mais importante do processo proposto pelos autores. O objetivo é capturar a conceituação do domínio, com base nas questões de competência. Assim, os conceitos relevantes e seus relacionamentos devem ser identificados e organizados.

O conhecimento representado por uma ontologia, necessário para responder às questões de competência, pode ser elicitado a partir de especialistas do domínio para o qual a ontologia está sendo desenvolvida, bem como por meio de conhecimentos consolidados em livros, padrões internacionais, modelos de referência, entre outros. Especialistas de domínio são pessoas com amplo conhecimento sobre o domínio da ontologia, responsáveis por prover o conhecimento que será modelado e implementado na ontologia.

A utilização de modelos gráficos é o instrumento chave para facilitar a comunicação, a negociação de significados e o estabelecimento de consenso com especialistas de domínio. A versão mais recente da abordagem *SABiO* (versão 2.0) recomenda o uso do perfil *OntoUML* como suporte para confecção desses modelos gráficos. *OntoUML* é um perfil para o diagrama de classes da UML que incorpora os principais elementos da ontologia de fundamentação *UFO* (*Unified Foundational Ontology*) (Guizzardi, 2005). Mais detalhes sobre *UFO* e o perfil *OntoUML* são apresentados nas Seções 2.4.2 e 2.4.3 deste trabalho.

Os conceitos e seus relacionamentos são a base de uma ontologia de domínio, porém, eles podem não ser suficientes para capturar adequadamente a conceituação do domínio. Quando necessário, restrições, especificadas por meio de axiomas, devem ser levadas em consideração. Os axiomas de uma ontologia podem se apresentar de duas formas: (i) axiomas de derivação (também conhecidos como axiomas ontológicos); e (ii) axiomas de consolidação. Axiomas de derivação são aqueles que permitem gerar informação nova a partir do conhecimento existente e documentado na ontologia. Axiomas de consolidação, por outro lado, não geram conhecimento novo, mas apenas restringem o estabelecimento de relações entre os conceitos de uma ontologia. Há ainda os axiomas epistemológicos, que são axiomas impostos implicitamente pela notação utilizada para construção da ontologia. Na Seção 2.4.3 são apresentados alguns exemplos de axiomas epistemológicos definidos pelo perfil *OntoUML*.

Para tornar clara a diferença entre axiomas de derivação e de consolidação, os exemplos apresentados no Quadro 2.5 são utilizados. Tais exemplos são extraídos do trabalho de Nardi e Falbo (2006), que propõe uma ontologia para requisitos de software.

Quadro 2.5. Exemplos de axiomas.

(∀r, m1, m2) (alocação(r, m1) ^ submódulo(m1, m2) → alocação(r, m2))	**(1)**
(∀ p, r, m) (alocação(r, m) ^ definição(r, p) ^ composição(m, e) → determinação (e, p))	**(2)**

(1) Axioma de Derivação; (2) Axioma de Consolidação.

No axioma do Quadro 2.5 (1), os seguintes predicados são utilizados: (i) "alocação(r, m1)": indica que o requisito "r" é alocado ao módulo "m1"; e (ii) "submódulo(m1, m2)": indica que "m1" é um sub-módulo de "m2". Esse axioma indica que os requisitos alocados a um módulo estão alocados também a seus super-módulos.

Já o axioma do Quadro 2.5 (2) define que um requisito só pode estar alocado a um módulo que esteja no escopo do projeto que definiu esse requisito. Nesse axioma, há três predicados: (i) "definição(r, p)": indica que o requisito "r" foi definido no projeto "p"; (ii) "composição(m, e)": indica que o módulo "m" compõe o escopo "e"; e (iii) "determinação(e, p)": indica que o escopo "e" foi determinado para o projeto "p".

Ressalta-se a diferença entre os axiomas do Quadro 2.5 (1) e (2): o axioma (1) deriva uma nova informação e, portanto, é dito um axioma de derivação, enquanto que o axioma (2) não deriva nova informação, mas apenas restringe o estabelecimento de relações.

Com o intuito de evitar ambiguidades, os autores da abordagem *SABiO* recomendam que os axiomas sejam escritos em lógica de primeira ordem (FOL – *First Order Logic*), assim como foram escritos os axiomas do Quadro 2.5. Além disso, para permitir a rastreabilidade entre os elementos da ontologia e de seus axiomas, sugere-se que os nomes dos predicados correspondam aos nomes dos elementos (conceitos e relacionamentos) da ontologia. A última recomendação é que os axiomas sejam acompanhados de sua descrição em linguagem natural, para facilitar a compreensão e comunicação do conhecimento representado na ontologia.

2.4.2 Ontologia de fundamentação *UFO*

Um tipo de ontologia, chamado Ontologia de Fundamentação, é uma ontologia cujos conceitos são independentes de domínio (*e.g*. conceitos como parte, todo, papel e evento) e é utilizada para articular conceituações dos diversos domínios. Um exemplo conhecido na literatura de computação é a *UFO* (*Unified Foundational Ontology*), uma ontologia de fundamentação desenvolvida com o intuito de prover uma fundamentação ontológica para linguagens gerais de modelagem conceitual (Zamborlini, 2011).

O enfoque desta tese está em uma das partes da *UFO*, denominada *UFO-A*, que define termos relacionados a aspectos estruturais como conceitos gerais de objetos,

suas propriedades intrínsecas e relacionais, os tipos que eles instanciam, os papéis que eles desempenham, entre outros.

OntoUML é um perfil UML composto por um conjunto de estereótipos e elementos de modelagem que representam as categorias ontológicas propostas na *UFO-A*. Esta seção apresenta os principais elementos da *UFO-A*, destacando suas respectivas representações no perfil *OntoUML*. As informações contidas nesta seção foram escritas com base nos trabalhos de Guizzardi (2005), Zamborlini (2011) e Benevides (2010).

Em *UFO*, "*Universal*" é a categoria ou tipo geral de elemento de modelagem conceitual, que representa os padrões de características presentes em diferentes indivíduos de um determinado domínio. Por exemplo, esse tipo se aplica aos conceitos "Pessoa*"*, "Professor*"*, "Empresa*"*, entre outros.

"Universals" podem ser classificados como "*Monadic*" ou "*Relation*". O tipo "*Monadic*" é a categoria que se aplica aos conceitos, que são padrões aplicados a indivíduos singulares, enquanto o tipo "*Relation*" se aplica às relações, que são padrões aplicados a grupos de dois ou mais indivíduos.

Elementos do tipo "*Monadic*". A principal distinção de tipos "*Monadic*" em *UFO* é a distinção entre "*Substantials*" e "*Moments*". Um conceito do tipo "*Substantial*" representa indivíduos que são existencialmente independentes de outros indivíduos, tal como uma pessoa, uma casa, um carro, entre outros. "*Moments*", por outro lado, classificam indivíduos que representam propriedades objetificadas de outros indivíduos e que são inerentes a eles. Por exemplo, a idade de uma pessoa é um "*Moment*", ou seja, um indivíduo desse tipo é inerente a outro indivíduo, no caso uma pessoa.

"*Substantials*" podem ser classificados ainda como "*Sortal*" ou "*Mixin*". Conceitos do tipo "*Sortal*" agregam indivíduos com o mesmo princípio de identidade. Por exemplo, supondo que a impressão digital defina a identidade de uma pessoa, são universais do tipo "*Sortal*" os conceitos "Pessoa", "Cliente", "Adulto", entre outros. Por outro lado, elementos do tipo "*Mixin*" são aqueles que agregam indivíduos com princípios de identidade diferentes. Supondo que o CNPJ defina a identidade de uma empresa, então o conceito "Item Assegurável", que agrega pessoas e empresas, é um universal do tipo "*Mixin*".

Os conceitos do tipo "*Sortal*" podem ser classificados como "*Rigid Sortal*" ou "*Anti-Rigid Sortal*". "*Rigid Sortal*" são conceitos rígidos como "Pessoa" e "Empresa", isto é, cujos indivíduos devem instanciá-los enquanto existirem. Por exemplo, João é necessariamente instância de "Pessoa" enquanto ele existir. Por outro lado, universais do tipo "*Anti-Rigid Sortal*" são conceitos como "Cliente" e "Adulto", cujos indivíduos

podem eventualmente instanciá-los enquanto existirem. Por exemplo, João pode ser uma instância de “Adulto” em determinado momento e em outro, não mais.

O tipo “*Rigid Sortal*” é ainda classificado como “*Substance Sortal*”, quando provê o princípio de identidade aos seus indivíduos ou como “*Subkind*”, quando apenas herda esse princípio de outro elemento. Por exemplo, considerando-se a impressão digital como o princípio de identidade provido a toda instância do conceito “Pessoa”, os conceitos “Homem” e “Mulher” são do tipo “*Subkind*”, pois herdam o princípio de identidade do conceito “Pessoa”.

O tipo “*Substance Sortal*”, por sua vez, é subdividido em “*Kind*”, “*Quantity*” e “*Collective*”. O tipo “*Kind*” é tal que suas instâncias podem ser compostas por outros indivíduos, desde que as partes exerçam papéis diferentes no todo. Por exemplo, o “*Corpo Humano*” é um “*Kind*” composto por partes com diferentes papéis, como coração e cérebro. Em contrapartida, as instâncias tipo “*Collective*” possuem partes que exercem o mesmo papel funcional no todo, por exemplo, “Floresta” (como conjunto de árvores) ou uma “Pilha de Livros” são exemplos de “*Collectives*”. Um elemento do tipo “*Collective*” pode ser “*Extensional*” ou não; ser “*Extensional*” significa que todas as partes da coleção são essenciais e a mudança (ou destruição) de alguma parte provoca a destruição de toda coleção. Finalmente, o tipo “*Quantity*” agrupa indivíduos que são porções de uma quantidade de matéria, por exemplo, a quantidade de água dentro de um copo.

Os universais do tipo “*Anti-Rigid Sortal*”, por sua vez, são classificados como “*Phase*” ou “*Role*”. Conceitos do tipo “*Phase*” são tais que sua instanciação é determinada por uma propriedade intrínseca do indivíduo. Por exemplo, João instancia a fase “Adulto” se sua idade (propriedade intrínseca) é maior que 18 anos. Já os conceitos do tipo “*Role*”, como por exemplo “Esposo”, são relacionalmente dependentes, ou seja, sua instanciação é determinada por uma propriedade relacional do indivíduo (propriedades relacionais são explicadas mais a frente neste texto). Por exemplo, João instancia o papel de “Esposo” se estiver casado com (propriedade relacional) Maria.

“Moments” podem ser classificados como “*Instrinsic Moment*” ou “*Relator*”. Elementos do tipo “*Instrinsic Moment*” denotam propriedades intrínsecas do indivíduo portador. Se essa propriedade é mensurável, ou seja, se tem um valor em uma ou mais dimensões de qualidade, ela é chamada de “*Quality*”, tal como as propriedades de “peso” ou “idade”. Por outro lado, se a propriedade não tem representação em um sistema de medida, ela é chamada de “*Mode*”, por exemplo, “dor de cabeça”. Quanto aos “*Relators*”, estes são elementos interventores ou mediadores, ou seja, indivíduos que mediam outros tornando verdadeira alguma relação entre eles. Exemplos de “*relators*” são apresentados mais adiante neste texto.

Elementos do tipo "*Relation*". O tipo "*Relation Universal*" é a categoria geral que se aplica aos universais de relação, também chamados apenas de relações. Esta categoria é subdividida em "*Formal*", "*Material*" e "*Meronymic*".

O tipo "*Material*" se aplica às relações que dependem de algum interventor para valer, a saber, um indivíduo do tipo "*Relator*". Por exemplo, a relação "casado com" vale enquanto existir um "casamento" (*Relator*).

Contrariamente, as relações do tipo "*Formal*" valem pela simples existência dos indivíduos relacionados. Por exemplo, a relação entre João e seu cérebro vale sempre que ambos existirem. As relações do tipo "*Formal*" podem ser ainda classificadas como "*Basic Internal Relation*" ou "*Domain Formal Relation*".

O tipo "*Basic Internal Relation*" aplica-se a relações formais internas, ou seja, que estão relacionadas a outras categorias da *UFO*. Os dois principais tipos de "*Basic Internal Relation*" são "*Characterization*" e "*Mediation*". "*Characterization*" é um tipo de relação binária que associa um elemento do tipo "*Mode*" (por exemplo, "estado mental") com um elemento do tipo "*Monadic*" (por exemplo, "Pessoa"). "*Mediation*", por sua vez, associa um "*Relator*" (por exemplo, "casamento") com o elemento "*Monadic*" mediado por ele (por exemplo, "Homem" e Mulher).

Já o tipo "*Domain Formal Relation*" se aplica às relações formais que são específicas de (inerentes ao) domínio e que, por isso, não estão relacionadas a qualquer tipo específico de elemento *UFO*. Exemplos de relações formais desse tipo são: "mais velha" (uma pessoa é mais velha do que outra), "mais pesado" (um átomo é mais pesado do que outro), entre outros.

Por fim, o tipo de relação "*Meronymic*" refere-se às relações todo-parte. Esse tipo é ainda subdividido em "*ComponentOf*", "*MemberOf*", "*SubcollectionOf*" e "*SubQuantityOf*". O tipo "*ComponentOf*" vale entre indivíduos do tipo "*Kind*" ou "*Subkind*", por exemplo, a mão faz parte do braço, a ULA (Unidade Lógica Aritmética) faz parte de um "Processador*"*; (ii) "*MemberOf*" vale entre indivíduos do tipo "*Kind*" ou "*Subkind*" e coleções, por exemplo, uma "*Árvore*" é parte de uma "Floresta*"*, uma "Carta" é parte de um "Baralho*"*; (iii) "*SubColectionOf*" vale entre coleções, uma "população de homens" faz parte da "população em geral"; e (iv) "*SubQuantityOf*" vale entre quantidades, por exemplo, "álcool" faz parte do "vinho", plasma faz parte do "sangue".

2.4.3 O perfil *OntoUML*

Muitos dos conceitos descritos anteriormente foram modelados como classes abstratas no perfil *OntoUML* e, portanto, não geram representações visuais por meio de estereótipos nos modelos *OntoUML*.

Assim, esta seção apresenta os conceitos *UFO* que se tornaram estereótipos no perfil *OntoUML*, bem como as restrições intrínsecas à aplicação desses estereótipos. Essas restrições podem ser entendidas como axiomas epistemológicos, pois, como já comentado, são axiomas impostos implicitamente pela notação utilizada para construção da ontologia. Além disso, apresenta-se a classe base do metamodelo da UML que foi estendida no perfil *OntoUML*. Essa informação é útil para se entender qual tipo de elemento de um diagrama de classes da UML é capaz de receber o estereótipo em questão.

Por questões de simplicidade, apenas os estereótipos utilizados no modelo da ontologia *O4C* são descritos nesta seção. Mais informações sobre os demais estereótipos podem ser obtidas em (Guizzardi, 2005; Benevides, 2010).

Elemento *UFO*:	**Estereótipo:**	**Classe base:**
Kind	«`kind`»	Classe
Domain Formal Relation	«`formal`»	Relacionamento de Associação

Não há restrições para uso desses dois elementos.

Elemento *UFO*:	**Estereótipo:**	**Classe base:**
Subkind	«`subkind`»	Classe

Restrições: elementos "*Subkind*" são do tipo "*Rigid Sortal*" e, portanto, não podem ter como superclasses elementos dos tipos "*Phase*", "*Role*" ou "*RoleMixin*".

Elemento *UFO*:	**Estereótipo:**	**Classe base:**
Quality	não há	Propriedade

Restrições: atributos das classes da UML são utilizados para representar elementos do tipo "*Quality*". Esta é a maneira adotada neste trabalho, por ser mais concisa, contudo, há outra forma possível de se representar elementos "*Quality*", que é por meio dos estereótipos «`datatypeRelationship`» e «`simpleDatatype`». Mais informações podem ser obtidas em (Guizzardi, 2005; Benevides, 2010).

Elemento *UFO*:	**Estereótipo:**	**Classe base:**
Relator	«`relator`»	Classe

Restrições: *relators* são elementos mediadores, ou seja, indivíduos que mediam outros, portanto cada elemento do tipo "*Relator*": (i) deve estar conectado a, pelo menos, uma associação do tipo "*Mediation*"; e (ii) deve medir o relacionamento entre, pelo menos, dois indivíduos.

Elemento *UFO*:	**Estereótipo:**	**Classe base:**
Mediation	«`mediation`»	Relacionamento de Dependência

Restrições: cada elemento do tipo "*Mediation*" deve estar conectado a, pelo menos, um elemento do tipo "*Relator*".

Elemento *UFO*:	**Estereótipo:**	**Classe base:**
-	«derivation relation»	Relacionamento de Dependência

Restrições: não há um representante explícito desse elemento no modelo *UFO*. Ele deriva-se da existência de relações do tipo "*Material*" e serve para tornar explícito de qual "*Relator*" esta relação é derivada. Por isso, uma "*Derivation Relation*" deve estar sempre relacionada a um "*Relator*" e a uma relação do tipo "*Material*".

Elemento *UFO*:	**Estereótipo:**	**Classe base:**
Material	«material»	Relacionamento de Associação

Restrições: cada elemento do tipo "*Material*" deve estar conectado a exatamente uma "*Derivation Relation*".

Elemento *UFO*:	**Estereótipo:**	**Classe base:**
ComponentOf	«componentOf»	Agregação/Composição

Restrições: cada elemento do tipo "*ComponentOf*" deve estar conectado a elementos do tipo "*Kind*" ou "*Subkind*".

Para ilustrar o uso do perfil *OntoUML*, na Figura 2.10 é apresentado um exemplo hipotético de ontologia para o domínio de controle acadêmico, adaptado de (Benevides, 2010) – esta adaptação consistiu unicamente na tradução dos termos utilizados pelo autor para a língua portuguesa.

Nesta figura, "Pessoa" é um elemento "*Rigid Sortal*", ou seja, uma instância desse conceito não pode deixar de ser uma pessoa enquanto ele existir. Além disso, *Pessoa* provê o princípio de identidade aos seus indivíduos, por isso é classificado como "*Kind*". "Estudante" é um "*Anti-Rigid Sortal*" do tipo "*Role*", pois um indivíduo pode ser um estudante em determinado momento e deixá-lo de ser em outro momento. Além disso, o fato de ele ser um "*Role*" é porque sua existência depende de uma propriedade de relacionamento do indivíduo com outro indivíduo. Neste caso, trata-se do relacionamento de um estudante com uma escola por meio da relação *estuda*.

A relação "estuda" é do tipo "*Material*", pois depende de algum interventor para se tornar válida. Neste caso, o interventor é o "Relator Matrícula". Isto significa que uma relação "estuda" só existirá se houve uma matrícula do aluno em uma escola. "Matrícula" é o "Relator" entre "estudante" e "escola", ou seja, é o elemento que intermedia o relacionamento "estuda" entre essas entidades e o torna possível. O último ponto a ser destacado são as relações entre os conceitos "Pessoa/Coração" e

"Pessoa/Cérebro", ambas do tipo "*ComponentOf"*. Isso faz sentido, pois os três elementos são do tipo "*Kind*" e estão relacionados por meio de uma ideia de todo-parte.

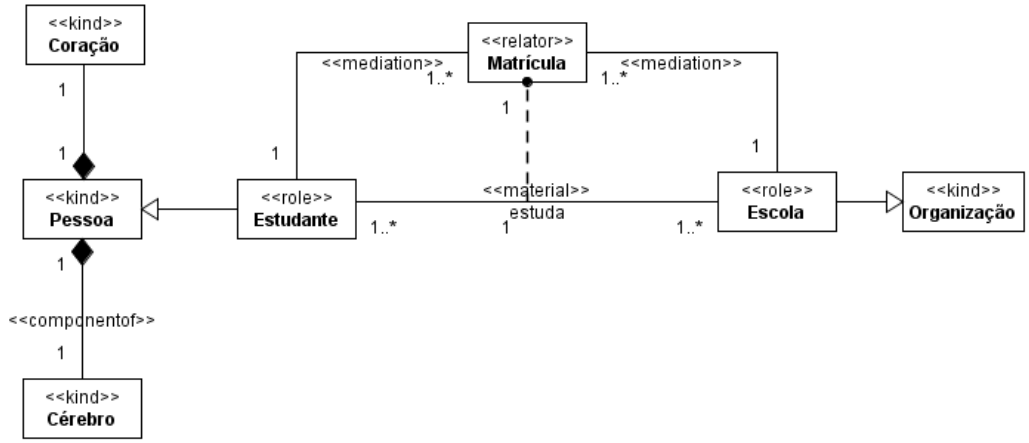


Figura 2.10. Exemplo de uso do perfil *OntoUML* (Benevides, 2010).

2.5 Considerações Finais

Este capítulo apresentou informações relevantes, com base nas limitações encontradas nos trabalhos relacionados ao tema deste doutorado, para o entendimento das motivações para desenvolvimento desta pesquisa. Além disso, ele serviu como base para compreensão dos conceitos adotados nos demais capítulos desta tese.

Como principais destaques, comentou-se sobre a escassez de recursos de apoio à realização da atividade "Identificação e Classificação de Interesses". Como "recursos", entende-se repositórios de informações sobre interesses específicos, bem como métodos/diretrizes de como utilizar esses repositórios para apoiar o engenheiro de software durante a realização dessas atividades.

Ainda neste capítulo, comentou-se sobre algumas propostas promissoras quanto ao uso de ontologias de domínio para solucionar alguns problemas relacionados à verificação, elicitação e análise de requisitos de software. Contudo, não foram encontrados trabalhos específicos para a área de interesses de software, em especial, para o domínio de interesses transversais (EROA).

Conforme discutido no Capítulo 1, este trabalho propõe o uso de uma ontologia de domínio para prover uma compreensão clara e consensual sobre o domínio de interesses de software. Para o desenvolvimento de tal ontologia, utilizou-se a abordagem *SABiO*, bem como a ontologia de fundamentação *UFO* e o perfil *OntoUML*, também abordados neste capítulo. Tais abordagens ofereceram suporte teórico

importante para o desenvolvimento da ontologia proposta neste trabalho, permitindo que a mesma fosse desenvolvida com base em boas práticas preconizadas pela comunidade de engenharia de ontologias. Esse é um dos principais diferenciais deste trabalho com relação a vários outros trabalhos que apresentaram ontologias de domínio como apoio para atividades da engenharia de requisitos, bem como para aqueles relacionados à EROA.

Capítulo 3

O4C: UMA ONTOLOGIA PARA O DOMÍNIO DE INTERESSES DE SOFTWARE

3.1 Considerações Iniciais

Como interesses e requisitos de software estão no foco do processo de EROA, é interessante que se tenha uma compreensão clara desses elementos, quais são suas principais características/propriedades, como eles se relacionam uns com os outros, entre outros. Como fruto dessa compreensão, pode-se permitir a construção de métodos, técnicas e ferramentas de apoio à EROA que sejam amplamente utilizáveis, já que estão baseados em definições compartilhadas sobre o domínio de interesses de software.

A ontologia para o domínio de interesse de software *O4C* (*Ontology for Concerns*) (Parreira Júnior e Penteado, 2015a), é uma ontologia de referência responsável por representar conceitos e relacionamentos bem conhecidos e já documentados na literatura sobre interesses de software no contexto da EROA. Também é a base para a definição de catálogos de tipos específicos de interesses, que são utilizados no processo de identificação e classificação de interesses a partir de requisitos de software. Além dos conceitos, propriedades e axiomas da ontologia *O4C* (Seções 3.3.1 à 3.3.8), este capítulo apresenta ainda o seu processo de desenvolvimento, com base na abordagem *SABiO* (Falbo, 2011), na ontologia de fundamentação *UFO* (*Unified Foundation Ontology*) e no perfil *OntoUML* (Guizzardi, 2005) – Seções 3.2 e 3.3. Por fim, na Seção 3.4 são apresentadas as considerações finais deste capítulo.

3.2 Identificação do Propósito da Ontologia *O4C* e Elicitação dos seus Requisitos

Conforme comentado na Seção 2.4.1, os requisitos de uma ontologia devem considerar os potenciais usos da mesma e podem ser expressos por meio de questões de competência. Nesse sentido, este trabalho considera que os principais usos da ontologia *O4C* são:

- prover um vocabulário comum para os conceitos e relacionamentos do domínio de interesses de software no contexto da EROA; e
- permitir a construção de catálogos de interesses de software, por meio da instanciação dos conceitos e relacionamentos disponíveis na *O4C*, que possam auxiliar na identificação e classificação mais efetiva dos interesses de um software.

A partir desses potenciais usos, um conjunto de questões de competência para a ontologia *O4C* foi criado e dividido em dois subconjuntos: (i) o conjunto das *Questões Independentes de Contexto* (*QIC*); e (ii) o conjunto das *Questões Dependentes de Contexto* (*QDC*).

As *QIC* visam a prover uma base de conhecimento independente de contexto sobre a natureza dos interesses de software, indicando quais são seus principais conceitos e como eles se relacionam entre si. Uma justificativa para a existência desse tipo de questão é que a clara definição das características de um interesse pode auxiliar na confecção de ferramentas, métodos e técnicas mais apropriadas para EROA, isto é, passíveis de integração e compatíveis entre si. Além disso, tais características podem servir de apoio para elaboração de catálogos de interesses de software mais abrangentes. As seguintes *QIC* foram elaboradas para a ontologia *O4C*:

- *QIC-1*) Quais são os possíveis tipos de interesses de software?
- *QIC-2*) Como saber quais fontes deram origem a um determinado interesse?
- *QIC-3*) Como saber quais são as palavras-chave correspondentes a um determinado interesse?
- *QIC-4*) Como saber quando um interesse depende de outro e quais são os interesses dos quais ele depende?
- *QIC-5*) Como saber quando um interesse é decomposto em subinteresses e quais são os seus subinteresses?
- *QIC-6*) Como saber quando um interesse contribui para outro e quais são os tipos de contribuição que ele exerce?

As *QDC* visam a especificar conceitos e relacionamentos que dependem do contexto sobre o qual o processo de EROA está sendo executado, como por exemplo, os requisitos de um software e seus relacionamentos, a prioridade de um interesse, o relacionamento entre interesses e os requisitos do software, entre outros. Além de contemplar as justificativas para as *QIC*, outra justificativa para a existência das questões dependentes de contexto é que as respostas para as mesmas podem ser úteis no processo de identificação e classificação de interesses de um software, como será apresentado no Capítulo 4 desta tese. As seguintes *QDC* foram elencadas para a ontologia *O4C*:

- *QDC-1*) Como saber qual é a prioridade de um determinado interesse em um projeto de software?
- *QDC-2*) Como saber quando um requisito depende de outro e quais são os requisitos dos quais ele depende?
- *QDC-3*) Como saber qual é o interesse principal de um requisito de software?
- *QDC-4*) Como saber quando um interesse é ou não transversal?
- *QDC-5*) Como saber quando um requisito é entrecortado por interesses transversais e quais são os interesses transversais que o entrecortam?

3.3 Formalização Ontológica da *O4C*

Após a identificação das questões de competência, deve-se elicitar o conhecimento necessário para responder a essas questões. Esse conhecimento pode ser obtido a partir de especialistas do domínio para o qual a ontologia está sendo desenvolvida, bem como por meio de livros, padrões internacionais, modelos de referência, entre outros (Falbo, 2011; Grüninger e Fox, 1995).

No MS sobre EROA conduzido durante este projeto (Parreira Júnior e Penteado, 2014; Parreira Júnior e Penteado, 2015d) notou-se que algumas das abordagens existentes na literatura propõem o uso de recursos (em geral, arquivos XML) para armazenar/representar o conhecimento a respeito dos interesses de um software e do relacionamento destes com os requisitos do mesmo (Chernak, 2012; Zheng *et al*., 2010; Alencar *et al*., 2010; Mussbacher *et al*., 2010; Agostinho *et al*., 2008; Chitchyan *et al*., 2006; Soeiro *et al*., 2006; Sampaio *et al*., 2005; Moreira *et al*., 2005; Clarke e Baniassad, 2005; Baniassad e Clarke, 2004; Whittle e Araújo, 2004; Brito e Moreira, 2003). As principais características desses recursos foram utilizadas como base para a formalização ontológica da ontologia *O4C*. Além disso, o conhecimento de dois pesquisadores envolvidos com a

área de interesses de software por mais de 12 (doze) anos também foi levado em consideração, principalmente, para a análise dos recursos das abordagens para EROA existentes na literatura.

A versão completa do modelo gráfico da ontologia *O4C*, baseado no perfil *OntoUML*, é apresentada na Figura 3.1. Para cada conceito da *O4C* apresentado nesta seção, são destacados os trabalhos relacionados utilizados para definição do mesmo.

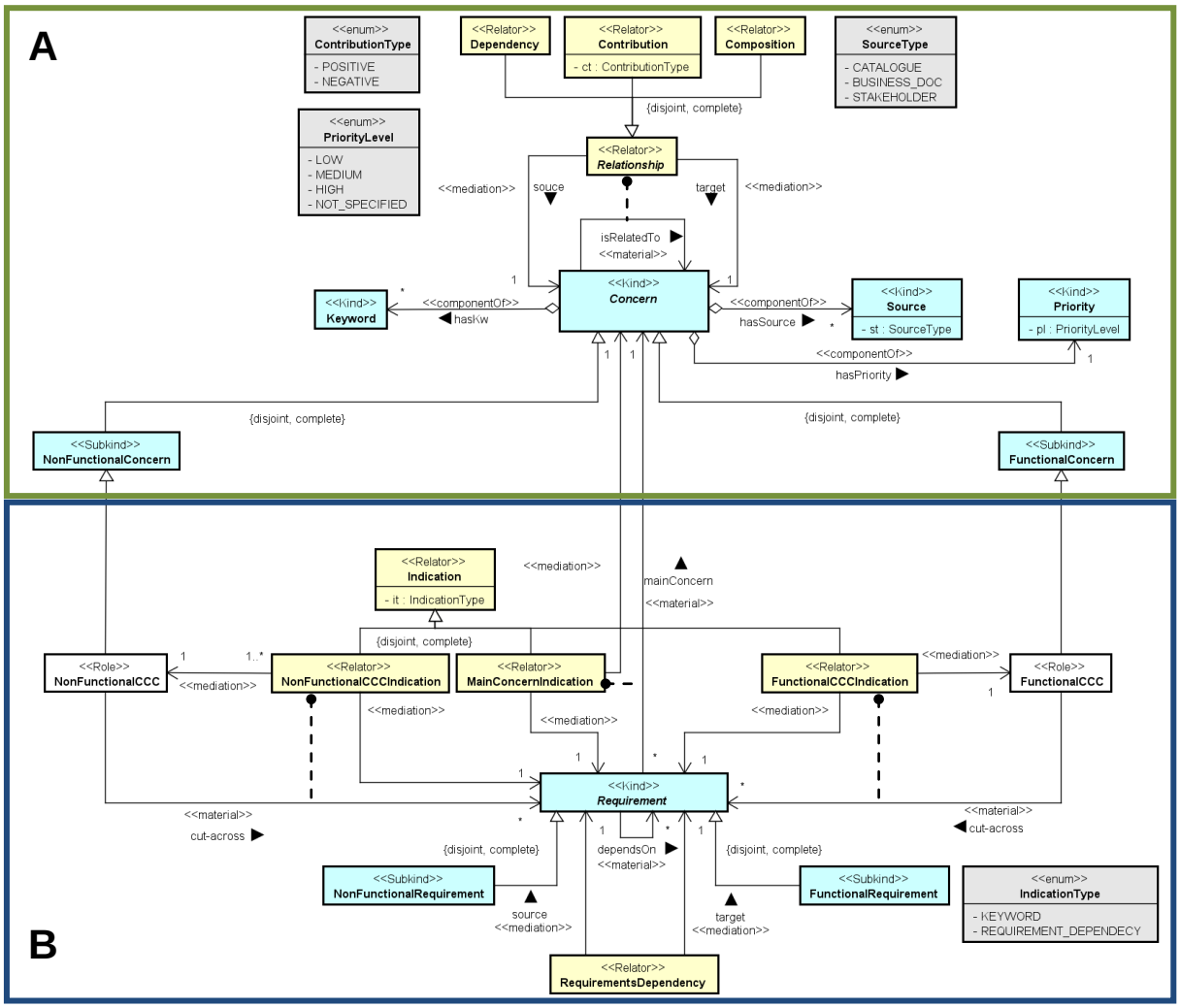


Figura 3.1. Representação gráfica da ontologia para interesses de software *O4C*.

Para facilitar o entendimento dos conceitos e relacionamentos da ontologia *O4C*, a Figura 3.1 foi dividida em duas partes A e B. A parte A dessa ontologia apresenta conceitos e relacionamentos necessários para responder às *Questões Independentes de Contexto* e a parte B descreve conceitos e relacionamentos que contemplam as *Questões Dependentes de Contexto*. Além disso, no diagrama ilustrado nessa figura, conceitos do tipo "*Relator*" estão destacados em amarelo, conceitos dos tipos "*Kind*" e "*Subkind*" estão em azul, "*Roles*" estão representados na cor branca e classes enumeradas (*enum*), na cor cinza.

3.3.1 *Concern*, *FuncionalConcern* e *NonFuncionalConcern*

O conceito "*Concern*" representa os indivíduos que atendem às propriedades estabelecidas para um interesse de software. Dois subtipos desse conceito encontrados na literatura são "*FunctionalConcern*" e "*NonFunctionalConcern*". "*FunctionalConcern*" refere-se a interesses relacionados a características funcionais de um software, como por exemplo, pagamento e gerenciamento de pedidos. O conceito "*NonFunctionalConcern*" refere-se a interesses relacionados à características não funcionais de um software, como por exemplo, segurança, *logging* e persistência.

Uma vez que uma instância de um interesse não pode deixar de ser um interesse ao longo de sua existência e que o conceito de "*Concern*" provê identidade às suas instâncias (por meio do nome do interesse), então o conceito "*Concern*" foi estereotipado com «`Kind`». Os conceitos "*FunctionalConcern*" e "*NonFunctionalConcern*" são «`Subkind`», uma vez que herdam o princípio de identidade do conceito "*Concern*".

É importante salientar que um interesse pode ou não existir em um determinado software, isto é, pode ou não estar relacionado a requisitos desse software. Entretanto, instâncias do conceito "*Concern*" são independentes do contexto sobre o qual está sendo realizada a identificação e classificação de interesses. Assim, um interesse "Segurança", por exemplo, sempre será uma instância do conceito "*Concern*" da ontologia *O4C*, mas ele pode estar presente ou não em um software específico. Essa situação fica mais clara à medida que os conceitos da parte B da ontologia *O4C* forem explicados neste capítulo.

É importante observar por meio da Figura 3.1 (A) que a classe que representa o conceito "*Concern*" é abstrata, de forma que apenas suas subclasses podem ser instanciadas, isto é, podem dar origem a indivíduos. Além disso, o relacionamento de generalização/especialização existente entre essa classe e suas subclasses é do tipo disjunto e completo *(disjoint* e *complete*), o que significa que uma instância de "*Concern*" só pode ser do tipo "*FunctionalConcern*" ou "*NonFunctionalConcern*", não os dois ao mesmo tempo.

Os conceitos "*Concern*", "*FunctionalConcern*" e "*NonFunctionalConcern*" são bem conhecidos pela comunidade de EROA e são relatados em todos os trabalhos analisados (Chernak, 2012; Zheng *et al*., 2010; Alencar *et al*., 2010; Mussbacher *et al*., 2010; Agostinho *et al*., 2008; Chitchyan *et al*., 2006; Soeiro *et al*., 2006; Sampaio *et al*., 2005; Moreira *et al*., 2005; Clarke e Baniassad, 2005; Baniassad e Clarke, 2004; Whittle e Araújo, 2004; Brito e Moreira, 2003). Tais conceitos são úteis para responder à *Questão de Competência QIC-1*.

3.3.2 *Keyword e Source*

O conceito "*Keyword*" (palavra-chave) aparece em diversas abordagens para EROA (Chernak, 2012; Zheng *et al*., 2010; Sampaio *et al*., 2005; Clarke e Baniassad, 2005; Baniassad e Clarke, 2004), mas nenhum dos trabalhos existentes apresentou a ideia de documentá-lo a fim de apoiar a atividade de identificação e classificação de interesses em projetos futuros. Na ontologia *O4C*, esse conceito foi criado para armazenar as palavras-chave comumente utilizadas para identificação de um determinado interesse. Uma instância do conceito "*Keyword*" não pode deixar de ser uma palavra-chave enquanto ela existir, sendo assim, esse conceito foi estereotipado com «`Kind`».

Os conceitos representados pela classe `Source`, seu atributo `st` e a classe enumerada `SourceType`, referem-se às possíveis fontes a partir das quais a descrição de um interesse de software, bem como de suas propriedades e relacionamentos, podem ser extraídas. Esse conceito aparece em algumas abordagens para EROA, tais como as propostas por Agostinho *et al*. (2008), Whittle e Araújo (2004) e Brito e Moreira (2003). Segundo essas abordagens, os possíveis tipos de fonte são: (i) *stakeholders* (por exemplo, um gerente de projetos, um especialista em segurança da informação, entre outros); (ii) catálogos, como por exemplo, os catálogos de requisitos não funcionais propostos por Cysneiro (2015), Chung e Leite (2000), Boehm e In (1996), entre outros; ou (iii) documentos de negócio, como por exemplo, um protocolo de segurança de uma empresa, um regimento interno, entre outros.

Um interesse de software pode estar relacionado a várias fontes e elas são importantes no processo de identificação e classificação de interesses, pois podem indicar ao engenheiro de software a quem ou a quê recorrer quando um determinado interesse não está sendo identificado adequadamente em um determinado projeto. As razões para a utilização dos estereótipos «`Kind`» e «`Subkind`» nestes conceitos são análogas àquelas dadas para os conceitos "*Concern*", "*NonFunctionalConcern*" e "*FunctionalConcern*".

Os relacionamentos existentes entre o conceito "*Concern*" e os conceitos "*Source*" e "*Keyword*", respectivamente "*hasSource*" e "*hasKw*", receberam o estereótipo «`componentOf`». Esse é o estereótipo indicado para relações do tipo todo-parte entre elementos do tipo "*Kind*" ou "*Subkind*". Ele é adequado para esses relacionamentos da ontologia *O4C*, pois considera-se que as palavras-chave de um interesse e suas fontes são parte da descrição deste interesse.

Os conceitos "*Keyword*" e "*Source*" e os relacionamentos "*hasKw*" e "*hasSource*" podem ser utilizados para responder às *Questões de Competência QIC-2* e *QIC-3*.

3.3.3 *Contribution*, *Dependency* e *Composition*

Os três tipos de relacionamentos entre interesses documentados pela ontologia *O4C* são definidos pelos conceitos "*Contribution*", "*Dependency*" e "*Composition*", que são subconceitos de "*Relationship*".

Os conceitos "*Relationship*", "*Contribution*", "*Dependency*" e "*Composition*" recebem o estereótipo «`Relator`». Já o relacionamento "*isRelatedTo*", que vincula uma instância do conceito "*Concern*" a outras instâncias do mesmo conceito recebe o estereótipo «`Material`». Conceitos do tipo "*Relator*" referem-se a elementos interventores ou mediadores, ou seja, que mediam outros conceitos, tornando verdadeira alguma relação entre eles. Já o tipo de relacionamento "*Material*" se aplica às relações que dependem de algum interventor para valer.

No caso da ontologia *O4C*, as subclasses de "*Relationship*" são as responsáveis por garantir a existência do relacionamento do tipo "*Material*", denominado "*isRelatedTo*". Ou seja, não é possível que um interesse de software esteja relacionado a outro, sem que haja uma instância de um conceito "*Relationship*" que descreva isso.

É importante destacar ainda os relacionamentos estereotipados com «`Mediation`» (Figura 3.1 - A). Um relacionamento do tipo "*Mediation*" vincula o "*Relator*" aos elementos cujo relacionamento é mediado por ele. Neste caso, um dos relacionamentos desse tipo descreve qual é a origem (*source*) e o outro descreve qual é o destino (*target*) do relacionamento "*isRelatedTo*". Esse tipo de distinção é importante para cada tipo de relacionamento, conforme será explicado posteriormente nesta seção.

Por fim, ressalta-se a existência de um relacionamento entre a classe "*Relationship*" e o relacionamento "*isRelatedTo*", representado por uma linha pontilhada. Na *UFO* esse tipo de relacionamento é tratado como "*Derivation Relation*" e seu objetivo é tornar explícito de qual "*Relator*" a relação do tipo "*Material*" é derivada (Guizzardi, 2005). Na ontologia *O4C* ele descreve que "*Relationship*" é o "*Relator*" responsável por tornar possível a relação denominada "*isRelatedTo*".

O tipo de relacionamento entre interesses tratado pelo conceito "*Composition*" descreve a ideia de decomposição de um interesse em subinteresses. Considera-se que o conceito "*Composition*" é importante, uma vez que um determinado interesse pode ser demasiadamente amplo, como é o caso da segurança, persistência, entre outros, e tratá-lo de uma vez só pode não ser trivial.

Reduzir a granularidade destes interesses amplos, tratando-os com base em seus subinteresses, pode facilitar o raciocínio do engenheiro de software sobre quais interesses realmente estão presentes no software e quais seriam as estratégias mais adequadas para modularizá-los. Por exemplo, o interesse "Segurança" pode ser decomposto em

subinteresses menores, tais como “Autorização”, “Criptografia”, entre outros. Em um determinado software, é possível que haja o subinteresse “Autorização”, mas não o subinteresse “Criptografia”.

Para ilustrar outra situação em que o conceito de decomposição de interesses pode ser útil, considere o interesse funcional “Transação”, que pode ter como subinteresses “Locação” ou “Venda”. Considere ainda o interesse funcional “Pagamento”, que depende dos interesses “Locação” ou “Venda” (o conceito de dependência é explicado mais adiante nesta seção). Com base na ideia de composição, pode-se especificar um único relacionamento de dependência entre o interesse “Pagamento” e “Transação”, uma vez que “Locação” e “Venda” são subinteresses de “Transação”. Assim, caso o interesse “Locação” tenha sido identificado em um determinado software, considera-se também que o interesse “Transação” foi identificado. Esse tipo de situação é mais bem explicado na Seção 3.3.8, na qual os axiomas da ontologia *O4C* são apresentados.

É importante ressaltar que cada um desses subinteresses são instâncias do conceito “*Concern*”. O relacionamento “*Composition*” serve, então, para tornar explícito o modo como os interesses são subdivididos. “*Composition*” também é um “*Relator*” e herda as propriedades do elemento “*Relationship*”, isto é, as relações de mediação “*source*” e “*target*”. Assim, é possível saber qual é o interesse composto (*source*) e quais são seus subinteresses (*targets*) – *Questão de Competência QIC-5*.

O relacionamento do tipo “*Dependency*” descreve uma relação de dependência entre dois interesses, de forma que para que um interesse “A” (*source*) esteja presente no software é preciso que “B” (*target*) também esteja. Ele é importante, pois: (i) permite que o engenheiro de software explore outros interesses antes não reconhecidos por ele, uma vez que, ao dizer que “A” depende de “B”, ele deverá procurar também por palavras-chave do interesse “B” no software; e (ii) permite ao engenheiro de software verificar inconsistências no documento de requisitos do software. Se um interesse “A” depende de “B” e “B” não está descrito nos requisitos do software, então o documento de requisitos pode estar inconsistente. O conceito “*Dependency*”, bem como os relacionamentos “*source*” e “*target*”, são úteis para se responder à *Questão de Competência QIC-4*.

Outro importante elemento da ontologia *O4C* é o representado pelo conceito “*Contribution*”, que está relacionado à influência mútua que um interesse de software pode exercer sobre outro. Uma “*Contribution*” pode ser “*Negative*” ou “*Positive*”, conforme definido pela classe enumerada `ContributionType` e pelo atributo `ct`, da classe `Contribution`.

Esse tipo de conceito é importante, principalmente durante a detecção e análise de conflitos entre os interesses relacionados, mas pode ser útil também durante o processo de identificação de interesse. Isso porque, se um interesse “A” contribui positivamente para um interesse “B” e “B” foi identificado no software, mas “A” não, há indícios de que a

identificação do interesse “A” pode ter apresentado problemas. Na Seção 2.2.1 desta tese foi apresentado um exemplo do relacionamento de contribuição existente entre os interesses “Recuperação de Informação” e “Mobilidade”. Outro exemplo é o caso dos interesses “Concorrência”, “Desempenho” e “Custo”. A implementação de mecanismos de concorrência em um software pode contribuir positivamente para o desempenho do mesmo, mas negativamente para o custo de um projeto. O conceito “*Contribution*”, bem como os relacionamentos “*source*” e “*target*”, são úteis para se responder à *Questão de Competência QIC-6*.

Ressalta-se que os relacionamentos do tipo “*Contribution*”, “*Composition*” e “*Dependency*” não são mutuamente exclusivos, isto é, pode haver casos em que o interesse “A” depende do interesse “B” e ao mesmo tempo “A” contribui positivamente para “B”.

Os conceitos “*Contribution*”, “*Composition*” e “*Dependency*” aparecem de forma bastante divergente nos trabalhos relacionados. A ideia representada pelo conceito “*Contribution*” é relatada na abordagem *MDSoC* (Moreira *et al*., 2005). Para isso, é criada uma matriz de contribuição por meio da qual o engenheiro de software deve, com base em sua experiência e a partir da leitura de alguns catálogos de requisitos não funcionais, dizer quais contribuições há entre os diversos interesses do software.

Analogamente, na abordagem *EROA/XML* (Soeiro *et al*., 2006), o *template* criado pelos autores para representação dos interesses do software também leva em consideração a contribuição existente entre esses interesses. Contudo, esse tipo de conhecimento fica limitado ao projeto em análise e não há mecanismos claramente definidos pelos autores para reusá-lo em projetos posteriores. Outro ponto a ser ressaltado sobre o trabalho de Moreira *et al*. (2005) é que o *template* proposto pelos autores descreve o relacionamento entre diferentes interesse por meio da *tag* `<Relationships>` (Seção 2.2.1), contudo, não há diferenciação entre os tipos de relacionamento existentes. O relacionamento do tipo “*Dependency*” foi encontrado apenas em *EROA/XML* (Soeiro *et al*., 2006). Já o conceito “*Composition*” não foi encontrado nos trabalhos analisados nesta pesquisa.

Por fim, ressalta-se que em todos os casos comentados anteriormente, as abordagens não relatam como a representação das informações sobre os relacionamentos entre interesses podem ser úteis no processo de identificação e classificação de interesses a partir dos requisitos do software, ficando dependente da experiência do engenheiro de software utilizar ou não essa informação de forma adequada.

3.3.4 *Priority*

Cada interesse de software possui uma prioridade atribuída por *stakeholders* com conhecimento no domínio do interesse em questão e pode assumir os valores como “*High*”,

"*Medium*", "*Low*" ou "*Not_specificed*" (definido pela classe enumerada `PriorityLevel` e pelo seu atributo `pl`). O conceito de prioridade é importante, principalmente, no momento em que dois interesses distintos exercem influências negativas sobre outro ou quando um exerce influência negativa e o outro positiva e é preciso definir qual deles será atendido. Por exemplo, considerando os interesses "Concorrência", "Desempenho" e "Custo", o engenheiro de software terá que decidir entre priorizar o "Custo" ou o "Desempenho", uma vez que a "Concorrência" pode afetar positivamente a "Desempenho", mas negativamente o "Custo".

Contudo, a prioridade de um interesse pode variar de projeto para projeto, a depender do domínio para o qual o software está sendo desenvolvido, bem como dos *stakeholders* envolvidos no projeto. Por exemplo, levando em consideração o exemplo acima, em um projeto de software crítico, o interesse "Desempenho" pode ter prioridade maior do que a prioridade do "Custo". Já em um projeto de software comercial, "Custo" pode ser um interesse mais importante de ser considerado do que "Desempenho". Por esse motivo, apesar de estar diretamente relacionado ao conceito "*Concern*", a especificação da prioridade de um interesse pode ser postergada até à fase de detecção e análise de conflitos entre interesses. Para isso, o valor "*Not_specificed*" pode ser utilizado durante a confecção de um catálogo de interesses.

"*Priority*" é um conceito contemplado em algumas abordagens para EROA (Soeiro *et al*., 2006; Moreira *et al*., 2005) e é responsável por contemplar a *Questão de Competência QDC-1*.

3.3.5 *Requirement*, *FunctionalRequirement*, *NonFunctionalRequirement* e *RequirementsDependency*

A partir desta subseção, são apresentados os conceitos correspondentes à parte B da ontologia *O4C* (Figura 3.1 - B). "*Requirement*", "*FunctionalRequirement*", "*NonFunctionalRequirement*" e "*RequirementsDependency*" correspondem aos conceitos utilizados para requisitos de software no contexto da EROA. Tais conceitos são utilizados na maioria das abordagens para EROA, tais como *MDSoC* (Moreira *et al*., 2005), *EA-Miner* (Chitchyan *et al*., 2006; Sampaio *et al*., 2005), *EROA/XML* (Soeiro *et al*., 2006), *Theme* (Clarke e Baniassad, 2005; Baniassad e Clarke, 2004), bem como em abordagens que fazem uso de ontologias no contexto da engenharia de requisitos (Verma e Kass, 2008).

Uma diferença da conceitualização sobre requisitos de software utilizada neste trabalho para a dos demais trabalhos relacionados à EROA é a especificação do relacionamento de dependência entre requisitos do software. A dependência entre requisitos

é um conceito bem conhecido na literatura da engenharia de requisitos e encontra-se documentado em ontologias específicas para requisitos de software, como a proposta por Nardi e Falbo (2006). Segundo Nardi e Falbo (2006), uma relação de dependência indica que, se um requisito do qual outros requisitos dependem for alterado, é provável que os requisitos dele dependentes sofram algum impacto. Por exemplo, considerando os requisitos "o software deve permitir que o hóspede cancele sua reserva em até 48 horas antes da data de *check-in*" e "para cancelar uma reserva, o hóspede deve utilizar o *token* de segurança encaminhado para sua conta de *email*". Considera-se que o primeiro requisito depende do segundo, uma vez que esse impõe uma restrição de segurança sobre o primeiro requisito.

Outra diferença do modelo conceitual para requisitos de software da ontologia *O4C* com relação aos modelos das demais abordagens para EROA consiste na especificação do tipo do requisito de software, isto é, se ele é funcional ou não funcional. Essa distinção entre tipos de requisitos é importante, pois oferece informações de apoio ao engenheiro de software para a realização da atividade de identificação e classificação de interesses, conforme é explicado no Capítulo 4 desta tese.

A explicação sobre os tipos de estereótipos e relacionamentos utilizados para especificação dos conceitos "*Requirement*", "*FunctionalRequirement*", "*NonFunctionalRequirement*" e "*RequirementsDependency*" é similar àquela dada para os conceitos "*Concern*", "*FunctionalConcern*", "*NonFunctionalConcern*" e "*Dependency*" e, portanto, não será comentada novamente. Esse conjunto de conceitos é fundamental para se responder à *Questão de Competência QDC-2*.

3.3.6 *FunctionalCCC* e *NonFunctionalCCC*

Os conceitos "*FunctionalCCC*" e "*NonFunctionalCCC*" referem-se, respectivamente, a interesses transversais funcionais e não funcionais do software. Como uma instância de interesse ("*Concern*") pode ser transversal em um determinado projeto de software e deixá-lo de ser em outro, então esses conceitos foram classificados como "*Role*".

Conforme discutido na Seção 2.4.2 desta tese, "*Role*" refere-se a conceitos do tipo "*Anti-Rigid Sortal*", isto é, conceitos cujos indivíduos podem eventualmente instanciá-los ou não enquanto existirem, como é o caso de um interesse transversal. Mais especificamente, um conceito "*Role*" é relacionalmente dependente, ou seja, sua instanciação é determinada por uma propriedade relacional do indivíduo com outro indivíduo (Guizzard, 2005). Por exemplo, o indivíduo "João" instancia o papel (*role*) de "Esposo" se estiver "casado com (propriedade relacional)" o indivíduo "Maria".

No caso dos conceitos "*FunctionalCCC*" e "*NonFunctionalCCC*", um interesse instanciará algum desses papéis, caso esteja relacionado com um requisito por meio do relacionamento "*cut-across*", do tipo "*Material*". Em outras palavras, para se tornar um interesse transversal, um interesse precisa entrecortar um ou mais requisitos do software. É importante salientar que o relacionamento de entrecorte (*cut-across*) só existe caso o interesse entrecortante (transversal) não seja o interesse principal do requisito, conforme será explicado no Axioma 3, da Seção 3.3.8.

O conceito "*NonFunctionalCCC*" é bem conhecido pela comunidade de EROA e é relatado em vários dos trabalhos analisados (Chernak, 2012; Zheng *et al*., 2010; Alencar *et al*., 2010; Mussbacher *et al*., 2010; Agostinho *et al*., 2008; Chitchyan *et al*., 2006; Soeiro *et al*., 2006; Sampaio *et al*., 2005; Moreira *et al*., 2005; Whittle e Araújo, 2004; Brito e Moreira, 2003). Já o conceito "*FunctionalCCC*" foi extraído do trabalho de Moreira *et al*. (2005), que foi um dos primeiros trabalhos a afirmar que "entrecorte" é um fenômeno não limitado apenas a interesses não funcionais, ou seja, interesses funcionais podem também entrecortar outras partes do software. Exemplos de interesses transversais funcionais são "Gerenciamento de Pedidos", "Pagamento", entre outros. Essa classificação tem sido bem aceita por outros pesquisadores, que a tem utilizado para avaliação de abordagens para EROA, tais como Sampaio *et al*. (2007) e Herrera *et al*. (2012).

3.3.7 *Indication, MainConcernIndication, FunctionalCCCIndication* e *NonFunctionalCCCIndication*

Conforme comentado na Seção 3.3.6, para uma instância do "*Concern*" ser considerada "*FunctionalCCC*" ou "*NonFunctionalCCC*", ela deve estar relacionada com um ou mais requisitos do software por meio do relacionamento "*cut-across*". Também foi dito que "*cut-across*" é um relacionamento do tipo "*Material*". Assim, para que esse relacionamento de entrelaçamento se concretize, é necessário que haja um "*Relator*".

Neste caso, os conceitos "*FunctionalCCCIndication*" e "*NonFunctionalCCCIndication*", que são subconceitos de "*Indication*", representam esses conceitos mediadores (*relators*). Eles representam indícios de que há um entrelaçamento entre o interesse e o requisito em questão. Por exemplo, para que o interesse "Persistência" entrecorte o requisito "o sistema deve permitir o cadastramento de funcionários", deve haver algum indício desse entrecorte, que pode ser do tipo "*Keyword*" ou "*Requirement_Dependency*" (classe enumerada `IndicationType`). Em outras palavras, um interesse "A" pode entrecortar um determinado requisito "r" caso haja palavras-chave desse interesse que casem com a descrição de "r" ou caso haja um relacionamento de

dependência entre o requisito “r” (*source*) e um requisito “r1” (*target*), de forma que os interesses relacionados “A” entrecorte “r1”.

Esses dois conceitos não foram explicitamente formalizados nos trabalhos relacionados à EROA, apesar de alguns deles utilizarem essas ideias. Por exemplo, na abordagem *Theme/Doc* (Clarke e Baniassad, 2005; Baniassad e Clarke, 2004), diz-se que um interesse entrecorta um requisito quando há palavras-chave identificadas pelo engenheiro de software como referentes a esse interesse na descrição desse requisito. Contudo, *Theme/Doc* não contempla a ideia de relacionamentos de dependência entre interesses para aprimorar o processo de identificação e classificação dos interesses, o que torna essa abordagem altamente dependente do conjunto de palavras-chave elencadas pelo engenheiro de software, bem como da boa descrição dos requisitos do software.

Toda instância do conceito “*Requirement*” deve estar associada a uma e somente uma instância do conceito “*Concern*”, por meio do relacionamento “*mainConcern*”, do tipo “*Material*”. Esse relacionamento especifica que todo requisito de software deve estar relacionado ao seu interesse principal, isto é, o interesse que se refere à finalidade para a qual o requisito foi escrito. Por exemplo, o requisito “as senhas dos usuários devem ser armazenadas de forma criptografada” possui o interesse “Segurança” como seu interesse principal, uma vez que ele foi escrito com o intuito de especificar um comportamento relacionado com a segurança do software. Em outro exemplo, o requisito “o tempo de resposta das funções do software não deve ser superior a 5 (cinco) segundos” refere-se primariamente ao interesse “Desempenho”. Para concretizar o relacionamento entre um requisito e seu interesse principal, há o conceito “*MainConcernIndication*” do tipo “*Relator*”, que também é um subconceito de “*Indication*”.

O conceito de interesse principal não é explicitamente mencionado nos trabalhos relacionados à EROA, mas a ideia representada por esse conceito é utilizada em alguns trabalhos. Por exemplo, na abordagem *Theme/Doc* (Clarke e Baniassad, 2005; Baniassad e Clarke, 2004), quando mais de uma ação-chave está relacionada a um mesmo requisito, o engenheiro de software deve decidir qual é a ação principal (interesse principal) desse requisito. Por meio das regras de composição utilizadas nos trabalhos de *MDSoC* (Moreira *et al*., 2005), *EROA/XML* (Soeiro *et al*., 2006), *EROA/Arcade* (Rashid *et al*., 2003; Rashid *et al*., 2002) e *EA-Miner* (Chitchyan *et al*., 2006; Sampaio *et al*., 2005), o engenheiro de software especifica como um interesse e os requisitos para os quais ele é o interesse principal entrecortam os demais requisitos do software.

Os conceitos apresentados nas Seções 3.3.6 e 3.3.7 são úteis para se responder às *Questões de Competência QDC-3, QDC-4 e QDC-5*.

3.3.8 Axiomas

Os axiomas desenvolvidos para a ontologia *O4C* são apresentados e explicados nesta seção. A lista de predicados utilizados nestes axiomas, bem como a descrição dos mesmos podem ser visualizadas no Quadro 3.1.

Quadro 3.1. Predicados utilizados nos axiomas da ontologia *O4C*.

Predicado	Descrição
`hasSubconcern(c1, c2)`	O interesse “c2” é subinteresse de “c1”.
`hasKw(c, kw)`	O interesse “c” possui “kw” como uma de suas palavra-chave.
`cutAcross(c, r)`	O interesse “c” entrecorta o requisito “r”.
`mainConcern(r, c)`	O requisito “r” possui “c” como seu interesse principal.
`dependsOn(r1, r2)`	O requisito “r1” depende do requisito “r2”.
`isRelatedTo(rel, c1, c2)`	O interesse “c1” está relacionado com o interesse “c2” por meio do relacionamento “rel”, tal que “c1” é o interesse fonte e “c2” é o interesse alvo.
`isAConcernDependency(rel)`	O relacionamento “rel” é do tipo “*Dependency*”.
`diffConcerns(c1, c2)`	Os interesses “c1” e “c2” são diferentes.
`hasIndication(ind, c)`	Há um indício “ind” de que o requisito “c” está presente no software em análise.

Axioma 1:
$$(\forall\, c1, c2, kw) : (hasSubConcern(c1, c2) \wedge hasKw(c2, kw) \rightarrow hasKw(c1, kw))$$

O Axioma 1 é um axioma de derivação e especifica que, dados dois interesses “c1” e “c2”, sendo “c2” um subinteresse de “c1”, se “kw” é uma palavra-chave de “c2”, então “kw” também é uma palavra-chave de “c1”.

Axioma 2:
$$(\forall\, c1, c2, r): (hasSubConcern(c1, c2) \wedge cutAcross(c2, r) \rightarrow cutAcross(c1, r))$$

O Axioma 2 (de derivação) especifica que, se um interesse “c2” entrecorta o requisito “r”, sendo “c2” um subinteresse de “c1”, então “c1” também entrecorta o requisito “r”.

Axioma 3:
$$(\forall\, c1, r, mc): (cutAcross(c1, r) \wedge mainConcern(r, mc) \rightarrow diffConcerns(c, mc))$$

O Axioma 3 é um axioma de consolidação, que especifica que um interesse “c1” só pode entrecortar um requisito “r”, caso ele não seja o seu interesse principal “mc”. A justificativa é que se “c1” é o interesse principal do requisito “r”, então ele não exerce comportamento transversal sobre o mesmo.

Axioma 4:
(∀ r1, r2, c1):
(dependsOn(r1, r2) ^
cutAcross(c1, r2) → cutAcross(c1, r1)
)

O Axioma 4 (de derivação) especifica que, se um requisito “r1” depende do requisito “r2” e “r2” é entrecortado por “c1”, então “r1” também é entrecortado por “c1”.

Axioma 5:
(∀r1, r2, mc1, mc2):
(dependsOn(r1, r2) ^
mainConcern(r2, mc2) ^
mainConcern(r1, mc1) ^
diffConcerns(mc1, mc2) → cutAcross(mc2, r1)
)

O Axioma 5 (de derivação) especifica que se um requisito “r1” depende do requisito “r2”, sendo “mc1” o interesse principal de “r1”, “mc2” o interesse principal de “r2” e “mc1” diferente de “mc2”, então “r2” é entrecortado por “mc2”. Em outras palavras, isso significa que o interesse principal de um requisito pode ser um interesse transversal de outro requisito. Por exemplo, considere os requisitos “r1: o software deve permitir a emissão da segunda via da conta do cliente, por meio do seu número de controle” e “r2: o custo da emissão de uma segunda via da conta de telefone deve ser cobrado na próxima fatura do cliente”, sendo que “r1” depende de “r2”. Considerando que “Gerenciamento de Nota Fiscal” seja o interesse principal de “r1” e “Faturamento” o interesse principal de “r2”, segundo o Axioma 5, “Faturamento” entrecorta o interesse “Gerenciamento de Nota Fiscal” no requisito “r1”.

Axioma 6:
(∀ rel, c1, c2, ind1, ind2):
(isRelatedTo(rel, c1, c2) ^
isAConcernDependecy(rel) ^
hasIndication(ind1, c1) → hasIndication(ind2, c2)
)

O Axioma 6 (de derivação) especifica que se há dois interesse “c1” e “c2”, tal que “c1” depende de “c2” e “c1” foi identificado no software, ou seja, há algum indício “ind1” de que “c1” está presente no software em análise, então deve haver algum indício “ind2” de que o interesse “c2” também está presente no software.

3.4 Considerações Finais

Neste capítulo apresentou-se a ontologia para interesses de software no contexto da EROA, denominada *O4C*. Tal ontologia foi utilizada como base para a construção da abordagem para identificação e classificação de interesses de software *ObasCId* e da ferramenta *ObasCId-Tool*, apresentadas nos Capítulos 4 e 5 desta tese, respectivamente.

A ontologia *O4C* foi simbolicamente dividida em duas partes A e B, sendo que a parte A apresenta conceitos independentes de contexto, isto é, que podem ser instanciados para formar uma base de conhecimento comum sobre interesses de software. Ao instanciar a parte A da ontologia *O4C*, o pesquisador estará elaborando catálogos para tipos específicos de interesses de software, descrevendo com quais palavras-chave esses interesses se relacionam, como eles dependem de outros interesses, entre outras coisas.

Assim, com a *O4C*, os pesquisadores poderão construir um arcabouço de catálogos de interesses que poderão ser (re)utilizados como apoio para facilitar a identificação de interesses transversais. No Capítulo 4 desta tese, são apresentados alguns exemplos de catálogos de interesses elaborados a partir de fontes diversas, como catálogos de requisitos não funcionais e projetos anteriores, bem como um conjunto de passos para utilização destes catálogos durante a identificação e classificação de interesses a partir de requisitos de software.

A parte B da ontologia *O4C* descreve conceitos que dependem do projeto de software sobre o qual as instâncias dos conceitos da parte A da ontologia estão sendo aplicadas. Esses conceitos incluem os requisitos do software e seus relacionamentos, dentre outras informações. Apesar de esses conceitos serem específicos para cada projeto, eles podem ser úteis para atualização das instâncias de conceitos da parte A da ontologia. Por exemplo, durante a identificação de interesses em um determinado software, pode-se notar a necessidade de atualizar o conjunto de palavras-chave de um interesse para aprimorar sua cobertura no software.

No Quadro 3.2 estão resumidos os principais conceitos e relacionamentos definidos na *O4C*, destacando-se os trabalhos relacionados à EROA que propõem alguma forma de representação dos mesmos. O símbolo "-", quando utilizado, representa que o conceito/relacionamento não foi explorado por um determinado trabalho relacionado.

Como pode ser visto no Quadro 3.2, muitas das abordagens contemplam apenas os conceitos "*Concern*", "*NonFunctionalConcern*", "*NonFunctionalConcern*" e "*NonFunctionalCCC*". Sendo assim, há carência de abordagens que propõem formas de representação para conceitos referentes aos tipos de requisitos de um software ("*FunctionalRequirement*" e "*NonFunctionalRequirement*"), aos tipos de relacionamentos

existentes entre interesses (tais como "*Composition*"), aos interesses e suas palavras-chave ("*Keyword*"), entre requisitos ("*RequirementDependency*") e aos interesses e requisitos ("*Indication*", "*MainConcernIndication*", "*FunctionalCCCIndication*" e "*NonFunctionalCCCIndication*").

Destaca-se ainda que não há abordagens que contemplam todos os conceitos descritos na ontologia *O4C*. As duas abordagens que apresentam maior quantidade de conceitos em comum com a ontologia *O4C* (oito conceitos) são *MDSoC* (Moreira *et al*., 2005) e *EROA/XML* (Soeiro *et al*., 2006).

Quadro 3.2. Trabalhos relacionados aos conceitos da ontologia *O4C*.

Conceito da *O4Cl* Trabalhos relacionados	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**
Concern	X	X	X	X	X	X	X	X	X	X	X	X	X
NonFunctionalConcern	X	X	X	X	X	X	X	X	X	X	X	X	X
FunctionalConcern	X	X	X	X	X	X	X	X	X	X	X	X	X
Keyword	-	-	-	-	-	-	-	-	-	-	-	-	-
Source	X	X	X	-	-	-	-	-	-	-	-	-	-
Composition	-	-	-	-	-	-	-	-	-	-	-	-	-
Dependency	-	-	-	-	X	-	-	-	-	-	-	-	-
Contribution	-	-	-	X	X	-	-	-	-	-	-	-	-
Priority	-	-	-	X	X	-	-	-	-	-	-	-	-
Requirement	-	-	-	X	X	X	X	X	X	-	-	-	-
FunctionalRequirement,	-	-	-	-	-	-	-	-	-	-	-	-	-
NonFunctionalRequirement	-	-	-	-	-	-	-	-	-	-	-	-	-
RequirementsDependency	-	-	-	-	-	-	-	-	-	-	-	-	-
NonFunctionalCCC	X	X	X	X	X	X	X	X	X	X	X	X	X
FunctionalCCC	-	-	-	X	-	-	-	-	-	-	-	-	-
Indication	-	-	-	-	-	-	-	-	-	-	-	-	-
FunctionalCCCIndication	-	-	-	-	-	-	-	-	-	-	-	-	-
NonFunctionalCCCIndication	-	-	-	-	-	-	-	-	-	-	-	-	-
MainConcernIndication	-	-	-	-	-	-	-	-	-	-	-	-	-

Legenda: 1. Agostinho *et al*., 2008; 2. Whittle e Araújo, 2004; 3. Brito e Moreira, 2003; 4. Moreira *et al*., 2005; 5. Soeiro *et al*., 2006; 6. Chitchyan *et al*., 2006; 7. Sampaio *et al*., 2005; 8. Clarke e Baniassad, 2005; 9. Baniassad e Clarke, 2004; 10. Chernak, 2012; 11. Zheng *et al*., 2010; 12. Alencar *et al*., 2010; 13. Mussbacher *et al*., 2010

Capítulo 4

ObasCId: uma Abordagem Ontologicamente Fundamentada para EROA

4.1 Considerações Iniciais

O aumento da complexidade do software e a sua aplicabilidade nas mais diversas áreas requerem que a engenharia de requisitos seja realizada de modo abrangente e completo. Isso deve ocorrer para que todas as necessidades dos *stakeholders* sejam contempladas, bem como para permitir que os engenheiros de software tenham um entendimento mais amplo a respeito da funcionalidade do software, das restrições impostas sobre ele e do ambiente sobre o qual o software deve operar (Sampaio *et al*., 2007).

Este capítulo apresenta a abordagem para identificação e classificação de interesses de software desenvolvida neste trabalho, denominada *ObasCId* (*Ontologically-based Concern Identification*) (Parreira Júnior e Penteado, 2015b; Parreira Júnior e Penteado, 2015c). Na Seção 4.2 é apresentada uma visão geral da abordagem *ObasCId*, destacando suas principais fases, artefatos consumidos e gerados/atualizados. Nas Seções 4.3 à 4.6 são detalhadas as atividades de cada fase dessa abordagem, ilustrando-as por meio de exemplos de uso das mesmas. Por fim, na Seção 4.7 estão as considerações finais deste capítulo.

4.2 Visão Geral da Abordagem *ObasCId*

Para apresentar a abordagem proposta neste trabalho de forma gráfica, utilizou-se um conjunto de ícones especiais, derivados do meta-modelo para engenharia de processos de software *SPEM* (*Software Process Engineering Metamodel*) (SPEM, 2008). No Quadro 4.1 estão listados os ícones utilizados, bem como a descrição de cada um deles.

Quadro 4.1. Ícones do meta-modelo *SPEM* utilizados para representação gráfica da abordagem *ObasCId*.

Estereótipo/Símbolo	Descrição
 Fase	Representam as fases da abordagem proposta.
 Atividade	Representam as atividades que são realizadas em cada fase da abordagem proposta.
 Artefato	Representam os artefatos produzidos/consumidos pelas atividades da abordagem.

A abordagem *ObasCId* é composta das seguintes fases (Figura 4.1):

i) **Preparação do Catálogo de Interesses de Software**: tem a responsabilidade de obter/elaborar/atualizar, com base nos conceitos da ontologia *O4C*, um catálogo de interesses de software que será utilizado nas demais fases da abordagem *ObasCId*;

ii) **Preparação do Documento de Requisitos do Software**: consiste em obter/elaborar/atualizar o documento de requisitos sobre o qual ocorrerá a identificação e classificação de interesses;

iii) **Identificação e Classificação de Interesses**: visa a identificar os interesses representados no catálogo de interesses sobre os requisitos preparados na fase anterior. Uma vez identificados, os mesmos são classificados quanto à sua natureza base ou transversal; e

iv) **Detecção de Conflitos entre Interesses**: busca por possíveis conflitos entre os interesses identificados na fase anterior, com base na influência mútua existente entre eles, registrada no catálogo de interesses.

A lista de artefatos consumidos e gerados/atualizados e sua classificação quanto a ser obrigatório ou opcional em cada fase da abordagem é apresentada no Quadro 4.2 e comentada ao longo desta seção. Esses artefatos são novamente evidenciados nas seções específicas para cada fase da abordagem *ObasCId*.

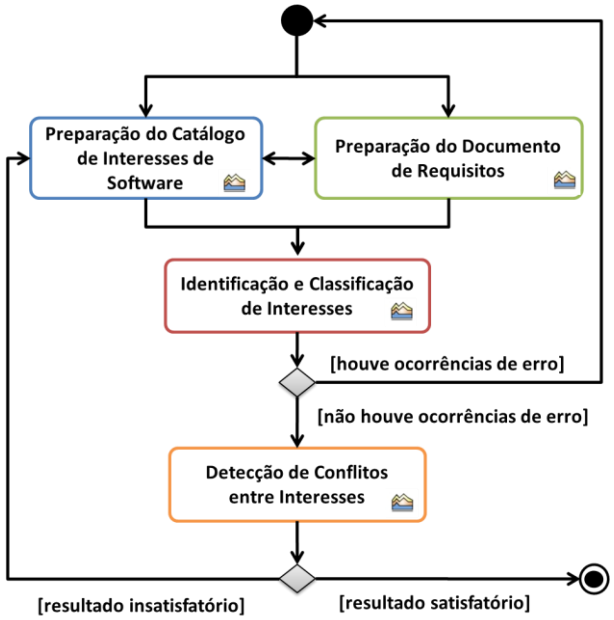


Figura 4.1. Representação gráfica da abordagem *ObasCId*.

Quadro 4.2. Listagem dos artefatos consumidos e gerados/atualizados pelas fases da abordagem *ObasCId*.

Artefatos Consumidos	Fase	Artefatos Gerados/Atualizados
• Repositório de catálogos de interesses de software (**obrigatório**) • Fontes de interesses de software, *e. g*. catálogo de requisitos não funcionais, documentos de negócios, entre outros (**opcional**) • Lista de ocorrências (**opcional**)	**Preparação do Catálogo de Interesses de Software**	• Catálogo de interesses de software (**obrigatório**) • Repositório de catálogos de interesses de software (**opcional**)
• Repositório de documentos de requisitos (**obrigatório**) • Lista de ocorrências (**opcional**)	**Preparação do Documento de Requisitos**	• Documento de requisitos (**obrigatório**) • Repositório de documentos de requisitos (**opcional**)
• Catálogo de interesses de software (**obrigatório**) • Documento de requisitos (**obrigatório**)	**Identificação e Classificação de Interesses**	• Lista de requisitos e interesses identificados (**obrigatório**) • Matriz de entrelaçamentos entre interesses (**obrigatório**) • Lista de ocorrências (**opcional**)
i) Catálogo de interesses de software (**obrigatório**) ii) Matriz de entrelaçamentos entre interesses (**obrigatório**)	**Detecção de Conflitos entre Interesses**	iii) Matriz de contribuições entre interesses (**obrigatório**)

Como pode ser visto na Figura 4.1, as fases “Preparação do Catálogo de Interesses de Software” e “Preparação do Documento de Requisitos” da abordagem *ObasCId* são

independentes entre si. Dessa forma, apesar de ser necessário executar as duas fases antes de prosseguir para a próxima, o engenheiro de software pode iniciar o uso da abordagem por qualquer uma dessas fases e pode ir de uma para a outra a qualquer momento do processo.

A fase "Preparação do Catálogo de Interesses de Software" faz uso, obrigatoriamente, do repositório de catálogos de interesses de software, pois todas as suas atividades dependem de forma direta desse artefato. Opcionalmente, a depender da atividade que estiver sendo realizada pelo engenheiro de software, fontes de interesses de software (como por exemplo, catálogos de requisitos não funcionais, documentos de negócios, entre outros) e uma lista de ocorrências gerada pela fase "Identificação e Classificação de Interesses" também podem ser utilizadas como artefatos de entrada desta fase. Como artefatos de saída, essa fase deve gerar/atualizar o catálogo de interesses de software que será utilizado posteriormente pela abordagem *ObasCId*. Além disso, opcionalmente, o repositório de catálogos de interesses de software poderá ser modificado após a finalização desta fase, incluindo novos catálogos ou atualizando algum catálogo pré-existente.

A fase "Preparação do Documento de Requisitos" tem como artefato de entrada obrigatório o repositório de documentos de requisitos e gera/atualiza, também obrigatoriamente, o documento de requisitos que será utilizado durante a identificação e classificação de interesses. Como artefato opcional de entrada, tem-se uma lista de ocorrências relacionadas à identificação e classificação de interesses. Também opcionalmente, o repositório de documentos de requisitos pode ser atualizado.

Para que o engenheiro de software possa iniciar a fase "Identificação e Classificação de Interesses", ele deve ter finalizado as duas fases anteriores. Isso porque essa fase depende, obrigatoriamente, de dois artefatos gerados pelas fases anteriores, a saber, o catálogo de interesses de software e o documento de requisitos. Como artefatos de saída, essa fase gera: (i) uma lista de requisitos e interesses identificados (obrigatório); (ii) uma matriz de entrelaçamentos entre interesses (obrigatório); e (iii) uma lista de ocorrências relacionadas à identificação e classificação de interesses (opcional). Após a execução dessa fase, pode ser necessário retornar para alguma das fases anteriores, com o intuito de atualizar o catálogo de interesses de software e/ou o documento de requisitos do software em análise. Isso ocorre devido à existência de erros na lista de ocorrências gerada nesta fase. Os tipos de ocorrências que podem acontecer, bem como as recomendações para resolução das mesmas são descritas na Seção 4.5.4.

A fase "Detecção de Conflitos entre Interesses" deve ser executada após a finalização da fase "Identificação e Classificação de Interesses", uma vez que ela faz uso do artefato gerado "matriz de entrelaçamentos entre interesses". Além desse artefato, essa fase

requer ainda o catálogo de interesses de software, que é utilizado com a finalidade de se recuperar os relacionamentos de contribuição existentes entre os interesses identificados no software. Como saída, essa fase gera uma matriz de contribuição, destacando a influência mútua existente entre interesses identificados no software.

A descrição detalhada de cada fase da abordagem *ObasCId*, destacando-se suas principais atividades, os artefatos gerados e consumidos e o fluxo em que essas atividades ocorrem, é apresentada nas Seções 4.3 à 4.6.

4.3 Preparação do Catálogo de Interesses de Software

Essa fase tem como finalidade gerenciar (obter/elaborar/atualizar) o catálogo de interesses que será utilizado durante a atividade de identificação e classificação de interesses de software (Figura 4.2). Caso haja catálogos pré-existentes no repositório de catálogos de interesses de software, esses podem ser obtidos por meio da atividade "Obter catálogo de interesses de software".

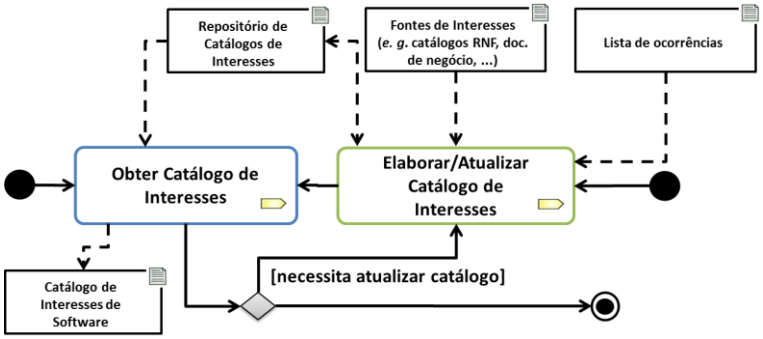


Figura 4.2. Atividades da fase "Preparação do Catálogo de Interesses de Software".

A atividade "Elaborar/atualizar catálogo de interesses de software" pode ser executada quando o engenheiro de software deseja elaborar seu próprio catálogo de interesses ou quando deseja atualizar um catálogo pré-existente. A atualização de um catálogo pode ser motivada pela ocorrência de problemas durante as fases de "Identificação e Classificação de Interesses" e "Detecção de Conflitos entre Interesses" ou pelo próprio desejo do engenheiro de software em modificar os elementos desse catálogo.

É importante ressaltar que catálogos de interesses de software são gerados a partir dos conceitos da parte A da ontologia *O4C* (Figura 3.1), correspondendo assim, a instâncias desses conceitos. Por exemplo, a ontologia *O4C* descreve o conceito "*NonFunctionalConcern*", que representa o tipo de interesse de software conhecido como interesse não funcional. Em um catálogo de interesses de software, haverá instâncias desse

conceito para tipos específicos de interesses não funcionais, tais como “Segurança”, “Persistência”, entre outros.

Um exemplo simples de catálogo para os interesses “Segurança”, “Autorização” e “*Logging*” pode ser visto na Figura 4.3. Para facilitar a visualização desse e dos demais catálogos de interesses apresentados nesta seção, instâncias dos conceitos “*Concern*”, “*Relationship*” e “*Keyword*” foram representadas por meio de classes da UML destacadas em cinza, azul e amarelo, respectivamente. Os nomes dos conceitos da ontologia *O4C* são representados por meio de estereótipos localizados acima dos nomes das classes.

Nessa figura, há seis instâncias de conceitos, três referentes ao conceito “*NonFunctionalConcern*”, duas referentes ao conceito “*Keyword*” e uma referente ao conceito “*Composition*”. Neste exemplo, os interesses não funcionais são “*Logging*”, “Segurança” e “Autorização”, sendo que “Autorização” é um subinteresse de “Segurança”. Além disso, o interesse “*Logging*” está relacionado com duas palavras-chave, a saber, “*logged*” e “*log*”. Os relacionamentos entre as instâncias desses conceitos são representados por meio de associações do diagrama de classes UML.

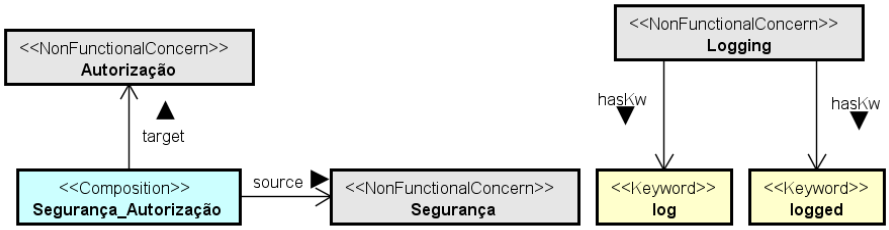


Figura 4.3. Exemplo simples de um catálogo de interesses de software.

Catálogos de interesses de software podem ser criados a partir: (i) de outros tipos de *catálogos*, como catálogos de requisitos não funcionais (Cysneiro, 2015; Chung e Leite, 2000; Boehm e In, 1996); (ii) do *conhecimento de especialistas* no domínio desses interesses; (iii) de *documentos de* negócios, tais como protocolos de segurança e privacidade, linguagens de padrões de negócio, tais como a linguagem para Gestão de Recursos de Negócio (GRN) (Braga, 2002), entre outros; ou (iv) de *dados históricos* de projetos já concluídos, para os quais os interesses do software já foram devidamente identificados por especialistas.

Na Figura 4.4 é apresentado o trecho de um catálogo de interesses criado com base em dados históricos. Mais especificamente, esse catálogo foi desenvolvido com base nos resultados da identificação e classificação de interesses do software *Health Watcher* (2015), realizada por especialistas em EROA com conhecimento sobre o domínio do software em questão. *Health Watcher* é uma aplicação *web* para registro e controle de reclamações na área da saúde bem conhecida pela comunidade científica de Desenvolvimento de Software Orientado a Aspectos (DSOA). Para melhorar a representatividade da Figura 4.4, apenas

interesses não funcionais do software *Health Watcher* foram considerados. Já a Figura 4.5 apresenta um trecho do catálogo para interesses funcionais desse software.

É importante salientar que a atividade de identificação e classificação dos interesses do software *Health Watcher* (2015) foi realizada com base na abordagem *MDSoC* (Moreira *et al*., 2005) e, conforme foi discutido nas Seções 2.2.1 e 3.3.2. O arquivo XML (*template*) proposto pelos autores dessa abordagem para representação de interesses não contempla o conceito de palavras-chave. Sendo assim, tais elementos tiveram que ser elicitados por especialistas da área de EROA, a partir da descrição dos requisitos com os quais os interesses identificados estavam relacionados. O catálogo da Figura 4.4 apresenta 7 (sete) interesses do tipo não funcional, relacionados com 28 (vinte e oito) palavras-chave. Além disso, há dois relacionamentos de contribuição (“*Contribution*”), sendo uma contribuição positiva entre os interesses “Concorrência” e “Desempenho” e uma contribuição negativa entre os interesses “Segurança” e “Desempenho”.

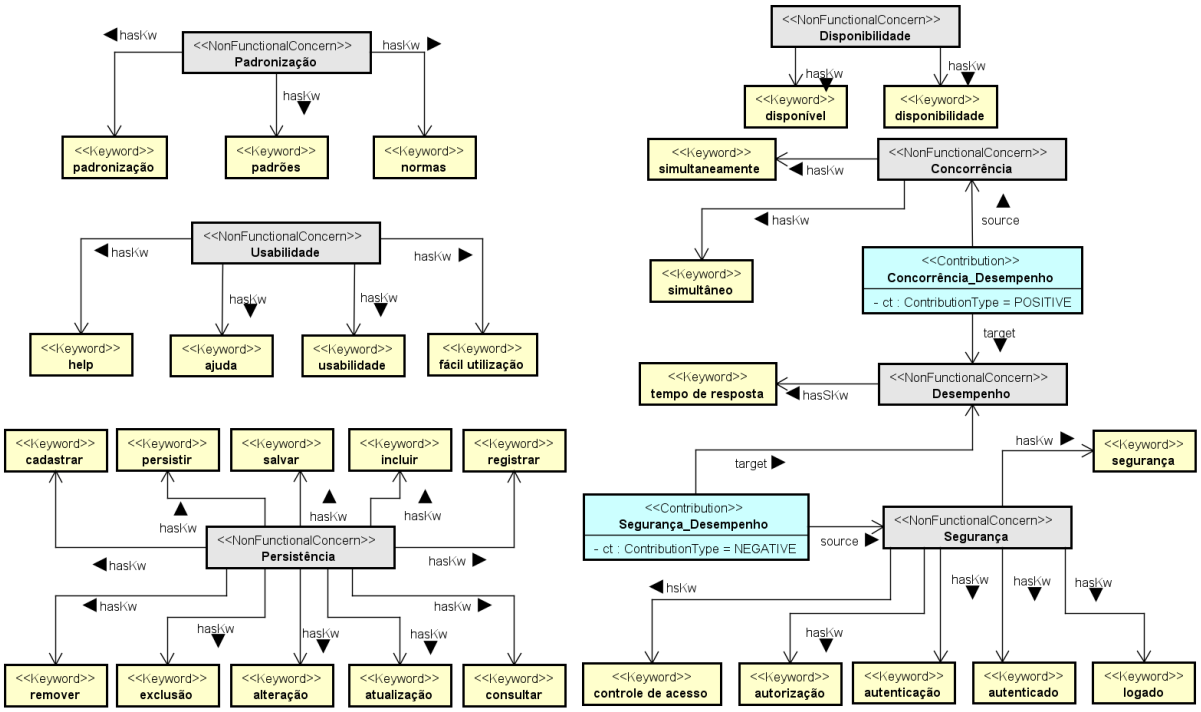


Figura 4.4. Catálogos para interesses não funcionais, desenvolvido a partir de dados históricos do software *Health Watcher*.

Ainda com base nos resultados da identificação e classificação de interesses do software *Health Watcher* (2015), na Figura 4.5 é apresentado um trecho de um catálogo da ontologia *O4C* com interesses funcionais identificados nesse software. Esse catálogo apresenta 3 (três) interesses do tipo funcional, 6 (seis) palavras-chave e 2 (dois) relacionamentos de composição entre os interesses “Reclamação” e “ReclamaçãoSobreAnimais” e “ReclamaçõesSobreAlimentos”.

A ideia é que o catálogo da Figura 4.4 possa ser utilizado para identificação e classificação de interesses não funcionais em outros projetos de softwares, assim como foi

utilizado no Capítulo 5 para identificação dos interesses não funcionais da ferramenta *ObasCId-Tool*. Analogamente, o catálogo da Figura 4.5 pode ser aplicado sobre requisitos de software do mesmo domínio do *Health Watcher*, isto é, o registro e controle de reclamações na área da saúde.

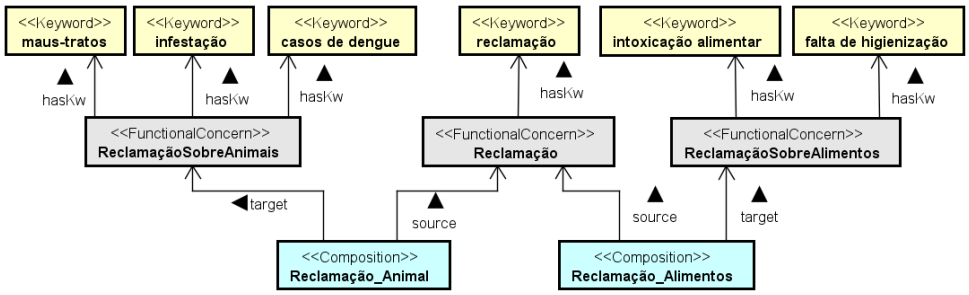


Figura 4.5. Catálogo para interesses funcionais, desenvolvido a partir de dados históricos do software *Health Watcher*.

Outro exemplo de catálogo de interesses de software pode ser visto na Figura 4.6. Ele foi construído a partir dos conceitos representados na linguagem de padrões Gestão de Recursos de Negócios (GRN) (Braga, 2002). A linguagem GRN foi elaborada para auxiliar o desenvolvimento de sistemas de informação no domínio de locação, comercialização e manutenção de recursos de negócios. Nessa linguagem, há um grupo de sete padrões relacionados às principais funcionalidades de um sistema de informações para gestão de recursos. Tais padrões foram mapeados para instâncias do conceito "*FunctionalConcern*" da ontologia *O4C*, gerando-se o catálogo de interesses da Figura 4.6.

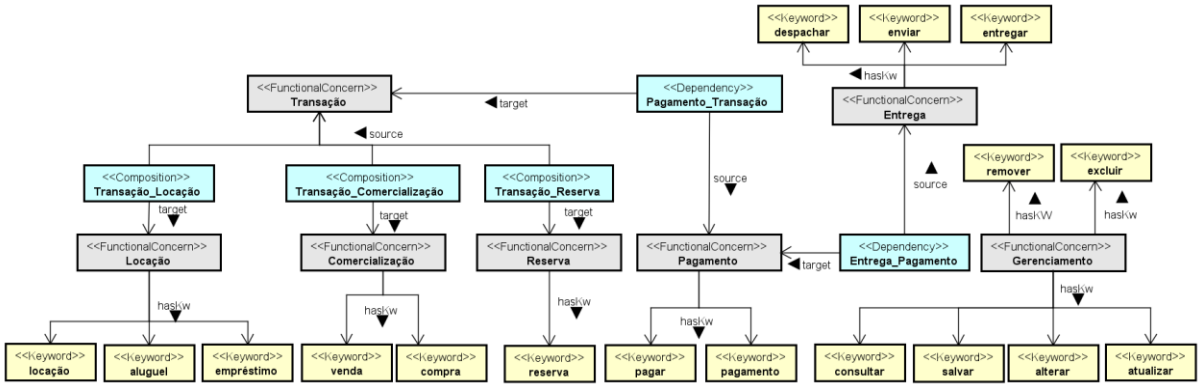


Figura 4.6. Catálogo de interesses de software desenvolvido a partir da linguagem de padrões para Gestão de Recursos de Negócios (GRN).

Esse catálogo possui 7 (sete) interesses funcionais, 17 (dezessete) palavras-chave e 5 (cinco) relacionamentos, sendo três deles de composição entre os interesses "Transação" e "Locação", "Transação" e "Comercialização" e "Transação" e "Reserva" e dois de dependência entre os interesses "Pagamento" e "Transação" e "Entrega" e "Pagamento". Os relacionamentos existentes entre os interesses, bem como as palavras-chave relacionadas aos mesmos, foram obtidas a partir da descrição dos padrões da linguagem GRN (Braga, 2002).

Ao combinar os interesses dos catálogos da Figura 4.4 e da Figura 4.6, pode-se gerar um catálogo mais amplo que pode ser utilizado para identificar tanto interesses funcionais de sistemas de informação para gestão de recursos, como também interesses não funcionais, tais como “Segurança”, “Persistência”, entre outros.

Nas Seções 4.5 e 4.6 é apresentado como esses catálogos de interesses de software, juntamente com um conjunto de passos para utilizá-los, podem auxiliar o engenheiro de software durante as atividades “Identificação e Classificação de Interesses” e “Detecção de Conflitos entre Interesses”.

4.4 Preparação do Documento de Requisitos

Analogamente ao que foi descrito na fase “Preparação do Catálogo de Interesses de Software”, essa fase tem como finalidade obter/elaborar/atualizar o documento de requisitos de software que será utilizado durante a atividade de identificação e classificação de interesses. Caso haja documentos pré-existentes, o engenheiro de software pode reutilizá-los, atualizando-os, caso necessário. O diagrama da Figura 4.7 apresenta as principais atividades dessa fase, bem como os artefatos consumidos e gerados/atualizados pelas mesmas.

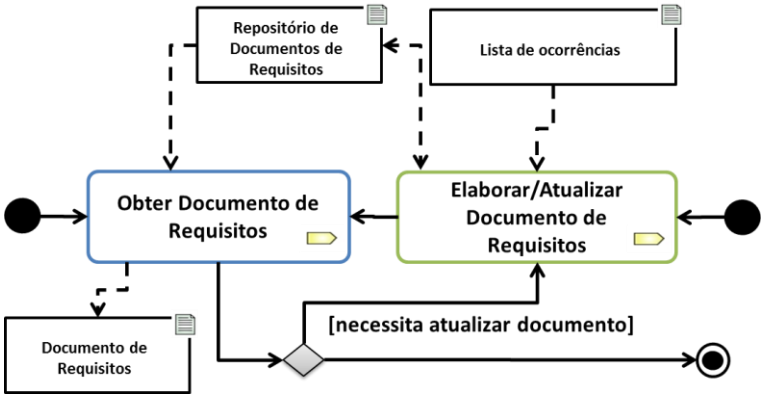


Figura 4.7. Atividades da fase “Preparação do Documento de Requisitos”.

O modelo (*template*) do documento de requisitos adotado pela abordagem *ObasCId* é baseado em uma listagem dos requisitos do software, cujas principais informações são o código do requisito, seu tipo (funcional e não-funcional), sua descrição em texto plano (*plain-text*) e a lista de outros requisitos dos quais ele depende, conforme exemplificado no Quadro 4.3. Esse modelo de requisito está em conformidade com os conceitos existentes na ontologia *O4C*, conforme discutido no Capítulo 3.

Quadro 4.3. Modelo de documento de requisitos adotado pela abordagem *ObasCId*.

Cód.	Tipo	Descrição do Requisito	Requisitos Relacionados
1	RF	Esta é a descrição do requisito funcional 1.	2
2	RNF	Esta é a descrição do requisito não funcional 2.	-
...	...	...	...
<N>	<tipo>	Está é a descrição do requisito <tipo> <*N*>.	-
Legenda: Requisito Funcional (RF); Requisito Não Funcional (RNF)			

No Quadro 4.4 e no Quadro 4.5 estão ilustrados alguns trechos de documentos de requisitos, especificados de acordo com o modelo do Quadro 4.3; o primeiro refere-se a um trecho do documento de requisitos do software *Health Watcher* (2015) e o segundo do software para gerenciamento de cursos apresentado por Baniassad e Clarke (2004).

No Quadro 4.4 são apresentados dois requisitos não funcionais e um requisito funcional, sendo que esse requisito funcional depende dos outros dois. Já no Quadro 4.5 são apresentados três requisitos funcionais e um não funcional, sendo que o requisito funcional “RF-01” depende do requisito não funcional “RNF-01”.

Quadro 4.4. Trecho do documento de requisitos do software *Health Watcher* (2015).

Cód.	Tipo	Descrição do Requisito	Requisitos Relacionados
RF-01	RF	Tem como propósito realizar a atualização do andamento de uma reclamação. A reclamação deve estar cadastrada, com a situação ABERTA e com o funcionário logado no sistema.	RNF-02, RNF-03
...	**...**	**...**	**...**
RNF-02	RNF	O sistema deve ter uma interface de fácil utilização, visto que o sistema pode ser utilizado por qualquer pessoa que tenha acesso a Internet. O sistema deve ter um HELP on-line para ser consultado por qualquer pessoa que acesse o sistema.	-
RNF-03	RNF	O tempo de resposta do sistema não deve ultrapassar 5 segundos por acesso.	-
Legenda: Requisito Funcional (RF); Requisito Não Funcional (RNF)			

Quadro 4.5. Trecho do documento de requisitos de um software para gerenciamento de cursos (adaptado de Baniassad e Clarke, 2004).

Cód.	Tipo	Descrição do Requisito	Requisitos Relacionados
RF-01	RF	Estudantes podem se matricular em cursos.	RNF-01
RF-02	RF	Estudantes podem cancelar suas matrículas em cursos.	-
RF-03	RF	Professores podem atribuir notas aos estudantes de seus cursos.	-
RFN-04	RNF	Quando um estudante se matricular em um curso, um registro (*log*) desse evento deve ser armazenado no sistema.	-
Legenda: Requisito Funcional (RF); Requisito Não Funcional (RNF)			

4.5 Identificação e Classificação de Interesses

O uso de catálogos de interesses de software construídos com base na ontologia *O4C*, juntamente com um conjunto de passos (diretrizes) para utilização dos mesmos, pode aprimorar a efetividade da atividade de identificação e classificação de interesses a partir de requisitos de software. Assim, uma vez tendo preparado o catálogo de interesses e o documento de requisitos, o engenheiro de software poderá passar para a fase de identificação e classificação de interesses, que é dividida em cinco atividades (Figura 4.8): (i) identificar interesses a partir de palavras-chave; (ii) identificar interesses a partir da interdependência entre requisitos do software; (iii) especificar interesses principais; (iv) verificar resultados da identificação de interesses e (v) classificar interesses. Cada uma dessas atividades é descrita com mais detalhes nesta seção.

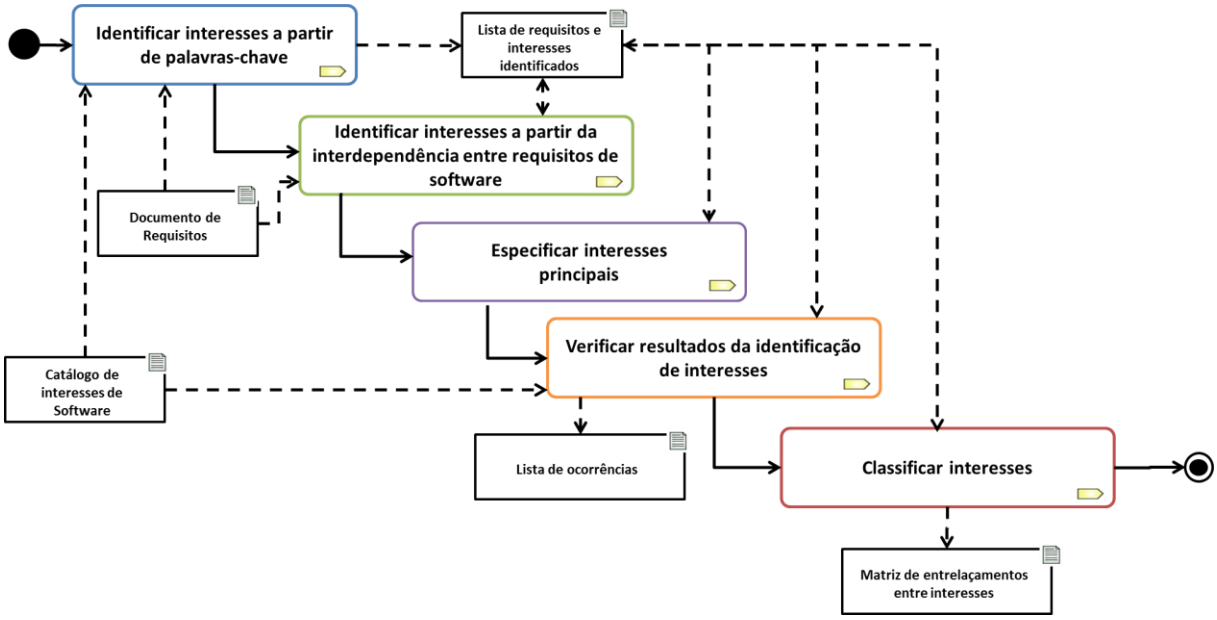


Figura 4.8. Atividades da fase "Identificação e Classificação de Interesses".

4.5.1 Identificar interesses a partir de palavras-chave

Esta atividade identifica interesses do software a partir das palavras-chave cadastradas para os interesses do catálogo selecionado pelo engenheiro de software nas fases anteriores dessa abordagem. Isso é feito por meio da busca das palavras-chave de cada interesse do catálogo sobre a descrição dos requisitos do software. Caso alguma das palavras-chave de um determinado interesse apareça na descrição de um requisito de software, diz-se que esse interesse está relacionado com o requisito em questão.

Essa atividade recebe como artefatos de entrada o catálogo de interesses de software e o documento de requisitos, e gera como saída uma lista de requisitos e interesses relacionados, isto é, uma lista em que, para cada requisito, há uma relação dos interesses identificados para o mesmo.

Considerando os requisitos do Quadro 4.4 e os catálogos de interesses de software da Figura 4.4 e da Figura 4.5, após a execução da atividade "identificar interesses a partir de palavras-chave" da abordagem *ObasCId*, tem-se como saída a lista de requisitos e interesses identificados apresentada no Quadro 4.6. As palavras-chave que permitiram a identificação dos interesses de cada requisito são destacadas no Quadro 4.6. A coluna "Principal" do Quadro 4.6 será explicada na Seção 4.5.3 deste capítulo.

Quadro 4.6. Exemplo de uma lista de requisitos e interesses identificados.

Requisito RF-01	**Interesses**	**Principal**
Tem como propósito realizar a **atualização** do andamento de uma **reclamação**. A **reclamação** deve estar cadastrada, com a situação ABERTA e com o funcionário **logado** no sistema.	Persistência	
	Segurança	
	Reclamação	
Requisito RNF-02	**Interesses**	**Principal**
O sistema deve ter uma interface de **fácil utilização**, visto que o sistema pode ser utilizado por qualquer pessoa que tenha acesso a Internet. O sistema deve ter um **HELP** on-line para ser consultado por qualquer pessoa que acesse o sistema.	Usabilidade	
Requisito RNF-03	**Interesses**	**Principal**
O **tempo de resposta** do sistema não deve ultrapassar 5 segundos por acesso.	Desempenho	

A Listagem 4.1 apresenta o procedimento para construção da lista de requisitos e interesses relacionados, conforme comentado anteriormente. A ideia é que, na descrição de cada requisito "r" do documento de requisitos, procura-se por palavras-chave dos interesses cadastrados no catálogo de interesses. Caso a quantidade de ocorrências de palavras-chave de um interesse "c" do catálogo em determinado requisito "r" seja maior do que 0 (zero), então esse interesse "c" é incluído na lista de interesses identificados para o requisito "r" em questão.

```
R ← lista de requisitos do software em análise
C ← lista de interesses da instância de ontologia utilizada
Para cada requisito r da lista R
    Para cada interesse c da lista C
       #pc ← quantidade de ocorrências de palavras-chave do interesse c
             na descrição do requisito r
       Se #pc > 0, então
           adicionar c na lista de interesses de r
       Fim se
    Fim para
Fim para
```

Listagem 4.1. Procedimento para construção da lista de requisitos e interesses identificados da abordagem *ObasCId*.

4.5.2 Identificar interesses a partir da interdependência entre requisitos de software

Esta atividade é responsável por identificar outros tipos de interesses para um determinado requisito, que não puderam ser identificados apenas por meio do uso de palavras-chave. Essa atividade faz uso do documento de requisitos, com a finalidade de descobrir os relacionamentos de dependência entre os requisitos do software, e da lista de requisitos e interesses identificados. Além disso, o resultado dessa atividade consiste em atualizar essa lista, incluindo novos interesses, caso necessário.

Para exemplificar uma situação para a qual essa atividade é importante, considere a lista de requisitos e interesses identificados do Quadro 4.6. É possível notar que o interesse "Desempenho" foi identificado apenas no requisito "RNF-03", que foi criado para especificar esse tipo de comportamento do software. Contudo, a própria descrição do requisito "RNF-03" deixa claro que a propriedade de tempo de resposta deve ser aplicada às demais funções do software. Ou seja, outros requisitos desse software dependem do comportamento de bom desempenho e são entrecortados por esse interesse.

O procedimento para execução dessa atividade equivale ao algoritmo apresentado na Listagem 4.2. A ideia é que para cada requisito "r1" do software em análise, recupere-se a lista "C1" de interesses identificados para ele. Posteriormente, verifica-se se há relacionamentos de dependência entre "r1" e os demais requisitos do software. Caso haja, para cada requisito "r2", do qual "r1" depende, recupera-se a lista de interesses identificados para "r2", denominada "C2". Uma vez feito isso, copia-se os interesses de "C2" que não existem em "C1" para a lista "C1".

```
R1 ← lista de requisitos do software em análise
Para cada requisito r1 da lista R1
    C1 ← lista de interesses identificados para r1
    R2 ← lista de requisitos dos quais r1 depende
    Para cada requisito r2 da lista R2
        C2 ← lista de interesses identificados para r2
        Se C1 ∩ C2 ≠ ∅, então
            C1 ← C1 U C2
        Fim se
    Fim para
Fim para
```

Listagem 4.2. Procedimento para identificação de interesses a partir da interdependência entre requisitos de software.

Após a execução dessa atividade sobre a lista de requisitos e interesses identificados do Quadro 4.6, bem como sobre o documento de requisitos do Quadro 4.4, o resultado será uma lista atualizada de requisitos e interesses identificados, conforme apresentado no Quadro 4.7.

Quadro 4.7. Lista atualizada de requisitos e interesses identificados.

Requisito RF-01	**Interesses**	**Principal**
Tem como propósito realizar a **atualização** do andamento de uma **reclamação**. A **reclamação** deve estar cadastrada, com a situação ABERTA e com o funcionário **logado** no sistema.	Persistência	
	Segurança	
	Reclamação	X
	Desempenho	
	Usabilidade	
Requisito RNF-02	**Interesses**	**Principal**
O sistema deve ter uma interface de **fácil utilização**, visto que o sistema pode ser utilizado por qualquer pessoa que tenha acesso a Internet. O sistema deve ter um **HELP** on-line para ser consultado por qualquer pessoa que acesse o sistema.	Usabilidade	X
Requisito RNF-03	**Interesses**	**Principal**
O **tempo de resposta** do sistema não deve ultrapassar 5 segundos por acesso.	Desempenho	X

O requisito de código “RF-01” passou a contemplar os interesses “Desempenho” e “Usabilidade”. Essa inclusão adveio do fato de que: (i) o requisito “RF-01” depende dos requisitos “RNF-02” e “RNF-03”, conforme pode ser visto no Quadro 4.4; e (ii) os requisitos “RNF-02” e “RNF-03” estão relacionados, respectivamente, com os interesses “Desempenho” e “Usabilidade”.

4.5.3 Especificar interesses principais

Para cada requisito, o engenheiro de software deve informar manualmente qual é o interesse principal a que se refere esse requisito. O interesse principal é aquele que representa a finalidade principal para a qual o requisito foi proposto. Conforme comentado no Capítulo 3, a ideia de interesse principal é utilizada em alguns trabalhos sobre EROA (Clarke e Baniassad, 2005; Baniassad e Clarke, 2004) e sua utilidade está relacionada ao processo de classificação de interesses como base ou transversal, conforme é explicado na Seção 4.5.5. Isso porque, uma vez que um requisito é escrito para uma finalidade (interesse) principal, as demais finalidades referentes a ele podem ser consideradas como comportamento transversal.

No exemplo do Quadro 4.7, os requisitos “RNF-02” e “RNF-03” estão relacionados a apenas um interesse, que por sua vez são seus interesses principais. É importante salientar, nesta abordagem, que o fato de um requisito estar relacionado a apenas um interesse, não significa que esse é o seu interesse principal. Isso porque, nem todos os interesses de um requisito podem ter sido corretamente identificados nas atividades anteriores, por diversos tipos de ocorrências, que são mais bem explicadas na Seção 4.5.4 deste capítulo.

O requisito “RF-01”, por sua vez, refere-se a cinco interesses distintos do software, a saber, “Persistência”, “Segurança”, “Reclamação”, “Desempenho” e “Usabilidade”. Sua descrição, entretanto, deixa claro que ele foi criado com o intuito de especificar a

funcionalidade relacionada à atualização de uma reclamação cadastrada no software. Assim, “Reclamação” deve ser escolhido como interesse principal desse requisito. O Quadro 4.7 foi atualizado para contemplar as decisões tomadas durante essa atividade.

A decisão de qual é o interesse principal de um requisito deve ser tomada pelo engenheiro de software e caso haja requisitos para os quais seja difícil decidir qual é o seu interesse principal, ele deve considerar a possibilidade de reescrever o requisito em questão, desmembrando em outros requisitos menores.

4.5.4 Verificar resultados da identificação de interesses

Esta atividade tem como finalidade verificar os resultados das últimas atividades realizadas na abordagem, a procura de problemas relacionados à identificação de interesses e sugerir soluções para esses problemas. Ela recebe como entradas a lista de requisitos e interesses identificados e o catálogo de interesse de software e pode gerar como saída uma lista de ocorrências. Essa lista de ocorrências consiste em um conjunto de mensagens de alerta ou erro, juntamente com os elementos (requisitos ou interesses) relacionados a essas ocorrências. Por exemplo, um tipo de ocorrência verificado nesta atividade é se todos os requisitos do software estão relacionados ao seu interesse principal. Caso um determinado requisito “r” não seja contemplado por qualquer interesse do catálogo de interesses em análise, uma ocorrência desse tipo será gerada e relacionada a esse requisito, conforme pode ser visto no Quadro 4.8.

Quadro 4.8. Exemplo de uma lista de ocorrências.

Ocorrências	Elemento relacionado
Tipo I (ERRO): O requisito “r” não está relacionado a um interesse principal do software.	Requisito “r”

Os seguintes tipos de ocorrências são verificados nessa atividade: (i) **Ocorrência do Tipo I (ERRO)** - verificar se todos os requisitos do software estão relacionados ao seu interesse principal; (ii) **Ocorrência do Tipo II (ERRO)** - verificar se há interesses interdependentes que não foram identificados nos requisitos do software; (iii) **Ocorrência do Tipo III (ALERTA)** – verificar se há interesses não funcionais identificados, mas que não entrecortam requisitos funcionais do software; e (iv) **Ocorrência do Tipo IV (ALERTA)** - verificar se há interesses relacionados por contribuição positiva que não foram identificados nos requisitos do software.

Cada ocorrência contemplada pela abordagem *ObasCId*, juntamente com a descrição dos elementos envolvidos, a justificativa para existência da mesma e as

recomendações propostas para solução da ocorrência em questão podem ser vistas no Quadro 4.9, no Quadro 4.10, no Quadro 4.11 e no Quadro 4.12.

Quadro 4.9. Descrição de uma ocorrência do tipo I.

Ocorrência do Tipo I (ERRO)	**Elementos relacionados**
O requisito “i” não está relacionado ao seu interesse principal.	O requisito “i”
Justificativa para a Ocorrência	
Cada requisito do software deve estar relacionado a um interesse principal, uma vez que cada requisito é escrito com uma finalidade (propósito).	
Recomendações	
1. Caso o requisito “i” esteja relacionado a pelo menos um interesse do catálogo, deve-se então decidir qual é o seu interesse principal. 2. Caso o requisito “i” não esteja relacionado a qualquer interesse do catálogo de interesses do software ou caso a lista de interesses identificados para esse requisito não contemple seu interesse principal, recomenda-se então: 2.1. Verificar a grafia das palavras-chave, tanto daquelas relacionadas com a descrição do requisito “i”, quanto daquelas relacionadas aos interesses do catálogo; 2.2. Verificar a possibilidade de adicionar um novo interesse ao catálogo de interesses do software; 2.3. Verificar a possibilidade de adicionar novas palavras-chave a um interesse pré-existente do catálogo de interesses do software; 2.4. Verificar a possibilidade de reescrever a descrição do requisito “i” para que o mesmo passe a ser contemplado por algum interesse pré-existente no catálogo de interesses do software; ou 2.5. Verificar a possibilidade de criar relacionamentos de dependência do requisito “i” para outros requisitos de software, de forma que ele passe a ser contemplado por algum interesse pré-existente no catálogo de interesses do software.	

Quadro 4.10. Descrição de uma ocorrência do tipo II.

Ocorrência do Tipo II (ERRO)	**Elementos relacionados**
Há um relacionamento “r” do tipo “dependência” que associa os interesses “A” (fonte) e “B” (alvo). O interesse “A” foi identificado no software, porém o interesse “B” nem qualquer um de seus subinteresses foi identificado.	Os interesses “A” e “B”
Justificativa para a Ocorrência	
O fato de um interesse “A” depender de outro interesse “B”, significa que para que “A” exista no software, “B” ou algum de seus subinteresses também deve existir (por exemplo, para haver o interesse “Cancelamento de Reserva”, deve haver também o interesse “Efetuar Reserva”), assim a identificação de “A” e a não identificação de “B” implica em um problema que deve ser observado pelo engenheiro de software.	
Recomendações	
1. Verificar a grafia das palavras-chave, tanto daquelas relacionadas com o interesse “B” (e seus subinteresses), quanto daquelas relacionadas aos requisitos do software; 2. Verificar a possibilidade de adicionar novas palavras-chave ao interesse “B” (ou aos seus subinteresses), para que o mesmo passe a ser identificado; ou 3. Verificar a possibilidade de reescrever a descrição de algum requisito do software para que o mesmo passe a ser contemplado pelo interesse “B” (ou pelos seus subinteresses).	

Os procedimentos para geração das ocorrências discutidas anteriormente são apresentados na Listagem 4.3, Listagem 4.4, Listagem 4.5 e Listagem 4.6.

Quadro 4.11. Descrição de uma ocorrência do tipo III.

Ocorrência do Tipo III (ALERTA)	**Elementos relacionados**
Há um relacionamento "r" do tipo "contribuição positiva" que associa os interesses "A" (fonte) e "B" (alvo). O interesse "B" foi identificado nos requisitos do software, porém "A" nem qualquer um de seus subinteresses foi identificado.	Os interesses "A" e "B"
Justificativa para a Ocorrência	
Analogamente ao caso da ocorrência do tipo II, o fato de "A" contribuir positivamente para "B" fornece indícios de que se "B" for identificado, "A" (ou algum de seus subinteresses) também deve ser. Contudo, essa não é uma situação de erro. Por exemplo, mais de um interesse pode contribuir positivamente para "B" e o engenheiro de software poderia escolher uma ou outra opção de contribuição. Mesmo existindo apenas um interesse que contribua positivamente para "B", não há obrigatoriedade de que esse interesse seja contemplado pelo software. Por exemplo, "Desempenho" e "Padronização" contribuem positivamente para "Usabilidade", porém apenas um deles pode ser contemplado no software. Mesmo assim, faz-se necessário gerar uma ocorrência de alerta, uma vez que isso pode indicar ao engenheiro de software interesses que ele não havia pensando anteriormente em considerar no software.	
Recomendações	
1. Verificar a grafia das palavras-chave, tanto daquelas relacionadas com o interesse "A" (e seus subinteresses), quanto daquelas relacionadas aos requisitos do software. 2. Verificar a possibilidade de adicionar novas palavras-chave ao interesse "A" (ou aos seus subinteresses), para que o mesmo passe a ser identificado; 3. Verificar a possibilidade de reescrever a descrição de algum requisito do software para que o mesmo passe a ser contemplado pelo interesse "A" (ou pelos seus subinteresses); ou 4. Ignorar a ocorrência, pois o interesse "A" realmente não deveria existir no software em análise.	

Quadro 4.12. Descrição de uma ocorrência do tipo IV.

Ocorrência do Tipo IV (ALERTA)	**Elementos relacionados**
Há um interesse não funcional "A" que foi identificado no software, porém o mesmo, nem qualquer um de seus subinteresses, entrecorta requisitos funcionais do software.	O interesse "A"
Justificativa para a Ocorrência	
É bem conhecido na comunidade científica que interesses não funcionais possuem uma maior tendência a serem considerados transversais do que os interesses funcionais (Sampaio *et al*., 2007). Assim, ao final do processo de identificação de interesses, caso haja interesses não funcionais identificados no software que não afetem requisitos funcionais do mesmo, o comportamento transversal desse interesse pode estar sendo omitido. Isso pode ocorrer por diversas razões, sendo uma delas, a especificação inadequada dos relacionamentos de dependência entre requisitos do software. Esta não é uma situação de erro, mas serve de alerta ao engenheiro de software para que o mesmo possa verificar se o interesse em questão realmente não possui comportamento transversal.	
Recomendações	
1. Verificar a grafia das palavras-chave, tanto daquelas relacionadas com o interesse "A" (e seus subinteresses), quanto daquelas relacionadas aos requisitos do software. 2. Verificar a possibilidade de adicionar novas palavras-chave ao interesse "A" (ou aos seus subinteresses), para que o mesmo passe a ser identificado em outros requisitos do software; 3. Verificar a possibilidade de reescrever a descrição de algum requisito do software para que o mesmo passe a ser contemplado pelo interesse "A" (ou pelos seus subinteresses); 4. Verificar se os relacionamentos de dependência entre os requisitos do software foram corretamente especificados; ou 5. Ignorar a ocorrência, pois o interesse "A" realmente não deveria apresentar comportamento transversal no software em análise.	

A Listagem 4.3 varre a lista de requisitos do software, verificando se cada um deles possui pelo menos um interesse relacionado. Caso não haja, então uma ocorrência do tipo I é gerada. Caso contrário, verifica-se se na lista de interesses relacionados a esse requisito, há a especificação do seu interesse principal. Caso não, então a ocorrência é gerada.

```
R ← conjunto de requisitos do software em análise
Para cada requisito r do conjunto R
    Cr ← conjunto de interesses identificados para r
    Se Cr = ∅, então gerar uma "Ocorrência do Tipo I", tendo o requisito
        r como elemento relacionado a essa ocorrência
    Senão
       C ← interesse principal do conjunto Cr
       Se C = null, então gerar uma "Ocorrência do Tipo I", tendo o
           requisito r como elemento relacionado a essa ocorrência
    Fim se
Fim para
```

Listagem 4.3. Procedimento para identificar ocorrências do tipo I.

A Listagem 4.4, por sua vez, recupera os relacionamentos de dependência dos interesses identificados no software em análise. Para cada relacionamento, verifica-se se o interesse alvo desses relacionamentos (ou algum de seus subinteresses) também foi identificado no software. Caso não, então uma ocorrência do tipo II é gerada. O funcionamento do procedimento da Listagem 4.5 é bem similar ao da Listagem 4.4, contudo, relacionamentos de contribuição positiva são verificados.

```
C ← conjunto de interesses identificados no documento de requisitos em
    análise
Para cada interesse c do conjunto C
    Rc ← conjunto de relacionamentos de dependência para os quais c é o
         interesse fonte
    Para cada relacionamento r do conjunto Rc
        a ← interesse alvo do relacionamento r
        Se a (ou algum de seus subinteresses) não está em C, então gerar
            uma "Ocorrência do Tipo II", tendo os interesses c e a como
            elementos relacionados a essa ocorrência
        Fim se
    Fim para
Fim para
```

Listagem 4.4. Procedimento para identificar ocorrências do tipo II.

```
C ← conjunto de interesses identificados no documento de requisitos em
    análise
Para cada interesse c do conjunto C
    Rc ← conjunto de relacionamentos de contribuição positiva para os
         quais c é o interesse alvo
    Para cada relacionamento r do conjunto Rc
        a ← interesse fonte do relacionamento r
        Se a (ou algum de seus subinteresses) não está em C, então
            gerar uma "Ocorrência do Tipo III", tendo os interesses
            c e a como elementos relacionados a essa ocorrência
        Fim se
    Fim para
Fim para
```

Listagem 4.5. Procedimento para identificar ocorrências do tipo III.

Por fim, a Listagem 4.6 recupera a lista de interesses identificados no software e para cada interesse “c” do tipo não funcional, recupera-se a listagem “Rc” de requisitos “afetados” por ele. Neste caso, o termo “afetados” refere-se a requisitos para os quais o interesse “c” não é o interesse principal. Para cada requisito de “Rc”, verifica-se se há algum que seja do tipo funcional. Caso não haja requisitos funcionais afetados, então uma ocorrência do tipo IV é gerada.

```
C ← conjunto de interesses identificados no documento de requisitos em
    análise
Para cada interesse c do conjunto C
    Se c é do tipo 'Não funcional'
        Rc ← conjunto de requisitos afetados por c
        afetaReqFuncional ← falso
        Para cada requisito r do conjunto Rc
            Se r é do tipo 'Funcional', então
                afetaReqFuncional ← verdadeiro
                sair do laço
            Fim se
        Fim para
        Se afetaReqFuncional = false, então
            gerar uma "Ocorrência do Tipo IV", tendo o interesse c
            como elemento relacionado a essa ocorrência
        Fim se
    Fim se
Fim para
```

Listagem 4.6. Procedimento para identificar ocorrências do tipo IV.

Executando essa atividade sobre a lista de requisitos e interesses identificados do Quadro 4.7, tendo como entrada os catálogos de interesses da Figura 4.4 e Figura 4.5, a lista de ocorrências do Quadro 4.13 é gerada. Como pode ser visto, uma ocorrência do tipo III (alerta) foi gerada, pois o interesse “Concorrência” contribui positivamente para “Desempenho”, porém o mesmo não foi identificado nos requisitos do software.

Quadro 4.13. Exemplo de uma lista de ocorrências relacionadas à identificação de interesses.

Ocorrências	Elemento relacionado
Tipo III (ALERTA): há um relacionamento “r” do tipo “contribuição positiva” que associa os interesses “A” (fonte) e “B” (alvo). O interesse “B” foi identificado nos requisitos do software, porém “A” nem qualquer um de seus subinteresses foi identificado.	Interesses “Concorrência” e “Desempenho”

Por meio da lista de ocorrências e das descrições dos tipos de ocorrências apresentados anteriormente, o engenheiro de software possui subsídios mais apropriados para entender as possíveis causas da não identificação de determinado interesse, bem como para tomar alguma providência a respeito disso.

Para continuar com a explicação da abordagem *ObasCId*, considera-se que, no caso da ocorrência do Quadro 4.13, o engenheiro de software optou por ignorar essa mensagem de alerta e continuar sem considerar o interesse “Concorrência” para desenvolvimento do

software. É importante ressaltar que o engenheiro de software pode postergar sua tomada de decisão para a fase de "Detecção de Conflitos entre Interesse", quando ele terá mais informações a respeito da influência mútua entre os interesses identificados no software.

4.5.5 Classificar interesses

Esta atividade usa a listagem de requisitos e interesses identificados gerada nas atividades anteriores para construir uma matriz de entrelaçamentos entre interesses, que representa os entrecortes existentes entre os diversos interesses de um software. Trata-se de uma matriz similar àquela apresentada na abordagem *EROA/Arcade* (Rashid *et al*., 2003; Rashid *et al*., 2002), porém, neste caso, a matriz de entrelaçamentos é do tipo "Interesse vs. Interesse*"* e não "Interesse Não Funcionais vs. Pontos de Vista". A vantagem é que o entrelaçamento de interesses funcionais sobre outros interesses do software pode também ser analisado. O conteúdo de cada célula "[C1, C2]" de uma matriz de entrelaçamentos corresponde a um valor booleano (verdadeiro ou falso), que informa se o interesse "C2" entrecorta ou não o interesse "C1".

O procedimento para construção de uma matriz de relacionamento pode ser visualizado na Listagem 4.7.

```
C1 ← lista de interesses identificados no documento de requisitos em
     análise
#C1 ← tamanho da lista de interesses identificados no documento de
      requisitos em análise
Seja ME a matriz de entrelaçamentos inicializada com valores 'Falso'
Para i de 1 até #C1
    ip ← C1[i]
    R ← lista de requisitos para os quais ip é o interesse principal
    Para j de 1 até #C1
        Se i ≠ j, então
            is ← C1[i]
            Para cada requisito r da lista R
                C2 ← lista de interesses relacionadas ao requisito r
                Se is está na lista C2, então
                    ME[i, j] ← 'Verdadeiro'
                Fim se
            Fim para
        Fim se
    Fim para
Fim para
```

Listagem 4.7. Procedimento para construção de uma matriz de entrelaçamentos entre interesses.

A ideia descrita nessa listagem é que, para cada interesse identificado no software, seja obtida a lista de requisitos para os quais ele foi tido como interesse principal. A partir daí, para cada requisito dessa lista, os demais interesses desse requisito são obtidos e as células correspondentes da matriz de entrelaçamentos são atualizadas. Por exemplo, a

partir da lista de requisitos e interesses identificados do Quadro 4.7, pode-se gerar a matriz de entrelaçamentos do Quadro 4.14; para simplificar a visualização dessa matriz, os valores iguais a “falso” foram substituídos por espaços vazios e os iguais a “verdadeiro”, por símbolos “X”.

É possível notar que o requisito “RF-01” do Quadro 4.7, cujo interesse principal é “Reclamação”, está relacionado com outros interesses, tais como “Persistência”, “Segurança”, “Usabilidade” e “Desempenho”. Por isso, as células “[3, 1]”, “[3, 2]”, “[3, 4]” e “[3, 5]” foram assinaladas na matriz de entrelaçamentos do Quadro 4.14.

Quadro 4.14. Exemplo de uma matriz de entrelaçamentos entre interesses.

↓ IP / IT →	**1**	**2**	**3**	**4**	**5**
1: Persistência					
2: Segurança					
3: Reclamação	X	X		X	X
4: Usabilidade					
5: Desempenho					
Legenda: IP: Interesses Principais; IT: Interesses Transversais					

Ao manter o foco nas colunas de uma matriz de entrelaçamentos, o engenheiro de software terá uma visão de quais interesses entrecortam o comportamento de outros interesses. Quanto mais interesses forem entrecortados por um determinado interesse “A”, maior é a probabilidade de “A” ser um interesse transversal. Para saber em quais requisitos esses interesses se entrecortam, pode-se consultar a lista de requisitos e interesses identificados.

Nesta abordagem, considera-se que cada interesse “A” deveria conter requisitos para os quais ele fosse o interesse principal. Em outras palavras, a coluna correspondente ao interesse “A” deveria conter apenas células vazias (com valor “falso”). Sendo assim, todas as colunas com pelo menos um valor “verdadeiro” (X) referem-se a candidatos a interesses transversais do software. No caso do Quadro 4.14, todos os interesses, com exceção de “3: Reclamação” (coluna destacada), são considerados candidatos a interesses transversais.

A partir das linhas da matriz de entrelaçamentos, o engenheiro de software contemplará os interesses mais afetados por outros interesses. Por exemplo, é possível notar no Quadro 4.14 que o interesse “3: Reclamação” é um dos mais afetados por outros interesses; já os interesses “1: Persistência”, “2: Segurança”, “4: Desempenho” e “5: Usabilidade” não se encontram entrelaçados com outros interesses, o que não quer dizer que eles não afetem de alguma forma esses interesses. Por exemplo, “Segurança” contribui negativamente para o “Desempenho” do software. Esse tipo de situação pode ser verificado na atividade “Detecção de Conflitos entre Interesses” (Seção 4.6).

4.6 Detecção de Conflitos entre Interesses

Essa fase tem como objetivo prover informações ao engenheiro de software a respeito dos conflitos e influências mútuas existentes entre os interesses identificados. Um conflito entre interesses acontece quando há influência negativa de um interesse sobre outro e os dois foram identificados no software. Neste contexto, o engenheiro de software deve tomar decisões tais como: (i) assumir o impacto de um interesse sobre outro e continuar com o desenvolvimento do mesmo; (ii) descartar o interesse que impacta negativamente outro interesse do ciclo de desenvolvimento do software; ou (iii) procurar substituir o interesse que impacta negativamente outro interesse.

O interesse “Concorrência”, por exemplo, contribuiu positivamente para o “Desempenho”, mas negativamente para o “Custo” do projeto do software. Supondo que os três interesses tenham sido identificados em um software hipotético, o engenheiro de software precisará tomar uma decisão a respeito desse conflito. Nessas ocasiões, a prioridade dos interesses do software deve ser consultada. Em um sistema crítico, por exemplo, “Desempenho” pode ter uma prioridade maior do que o “Custo”, podendo-se então assumir os efeitos negativos da “Concorrência” sobre esse interesse. Em um sistema de informação para bibliotecas, entretanto, talvez o “Custo” seja o mais importante, portanto, uma opção seria descartar o interesse “Concorrência” do ciclo de desenvolvimento do software. Caso os interesses “Desempenho” e “Custo” tenham prioridades similares, novas reuniões com *stakeholders* do projeto e/ou com a equipe de desenvolvimento devem ser realizadas com o intuito de resolver o conflito. Para facilitar essa tomada de decisão, os artefatos gerados nesta fase podem ser úteis.

Apenas uma atividade ocorre nesta fase (Figura 4.9) e para sua execução são necessários os seguintes artefatos: (i) o catálogo de interesses de software: que contém a prioridade dos interesses envolvidos em um conflito; e (ii) a matriz de entrelaçamentos entre interesses: que é útil para se conhecer quais são os interesses identificados no software e com quais outros interesses eles se relacionam. Como artefato gerado, é produzida uma matriz de contribuição entre interesses, que aponta as influências existentes entre os diferentes interesses do software.

Uma matriz de contribuição é também uma matriz do tipo “Interesse vs. Interesses”. Cada célula “[C1, C2]” dessa matriz apresenta uma lista de interesses que contribuem de alguma forma para o interesse “C2” e que também entrecortam o interesse “C1”.

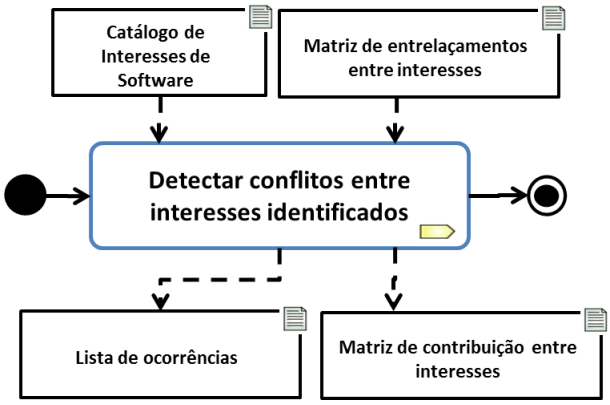


Figura 4.9. Atividade da fase "Detecção de Conflitos entre Interesses".

A matriz de contribuição é construída a partir da matriz de entrelaçamentos entre interesses, de acordo com o procedimento descrito na Listagem 4.8.

```
C ← lista de interesses identificados no documento de requisitos em
    análise
#C ← tamanho da lista de interesses identificados no documento de
     requisitos em análise
ME ← matriz de entrelaçamentos entre os interesses identificados
Seja MC a matriz de contribuição inicializada com strings vazias ("").
Para i de 1 até #C
    Para j de 1 até #C
        Se (i ≠ j e ME[i, j] ≠ 0), então
            intPrincipal ← C[i]
            intAlvo ← C[j]
            R ← lista de relacionamentos de contribuição em que intAlvo é
                o alvo
            Para cada relacionamento r da lista R
                intFonte ← interesse fonte do relacionamento r
                Se ME[intPrincipal, intFonte] ≠ 0, então
                    tipo ← tipo do relacionamento r
                    Se tipo = 'POSITIVE', então
                        MC[i, j] ← MC[i, j] + fonte + "(+)"
                    Senão
                        pf ← prioridade do interesse intFonte
                        pa ← prioridade do interesse intAlvo
                        Se (pf = NULL OU pa = NULL), então
                            MC[i, j] ← MC[i, j] + fonte + "(-)"
                        Senão se (pf < pa), então
                            MC[i, j] ← MC[i, j] + fonte + "(<)"
                        Senão se (pf > pa), então
                            MC[i, j] ← MC[i, j] + fonte + "(>)"
                        Senão
                            MC[i, j] ← MC[i, j] + fonte + "(=)"
                        Fim se
                     Fim se
                Fim se
            Fim para
        Fim se
    Fim para
Fim para
```

Listagem 4.8. Procedimento para construção de uma matriz de contribuição entre interesses.

O funcionamento do procedimento dessa listagem é o seguinte: para cada célula "[C1, C2]" da matriz de entrelaçamentos cujo conteúdo é "verdadeiro", procura-se por relacionamentos de contribuição, nos quais "C2" é o interesse alvo. Isto é, relacionamentos nos quais "C2" é afetado positivamente ou negativamente por outro interesse. Uma vez tendo encontrado algum relacionamento desse tipo, verifica-se se o interesse fonte "C3" desse relacionamento entrecorta o interesse "C1"; isso é feito verificando se o valor da célula "[C1, C3]" é "verdadeiro". Caso sim, deve-se então adicionar ao conteúdo da célula "[C1, C2]" da matriz de contribuição as seguintes informações: (i) o nome do interesse "C3"; (ii) o sinal "+", caso seja um relacionamento de contribuição positiva; e (iii) um dos sinais "<", ">", "=", "-", para relacionamentos de contribuição negativa.

Para decidir qual sinal utilizar no caso (iii), deve-se consultar a prioridade estabelecida para cada interesse do relacionamento. Caso a prioridade de "C3" seja menor do que a prioridade do interesse "C2", então o sinal "<" deve ser utilizado; caso seja maior ou igual, então os sinais ">" e "=" devem ser utilizados, respectivamente. Por fim, caso não haja informações sobre a prioridade de algum dos interesses envolvidos no relacionamento, então o sinal "-" deve ser utilizado.

A matriz de contribuição apresentada nesta tese é similar àquela apresentada para a abordagem *MDSoC* (Moreira *et al*., 2005), no sentido de que é uma matriz do tipo "Interesses vs. Interesse". Contudo, em cada célula "[C1, C2]" da matriz proposta na abordagem *MDSoC* está a lista de interesses impactados pela contribuição existente entre os interesses "C1" e "C2". Além disso, o tipo de contribuição (positiva ou negativa) é apresentado.

As principais diferenças da matriz de contribuição proposta para a abordagem *ObasCId* com relação à da abordagem *MDSoC* são que a matriz da abordagem *ObasCId*:

i) permite identificar rapidamente quais contribuições impactam diretamente no desenvolvimento de um determinado interesse. Para isso, basta manter o enfoque sobre a linha da matriz correspondente ao interesse desejado. No caso da abordagem *MDSoC*, o pesquisador deve observar as listas de interesses de cada célula da matriz, a procura daquelas em que o interesse desejado aparece; e

ii) permite conhecer rapidamente não apenas o tipo de contribuição (positiva ou negativa), mas também a relação entre as prioridades de cada interesse envolvido nesta contribuição.

A partir da matriz de entrelaçamentos do Quadro 4.14 e dos catálogos de interesses da Figura 4.4 e Figura 4.5, pode-se construir a matriz de contribuição do Quadro 4.15. Nela, nota-se que o interesse "2: Segurança" afeta negativamente o interesse "5: Desempenho" e ambos os interesses estão entrelaçados com o interesse "3: Reclamação". Além disso, nota-

se que “2: Segurança” possui prioridade menor do que “5: Desempenho” (apesar de não ter sido apresentada no catálogo da Figura 4.4, as prioridades desses interesses foram extraídas do documento de requisitos do software *Health Watcher* (2015)).

Quadro 4.15. Exemplo de uma matriz de contribuição entre interesses.

↓ IP / IT →	1	2	3	4	5
1: Persistência					
2: Segurança					
3: Reclamação					2 (<)
4: Usabilidade					
5: Desempenho					
Legenda: IP: Interesses Principais; IT: Interesses Transversais					

Em resumo, tendo em mãos a matriz de entrelaçamentos e a matriz de contribuição, o engenheiro de software pode fazer a seguinte leitura da situação atual do software, com relação à saída do processo de identificação e classificação de interesses: *três interesses contemplam, simultaneamente, pelo menos um dos requisitos do software Health Watcher, a saber, “Reclamação”, “Desempenho” e “Segurança”. Sabe-se ainda que para implementar esse(s) requisito(s), relacionado(s) primariamente ao interesse “Reclamação”, os requisitos de “Segurança” e “Desempenho” devem ser considerados. Contudo, a utilização de mecanismos de segurança pode afetar negativamente o desempenho do sistema, que nesse sistema, possui prioridade mais alta do que a do interesse “Segurança”*. Essa informação pode ser útil para que o engenheiro de software tome uma decisão a respeito do que fazer com essa contribuição negativa no contexto do desenvolvimento do software.

Outro ponto importante a ser ressaltado é sobre a contribuição positiva entre “Concorrência” e “Desempenho”, explicitamente declarada no catálogo de interesses da Figura 4.4. Essa contribuição não aparece na matriz do Quadro 4.15, pois “Concorrência” não foi identificada nos requisitos do software. Contudo, conforme apresentado no Quadro 4.13, foi gerada uma ocorrência na atividade anterior, que indica ao engenheiro de software essa contribuição. Naquele momento, o engenheiro de software decidiu por ignorar essa contribuição, porém sua decisão pode ser reavaliada a partir das novas informações obtidas nesta atividade de detecção e análise de conflitos. Isto é, sabendo que a “Segurança” contribui negativamente para “Desempenho” e “Desempenho” tem prioridade maior do que “Segurança”, uma alternativa poderia ser considerar o interesse “Concorrência” no desenvolvimento do software para minimizar o impacto negativo dos mecanismos de segurança sobre o desempenho do software.

4.7 Considerações Finais

Este capítulo apresentou uma descrição detalhada da abordagem *ObasCId*, incluindo informações sobre a construção de catálogos de interesses de software, bem como sobre o uso desses catálogos para a identificação e classificação de interesses. Além disso, exemplos práticos foram utilizados para facilitar a compreensão das etapas dessa abordagem.

O principal diferencial da *ObasCId* com relação às demais abordagens é a sua preocupação em auxiliar os engenheiros de software durante as etapas de identificação e classificação de interesses. Esse tipo de ajuda ocorre por meio:

i) dos artefatos gerados pela abordagem, tais como a lista de requisitos e interesses relacionados, que fornece ao engenheiro de software uma visão geral dos requisitos do software, dos interesses relacionados com os mesmos, bem como do interesse principal para o qual o requisito foi escrito. Além dessa lista, há também as matrizes de entrelaçamentos e de contribuição, bem como a lista de ocorrências, que fornecem informações importantes para que o engenheiro de software possa identificar problemas e/ou situações em que ele deve ficar atento durante o processo de identificação e classificação de interesses; e

ii) de um conjunto de passos e procedimentos bem definidos para utilização das informações existentes no catálogo de interesses durante o processo de identificação e classificação de interesses a partir de requisitos de software.

Com base nos resultados de estudos experimentais realizados sobre a abordagem *ObasCId* (Parreira Júnior e Penteado, 2015b e Parreira Júnior e Penteado, 2015c), notou-se um ganho em efetividade, principalmente com relação à cobertura, sem ter prejudicado a precisão e a eficiência (em termos de tempo de execução) dessa abordagem. Contudo, a execução manual da abordagem *ObasCId* pode ser onerosa para software de médio e grande porte, bem como propensa a erros. Nesse sentido, o próximo capítulo apresenta a ferramenta *ObasCId-Tool*, criada com o intuito de automatizar algumas das principais etapas da abordagem *ObasCId*.

Capítulo 5

OBASCID-TOOL: UMA FERRAMENTA DE APOIO À EROA BASEADA NA ABORDAGEM *OBASCID*

5.1 Considerações Iniciais

Baniassad e Clarke (2004) afirmam que a intuição ou mesmo o conhecimento de domínio de um engenheiro de software não são suficientes o bastante para se identificar, em um período razoável de tempo, potenciais ITs em softwares de médio e grande porte. Sampaio *et al*. (2007) corroboram com essa colocação, reforçando a importância da existência de ferramentas computacionais para aprimoramento da eficiência das abordagens para EROA.

Uma ferramenta computacional, denominada *ObasCId-Tool*, para automatização de algumas das atividades da abordagem *ObasCId*, foi desenvolvida neste doutorado para minimizar parte dos problemas que o uso de uma abordagem manual apresenta. Na Seção 5.2 são descritos alguns dos requisitos necessários para desenvolvimento desta ferramenta. Nas Seções 5.3 à 5.9, por sua vez, são apresentados detalhes da ferramenta computacional *ObasCId-Tool*, com enfoque em sua arquitetura e suas principais funções. Além disso, ao longo do texto, a funcionalidade da ferramenta será exercitada sobre o trecho do documento de requisitos apresentado na Seção 5.2. Na Seção 5.10 apresentam-se algumas estratégias para extensão da ferramenta *ObasCId-Tool*, visando a facilitar o seu reuso em outros contextos e na Seção 5.11 estão as considerações finais.

5.2 Trecho do documento de requisitos da Ferramenta *ObasCId-Tool*

No Quadro 5.1 é apresentado um trecho do documento de requisitos desenvolvido para a ferramenta *ObasCId-Tool* (a lista completa de requisitos encontra-se no Apêndice A). Esses requisitos foram elicitados, principalmente, a partir dos conceitos e relacionamentos da ontologia *O4C*, bem como das atividades e artefatos gerados/consumidos pela abordagem *ObasCId*. Tais requisitos são descritos de acordo com o modelo proposto para a abordagem *ObasCId* (Seção 4.4).

Quadro 5.1. Trecho do documento de requisitos da ferramenta *ObasCId-Tool*.

Nome do Requisito (Tipo + ID): Descrição do Requisito (RF – Requisito Funcional; RNF – Requisito Não Funcional)	**Requisitos Relacionados**
RF-01. O software deve permitir o cadastramento, alteração e exclusão de catálogos de interesses de software. Cada catálogo deve conter nome, descrição (opcional) e tipo de licença, que pode ser pública ou privada.	RNF-01, RNF-03
RF-02. O software deve permitir o cadastramento de pesquisadores. Cada pesquisador deve conter, obrigatoriamente, nome, email e senha. Opcionalmente, pode-se cadastrar: cidade, estado, país e nome da instituição a qual o pesquisador está vinculado. **...**	RNF-01, RNF-03 **...**
RNF-01. A interface de todas as funções do software deve ser responsiva, de forma que os elementos da mesma se adaptem a dispositivos com telas menores (tais como, *smartphones* e *tablets*) com no mínimo 5 polegadas. **...**	- **...**
RNF-03. O software deve ser de fácil utilização, permitindo que seus usuários, tendo passado por um treinamento de 40 (quarenta) minutos, consigam executar corretamente a maioria de suas funções.	-

5.3 Visão Geral da Ferramenta *ObasCId-Tool*

Na Figura 5.1 é apresentada a arquitetura da *ObasCId-Tool*, destacando seus principais componentes, bem como as dependências existentes entre eles. Nessa figura, as caixas retangulares especificam os módulos da ferramenta e as setas pontilhadas indicam as dependências entre esses módulos, de forma que o módulo ligado à origem da seta depende do módulo ligado ao destino da seta. Um cilindro representa um repositório de dados mantido/utilizado pela ferramenta.

ObasCId-Tool é composta por cinco módulos: (i) Módulo de Consulta aos Repositórios; (ii) Módulo de Gerenciamento de Pesquisadores; (iii) Módulo de Gerenciamento de Catálogos de Interesses de Software; (iv) Módulo de Gerenciamento de Documentos de Requisitos; e (v) Módulo de Identificação e Classificação de Interesses.

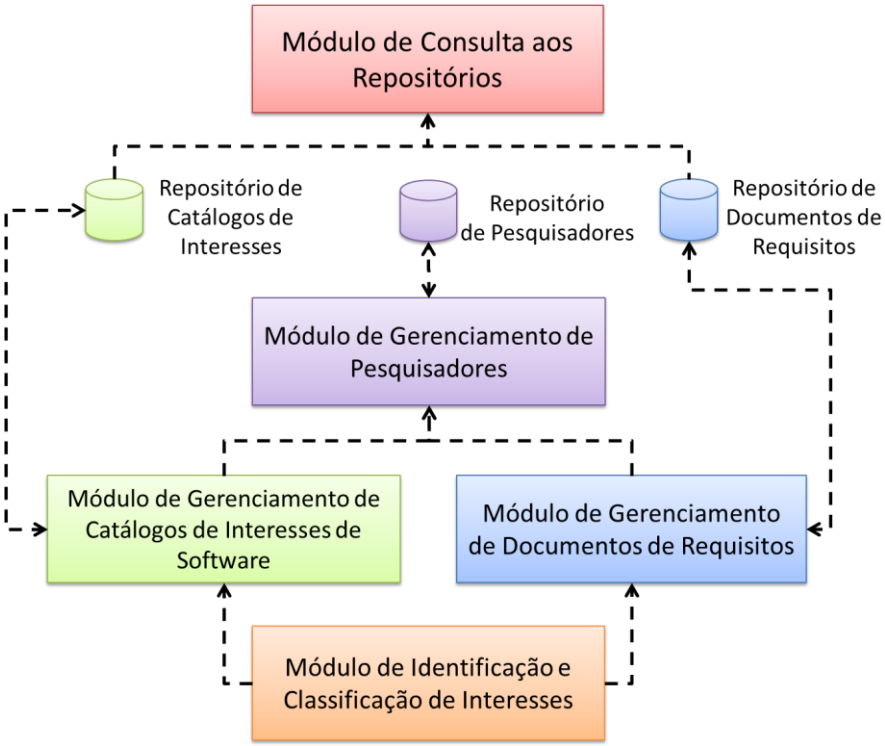


Figura 5.1. Arquitetura da ferramenta *ObasCId-Tool.*

Para que um pesquisador possa utilizar os módulos (iii), (iv) e (v) da ferramenta, ele precisa estar cadastrado e autenticado na mesma. Sendo assim, tais módulos dependem, direta ou indiretamente, do “Módulo de Gerenciamento de Pesquisadores”. Em seguida, o pesquisador pode criar e manter catálogos de interesses de software e documentos de requisitos, utilizando o “Módulo de Gerenciamento de Catálogos de Interesses de Software” e o “Módulo de Gerenciamento de Documentos de Requisitos”.

O “Módulo de Consulta aos Repositórios” pode ser utilizado por qualquer usuário interessado na ferramenta *ObasCId-Tool*, esteja ele cadastrado ou não. Esse módulo oferece opções de consulta e visualização dos catálogos de interesses de software e dos documentos de requisitos cadastrados na ferramenta, desde que tenham licença do tipo “Pública”. Mais detalhes sobre esse módulo são apresentados na Seção 5.5.

Para que o usuário possa realizar a identificação e classificação dos interesses de um software, deve existir pelo menos um catálogo de interesses e um documento de requisitos cadastrado para ele. Sendo assim, o “Módulo de Identificação e Classificação de Interesses” depende do “Módulo de Gerenciamento de Catálogos de Interesses de Software” e do “Módulo de Gerenciamento de Documentos de Requisitos”. Não há dependência explícita entre esses módulos, de forma que o pesquisador pode gerenciar livremente seus documentos de requisitos e catálogos de interesses de software.

5.4 Módulo de Gerenciamento de Pesquisadores

Este módulo é responsável pelo cadastramento, atualização e autenticação de pesquisadores da ferramenta *ObasCId-Tool*, alimentando e fazendo uso do "Repositório de Pesquisadores". Por meio dele, qualquer pesquisador pode criar um perfil na ferramenta e fazer uso das demais funcionalidades da mesma.

O cadastro de um pesquisador exige apenas nome, *email* e senha como informações obrigatórias (Figura 5.2). Outros dados opcionais são a instituição, a cidade, o estado e o país onde o usuário realiza seus trabalhos.

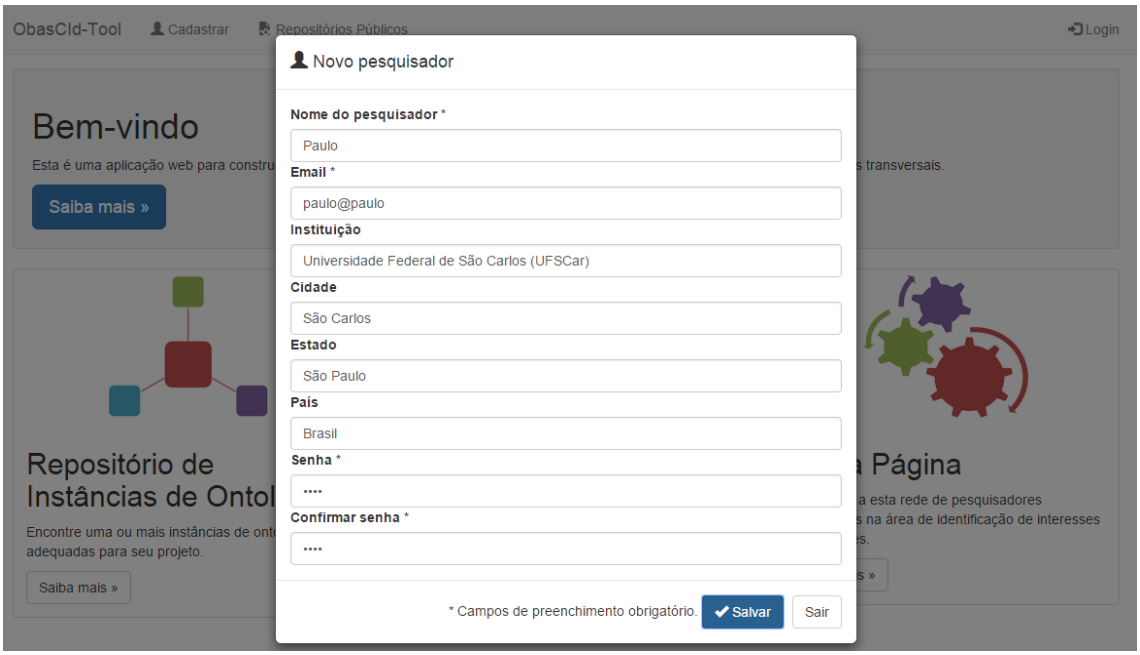


Figura 5.2. Cadastro de pesquisadores.

Uma vez cadastrado, serão gerados automaticamente para esse usuário, uma lista de *stopwords* e um conjunto de configurações padrão (*default*) para entrada de dados, que serão utilizadas nas demais telas da ferramenta *ObasCId-Tool*. Exemplos dessas configurações padrão são: (i) o tipo de licença de um catálogo de interesses de software ou de um documento de requisitos, que pode ser "Pública" ou "Privada"; (ii) o tipo de um interesse de software, que pode ser "Funcional" ou "Não funcional", entre outras. Essas configurações podem ser alteradas a qualquer momento pelo usuário, conforme será comentado na Seção 5.6.1.

Para ter acesso aos demais recursos oferecidos pela *ObasCId-Tool*, o pesquisador deve autenticar-se, informando seu *email* e sua senha, conforme ilustrado na tela da Figura 5.3.

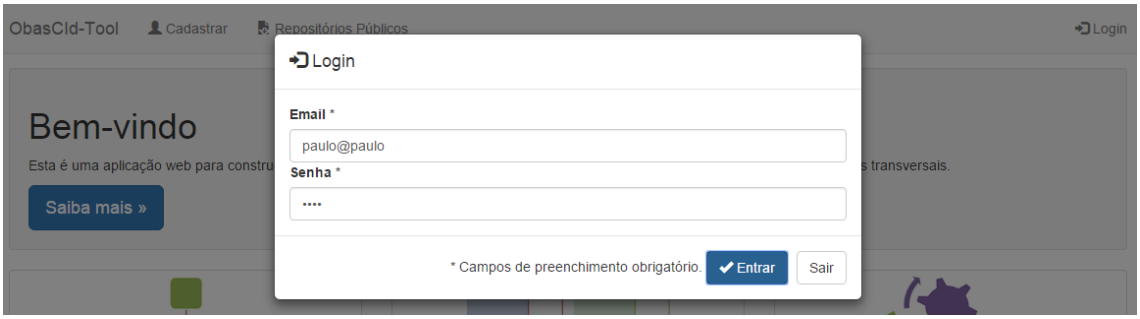


Figura 5.3. Autenticação de um pesquisador.

5.5 Módulo de Consulta aos Repositórios

Qualquer usuário, cadastrado ou não na ferramenta *ObasCId-Tool*, pode consultar e visualizar catálogos de interesses de software, bem como documentos de requisitos, desde que esses artefatos possuam licença pública.

Na Figura 5.4 é apresentada a lista de catálogos de interesses de software públicos disponíveis na ferramenta *ObasCId-Tool* em um dado momento. Essa lista apresenta o nome e a descrição do catálogo, o nome do pesquisador que o elaborou e um *link* para acessar o conteúdo desse catálogo, juntamente com o número de vezes em que ele já foi acessado por outros usuários. Para procurar por um catálogo de interesses de software, o usuário pode utilizar um trecho do nome ou da descrição do catálogo.

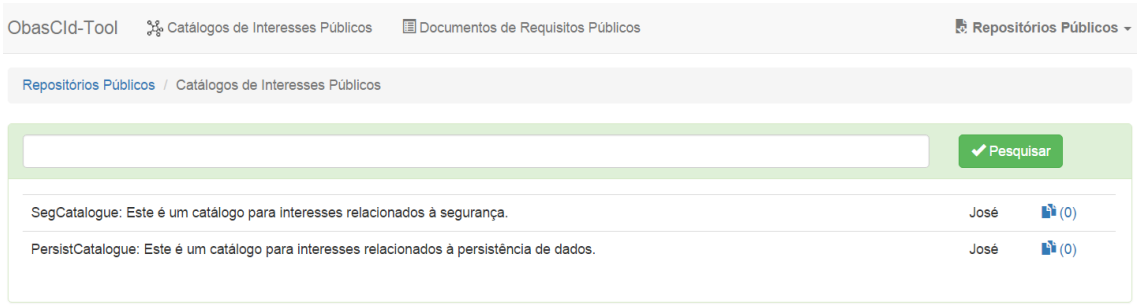


Figura 5.4. Repositório público de catálogos de interesses de software.

Pesquisadores cadastrados e autenticados na ferramenta também podem fazer uso do “Módulo de Consulta aos Repositórios”. Neste caso, além das funções já comentadas, eles terão acesso à opção de importação de catálogos de interesses de software para suas respectivas contas. Esse processo irá criar um clone do catálogo selecionado, alterando apenas o proprietário do mesmo.

A mesma funcionalidade comentada anteriormente também está disponível para documentos de requisitos, isto é, o pesquisador pode consultar e, caso esteja autenticado, importar documentos de requisitos públicos para sua conta pessoal.

5.6 Módulo de Gerenciamento de Catálogos de Interesses de Software

Este módulo permite que o pesquisador elabore e gerencie catálogos de interesses de software específicos. Baseado na ontologia *O4C*, ele representa um *wizard* para instanciação dos conceitos e relacionamentos definidos nesta ontologia (Capítulo 3). Por exemplo, na ontologia *O4C* existe o conceito "*Concern*", que representa um interesse de software. No "Módulo de Gerenciamento de Catálogos de Interesses de Software", há mecanismos apropriados, conforme será apresentado mais a frente neste capítulo, para que o pesquisador possa criar uma instância desse conceito para um tipo específico de interesse.

Os axiomas definidos na Seção 3.3.8 foram importantes para implementação do mecanismo de tratamento de exceções da ferramenta *ObasCId-Tool*, bem como dos algoritmos para identificação e classificação de interesse e detecção de conflitos. Por meio desses axiomas, situações consideradas incorretas para um catálogo de interesses de software puderam ser conhecidas, evitadas e notificadas ao pesquisador por meio de mensagens de erro apresentadas pela ferramenta.

5.6.1 Gerenciamento dos elementos básicos de uma instância

É importante salientar que cada catálogo de interesses de software está vinculado a um pesquisador cadastrado na ferramenta. Uma vez autenticado, o usuário terá acesso à listagem de catálogos cadastrados por ele (Figura 5.5).

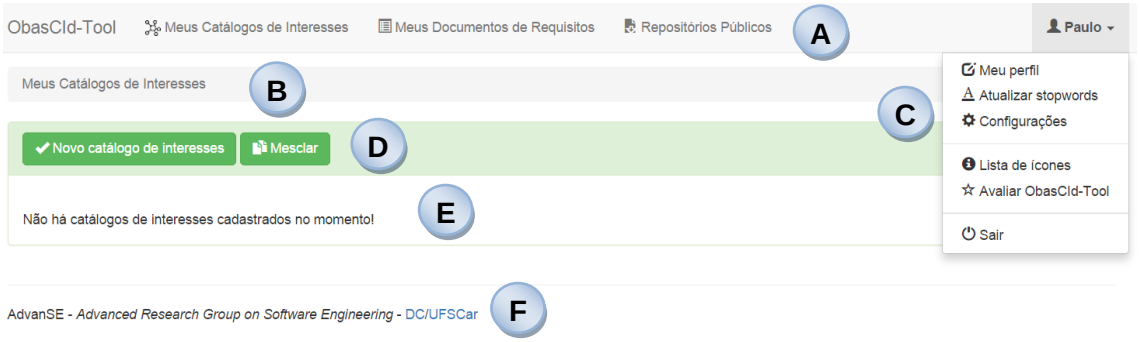


Figura 5.5. Tela para gerenciamento de catálogos de interesses de software.

Na Figura 5.5 (A) está apresentado o *menu* de acesso às principais funções da ferramenta *ObasCId-Tool*. Esse *menu* é contextual, isto é, dependendo da página na qual o pesquisador se encontra, seus itens podem ser diferentes. Na Figura 5.5 (B) é apresentado um recurso de *feedback* ao usuário sobre sua localização na ferramenta, denominado

breadcrumbs. Além de indicar em qual página (função) o usuário se encontra, por meio do *breadcrumbs,* é possível navegar pelo caminho que o pesquisador percorreu até chegar àquela página/função.

Na Figura 5.5 (C) é apresentado outro tipo de *menu* contextual, que agrega itens para outras funções da ferramenta. No exemplo da Figura 5.5 (C), por meio desse *menu*, o pesquisador pode: (i) atualizar seu perfil; (ii) atualizar a lista de *stopwords* cadastradas para ele; (iii) atualizar as configurações padrão da ferramenta; (iv) consultar a lista de ícones utilizados na ferramenta e seus significados; (v) avaliar a ferramenta *ObasCId-Tool* por meio de um formulário eletrônico apropriado; ou (vi) sair da ferramenta. Essas funções são apresentadas mais a frente neste capítulo.

Na Figura 5.5 (D, E) encontra-se o painel com a listagem dos catálogos de interesses de software e, como pode ser visto, não há catálogos cadastrados até o momento. Esse painel é dividido em duas partes principais: (i) a parte com os botões de ação correspondentes à página em que o pesquisador se encontra – Figura 5.5 (D); e (ii) a parte com a listagem dos registros referentes à página em que o pesquisador se encontra – Figura 5.5 (E). Essa estrutura se repete para as demais funções de gerenciamento de recursos da ferramenta, tais como documentos de requisitos, interesses, palavras-chave, entre outros.

Por fim, na Figura 5.5 (F) é apresentado o rodapé das páginas da ferramenta *ObasCId-Tool*. Por ser um elemento que é comum a todas as páginas da ferramenta, ele não será mais apresentado nas próximas figuras deste capítulo.

Conforme pode ser visto na Figura 5.6, para cadastrar um catálogo de interesses de software, o usuário deve informar obrigatoriamente um nome único para esse catálogo e o seu tipo de licença, que pode ser “Privada” ou “Pública”. Opcionalmente, o usuário poderá informar uma descrição para seu catálogo.

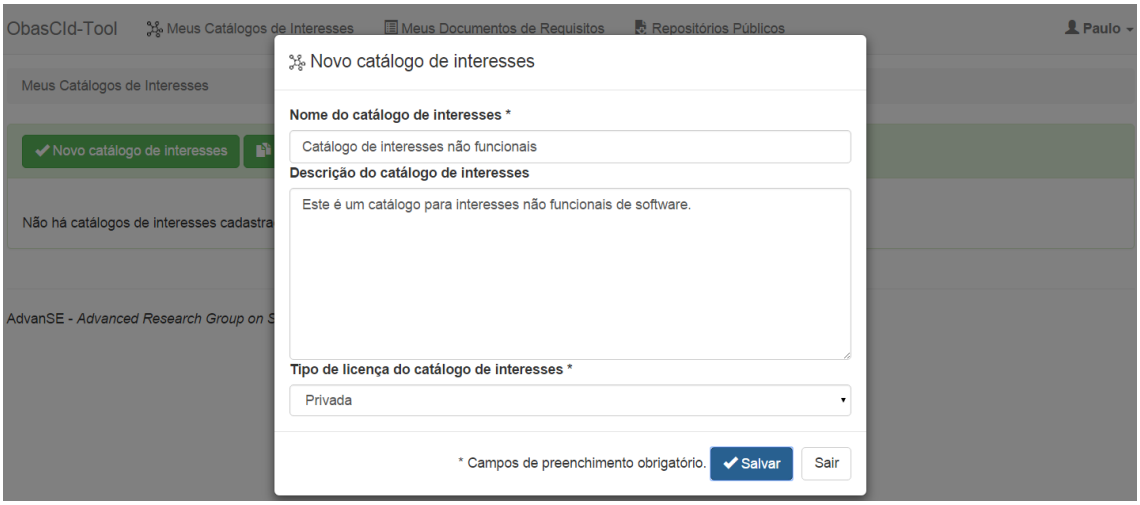


Figura 5.6. Cadastro de um catálogo de interesses de software.

O tipo de licença “Pública” permite que qualquer pessoa interessada busque e visualize o catálogo de interesses. Caso esse interessado seja também um pesquisador cadastrado na ferramenta, ele poderá importar o mesmo para seu repositório de catálogos pessoal. O tipo de licença “Privada” restringe o uso do catálogo de interesses ao contexto do pesquisador que o cadastrou. Assim, ele não poderá ser consultado ou importado por outros usuários da ferramenta. Isso é útil quando o pesquisador está preparando seu catálogo e não quer torná-lo disponível até que o mesmo esteja finalizado.

Na Figura 5.6 é possível notar que o campo correspondente ao tipo de licença do catálogo de interesses aparece com a opção “Privada” selecionada. Essa opção vem configurada por padrão na ferramenta *ObasCId-Tool*. Contudo, conforme discutido anteriormente, o pesquisador poderá modificar a qualquer momento essa e outras configurações da ferramenta, acessando o item de *menu* “Configurações”, apresentado na Figura 5.5 (C). As configurações padrão da ferramenta são apresentada na Figura 5.7.

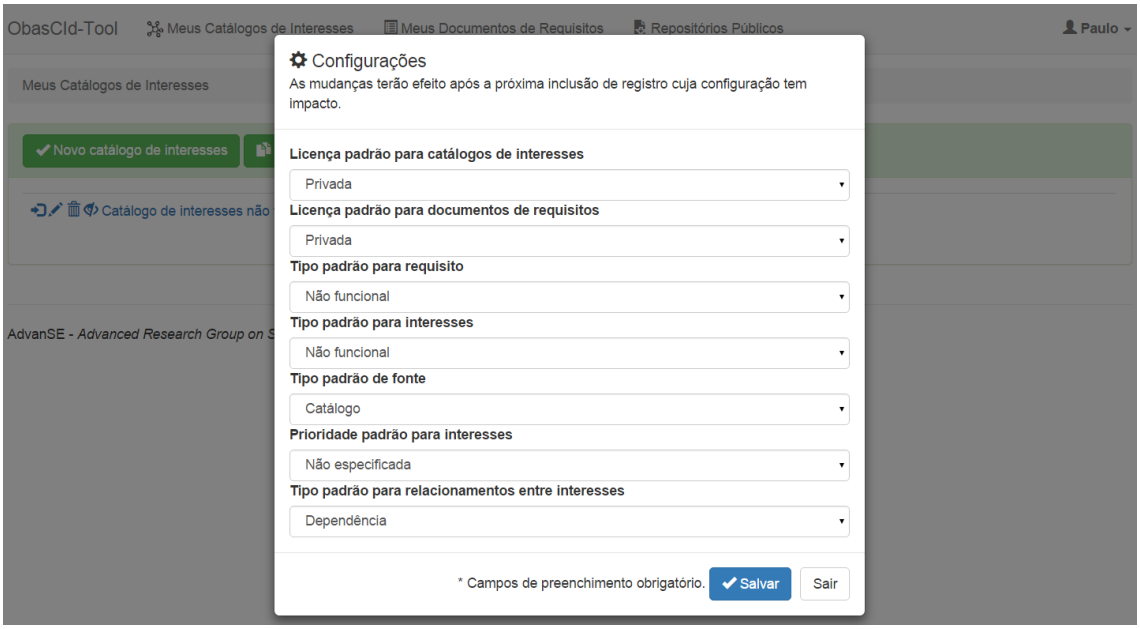


Figura 5.7. Configurações padrão (*default*) da ferramenta.

Uma vez cadastrado o catálogo de interesses, o mesmo passa a constar na lista de catálogos cadastrados pelo pesquisador, conforme pode ser visto na Figura 5.8. Para melhorar a legibilidade desta figura, apenas o painel com a lista de catálogos é apresentado.

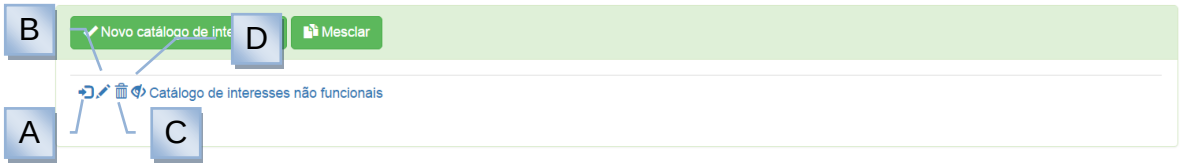


Figura 5.8. Listagem de catálogos de interesses de software.

A partir da Figura 5.8 é possível observar alguns elementos importantes, tais como o nome do catálogo de interesses de software e um conjunto de ícones. Os ícones da Figura 5.8 (B, C) aparecem em outras telas da ferramenta e são responsáveis por acionar as

funções de atualização e remoção de um determinado recurso. Além disso, para a maioria dos recursos mantidos pela ferramenta, o nome dos mesmos corresponde a um *link* que permite ao pesquisador visualizar mais informações sobre esse recurso. Por exemplo, nota-se que a descrição do catálogo de interesses de software não aparece na Figura 5.8. Para visualizá-lo, o pesquisador deve clicar sobre o nome desse catálogo (Figura 5.9).

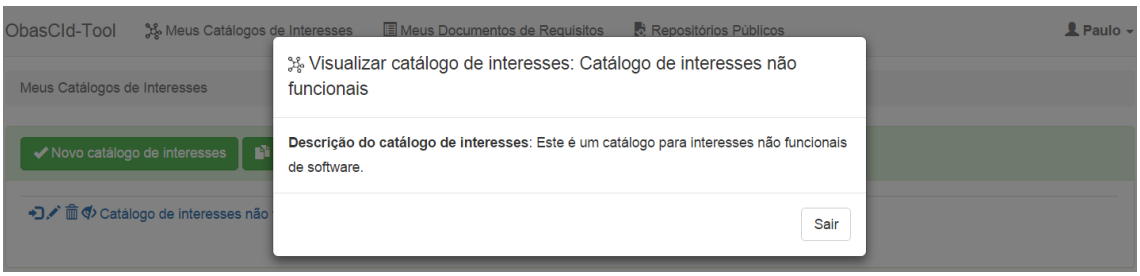


Figura 5.9. Visualização de mais informações sobre um catálogo de interesses de software.

Dois ícones específicos da tela de gerenciamento de catálogos de interesses de software são destacados na Figura 5.8 (A, D). O ícone da Figura 5.8 (D) representa o tipo de licença do catálogo cadastrado, que neste caso é “Privada”, e permite a troca do tipo de licença do catálogo de “Pública” para “Privada” e vice-versa. Trata-se de um atalho para que o pesquisador possa modificar rapidamente o tipo de licença de seu catálogo de interesses, sem precisar abrir a página de atualização do mesmo. O ícone da Figura 5.8 (A) permite que o pesquisador selecione um catálogo específico para gerenciar outros elementos do mesmo, tais como seus interesses e relacionamentos, bem como gerar relatórios do catálogo, com todas as informações correspondentes a ele.

Para manter a visualização das páginas da ferramenta *ObasCId-Tool* menos “sobrecarregada”, diversos ícones foram utilizados para representar conceitos da ontologia *O4C*, bem como da abordagem *ObasCId*. Para facilitar o entendimento do pesquisador quanto ao significado de cada ícone, a ferramenta *ObasCId-Tool* provê dois mecanismos. O primeiro são as dicas (*tooltips*) atribuídas aos ícones da ferramenta, que aparecem quando o usuário passa o cursor sobre um determinado ícone (Figura 5.10 - A). O segundo mecanismo consiste em uma lista de ícones e seus significados, disponível no *menu* suspenso do canto superior esquerdo das telas da ferramenta, conforme pode ser visto na Figura 5.5 (C). Na Figura 5.10 (B) é apresentada a tela com a lista de ícones utilizados na ferramenta e seus significados.

O último recurso a ser comentado nesta subseção diz respeito à possibilidade de o pesquisador combinar dois catálogos de interesses de software em um só. Isso é útil quando o pesquisador encontra, em catálogos distintos, interesses importantes e que ele gostaria de utilizar durante o processo identificação e classificação de interesses. Por exemplo, o pesquisador pode desejar utilizar os interesses cadastrados em um catálogo para o domínio de segurança, juntamente com os interesses de usabilidade cadastrados em

outro catálogo; ou então mesclar um catálogo para interesses não funcionais com outro para interesses funcionais de um determinado domínio.

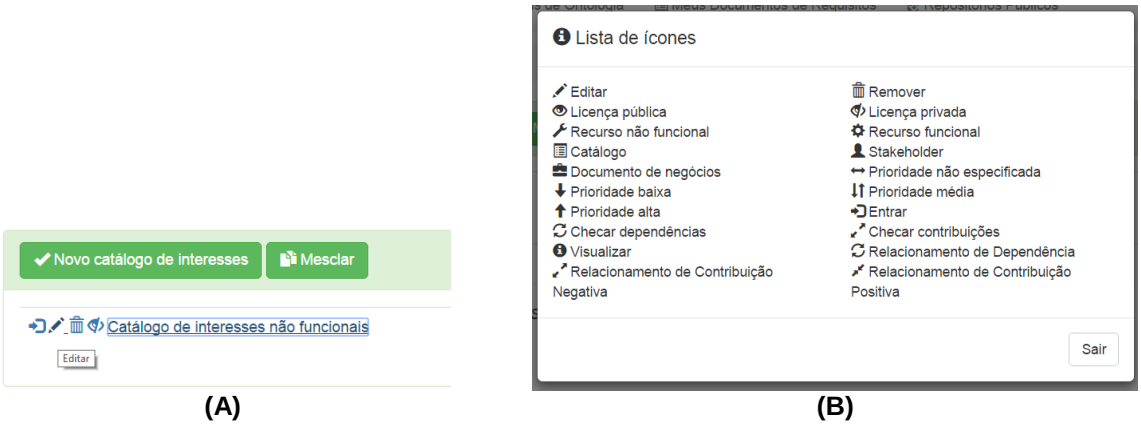


Figura 5.10. Recursos para entendimento do significado dos ícones utilizados na ferramenta.

Para executar essa função, o pesquisador precisa de pelo menos dois catálogos cadastrados em sua conta. Esses catálogos podem ter sido elaborados por ele ou importados de outro pesquisador. Além disso, ele deve informar um novo nome para o catálogo a ser gerado e o seu tipo de licença (Figura 5.11). Opcionalmente, ele também pode fornecer uma descrição para seu catálogo.

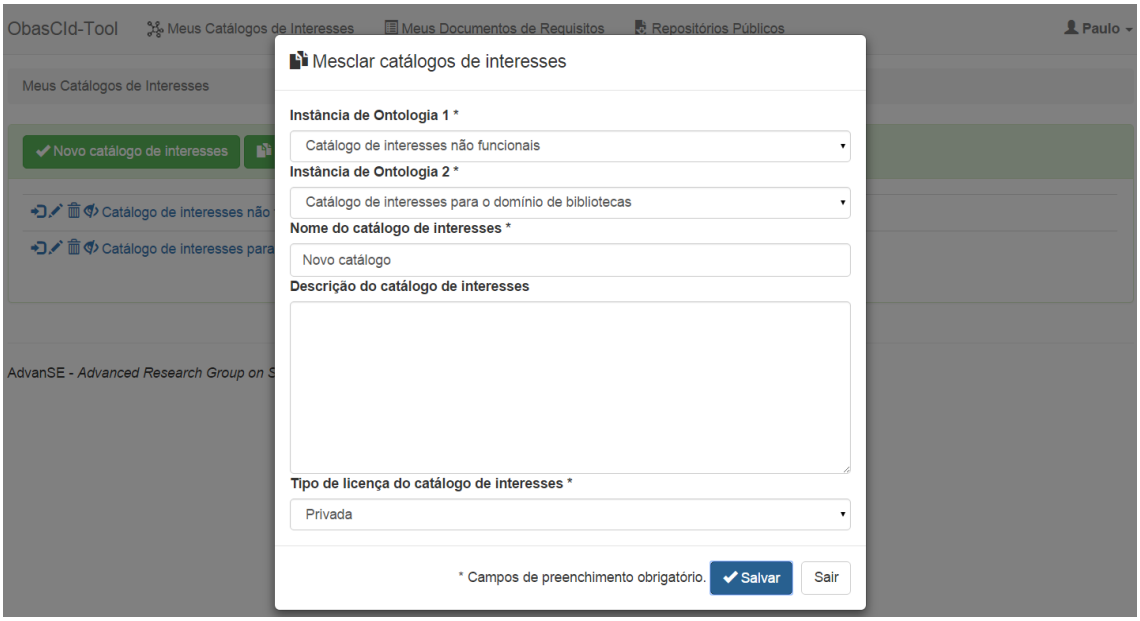


Figura 5.11. União de dois catálogos de interesses de software pré-cadastrados.

5.6.2 Gerenciamento dos interesses de um catálogo

Na Figura 5.12 é ilustrada a tela da ferramenta *ObasCId-Tool* para gerenciamento dos componentes de um catálogo de interesses de software.

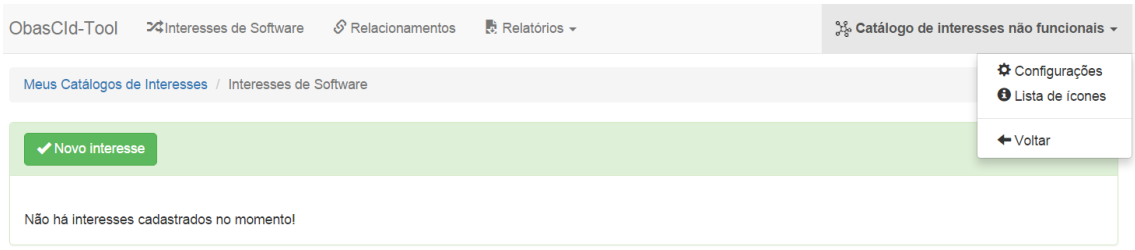


Figura 5.12. Tela para gerenciamento de interesses de software.

Alguns elementos dessa página, como seu *menu* principal e o *breadcrumbs* se alteraram para o contexto do gerenciamento dos componentes de um catálogo de interesses de software. É possível notar no canto superior direito da Figura 5.12, o nome do catálogo de interesses ativo (em atualização). Isso é importante, pois auxilia o pesquisador a identificar rapidamente qual catálogo está sendo atualizado por ele em um determinado momento. Além do nome do catálogo, há também itens de *menu* para acesso às configurações e à lista de ícones da ferramenta, bem como para voltar à listagem de catálogos de interesses de software.

Os principais itens de *menu* da página da Figura 5.12 são “Interesses de Software”, “Relacionamentos” e “Relatórios”, que correspondem, respectivamente à: (i) tela para gerenciamento de interesses relacionados com o catálogo ativo no momento; (ii) tela para gerenciamento de relacionamentos existentes entre os interesses cadastrados para o catálogo ativo no momento; e (iii) tela de relatórios do catálogo ativo no momento.

O gerenciamento dos interesses de um catálogo será detalhado nesta seção. Na Figura 5.13 é apresentada a tela para cadastramento de um interesse.

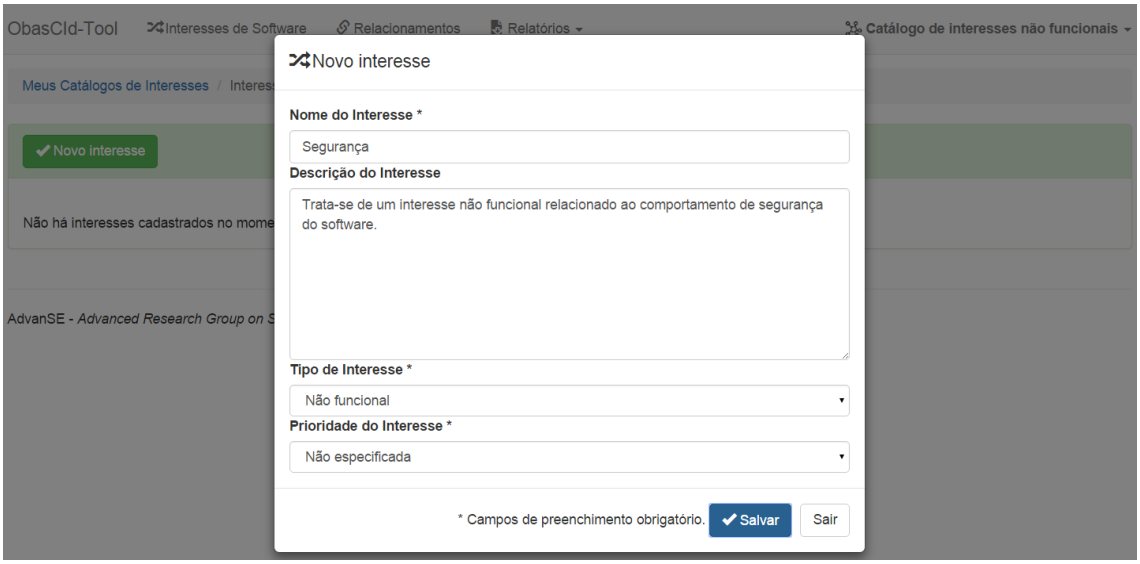


Figura 5.13. Cadastro de um interesse de software.

O pesquisador deve informar, obrigatoriamente, o nome do interesse, que deve ser único para o catálogo ativo no momento, o tipo desse interesse, que pode ser “Funcional” ou “Não funcional” e sua prioridade, que pode ser “Alta”, “Média”, “Baixa” ou “Não

especificada". Como informação opcional, o pesquisador pode informar uma descrição para o interesse.

É importante salientar que, apesar de estar disponível no momento do cadastramento do interesse, a especificação da prioridade do mesmo pode ser postergada para as fases posteriores do processo de identificação e classificação de interesses, uma vez que essa informação pode variar de projeto para projeto. Para isso, o pesquisador pode escolher a opção "Não especificada" como prioridade de um determinado interesse.

Uma vez cadastrado, esse interesse vai para a listagem de interesses do catálogo ativo no momento, conforme pode ser visto na Figura 5.14. Há novos ícones, específicos dessa tela, que são detalhados a seguir.

Figura 5.14. Listagem de interesses de software.

Os ícones da Figura 5.14 (A, B) descrevem, respectivamente, o tipo do interesse e sua prioridade. Isso serve para que o usuário reconheça rapidamente, dentre vários interesses, aqueles que são funcionais ou não funcionais, e que possuem prioridade "Alta", "Média", "Baixa" ou "Não especificada". Para atualizar essas informações, o usuário deve clicar sobre o ícone de atualização de recursos (similar àquele exibido na Figura 5.8 - B).

Na Figura 5.15 é exibida a lista de interesses cadastrados com base no catálogo da Figura 4.4 (Capítulo 4). Neste caso, todos os interesses são não funcionais e possuem prioridade "não especificada".

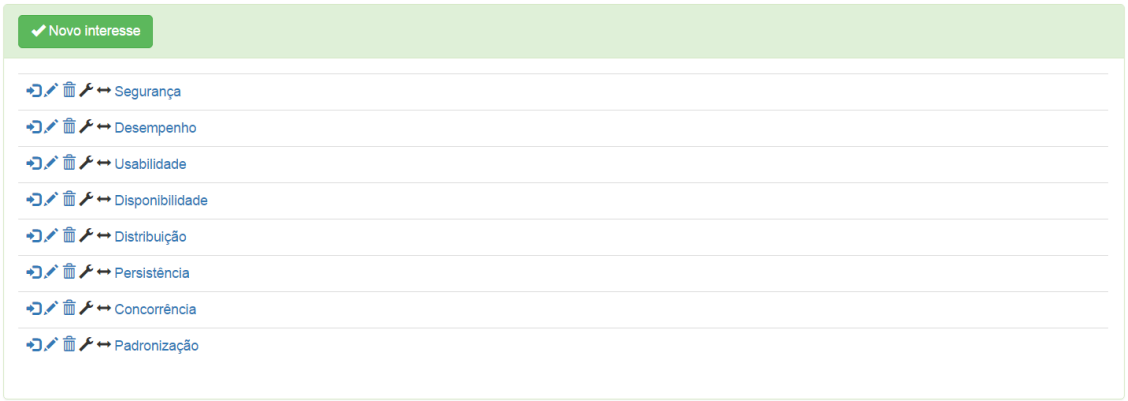


Figura 5.15. Listagem de interesses do "Catálogo de interesses não funcionais".

Pode-se notar na Figura 5.14 que há um ícone similar àquele apresentado na Figura 5.8 (A). Neste caso, ele permite que o pesquisador selecione um interesse específico para gerenciar suas fontes e palavras-chave. Uma vez que fontes e palavras-chave estão intimamente ligadas ao conceito de interesses, o gerenciamento desses recursos também é tratado nesta seção.

Gerenciamento de palavras-chave. A tela apresentada na Figura 5.16 é para o gerenciamento de palavras-chave de um interesse. De forma análoga ao que foi apresentado na tela para gerenciamento de componentes de um catálogo de interesses de software, nesta tela, o nome do interesse ativo (em atualização) é apresentado no canto superior direito da tela.

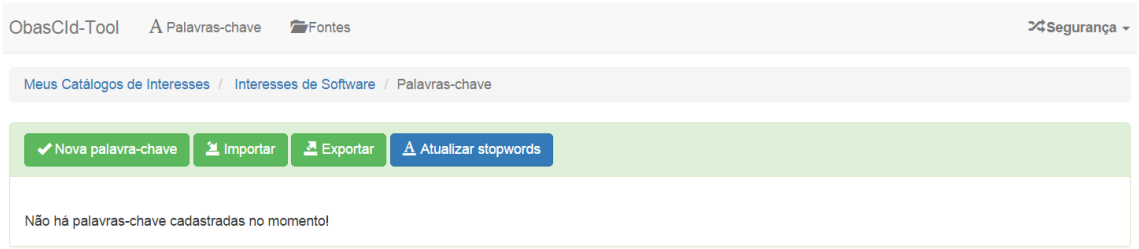


Figura 5.16. Tela para gerenciamento de palavras-chave.

Para cadastrar uma palavra-chave, o pesquisador deve informar, obrigatoriamente, a sua descrição (que deve ser única para o interesse transversal ativo no momento) (Figura 5.17). Palavras-chave compostas, tais como “tempo de resposta” e “controle de acesso” devem ser informadas entre aspas duplas.

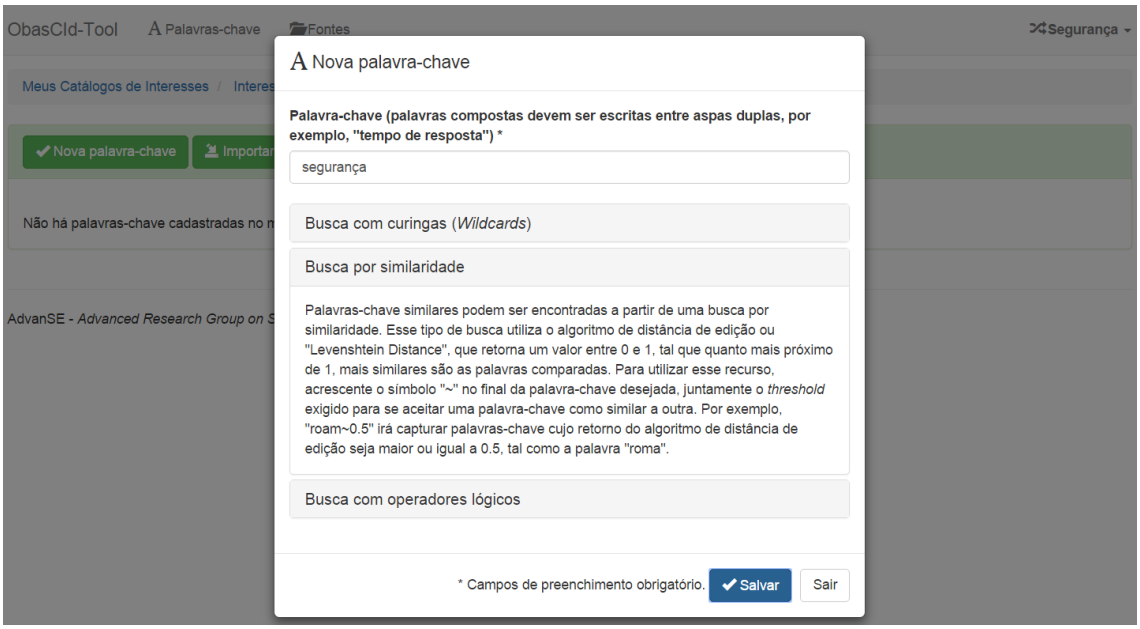


Figura 5.17. Cadastro de palavras-chave.

Um ponto importante sobre a função de cadastramento de palavras-chave é que a descrição da palavra-chave não pode estar contida na lista de *stopwords* cadastradas para o pesquisador autenticado na ferramenta. *Stopwords* são palavras que não possuem significado intrínseco e que, por isso, não são adequadas para identificação de conceitos específicos, tais como interesses de software (Manning *et al.*, 2008). Esse recurso da ferramenta *ObasCId-Tool* é útil, pois: (i) serve de orientação, principalmente para pesquisadores pouco experientes, com relação ao cadastramento de conjuntos de palavras-

chave mais adequados; e (ii) pode evitar a incidência de muitos falsos positivos durante a identificação de interesse do software.

Caso o pesquisador tenha certeza da necessidade de inclusão de uma palavra-chave que aparece em sua lista de *stopwords*, ele poderá atualizar a lista de *stopwords* relacionada ao seu cadastro. Para isso, o pesquisador deve clicar sobre o botão "Atualizar *stopwords*", disponível na tela exibida na Figura 5.16, ou sobre o item do *menu* suspenso apresentado na Figura 5.5 (C). A tela para edição da lista de *stopwords* é apresentada na Figura 5.18.

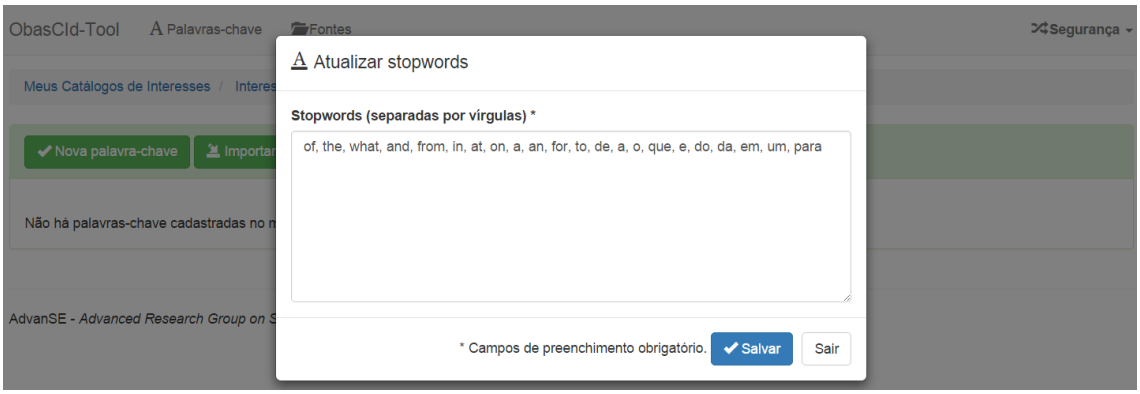


Figura 5.18. Atualização da lista de *stopwords*.

Uma lista prévia de *stopwords* para os idiomas inglês e português (do Brasil), obtida a partir de conhecidos repositórios públicos de *stopwords* (RanksNL, 2015), é automaticamente cadastrada para cada novo pesquisador. Na Seção 5.10 é descrita uma maneira de se estender essa lista de *stopwords* para outros idiomas, bem como adicionar ou remover palavras desse conjunto prévio.

O conceito de palavras-chave consiste em um importante recurso para identificação de interesses a partir dos requisitos de um software. Sendo assim, a ferramenta *ObasCId-Tool* oferece três mecanismos para confecção de palavras-chave que possam contribuir de forma mais efetiva para o processo de identificação de interesses. Esses mecanismos são: (i) busca com curingas (*wildcards*); (ii) busca por similaridade; e (iii) busca com operadores lógicos. Na Figura 5.17 é apresentada a tela que descreve esses mecanismos ao pesquisador, incluindo exemplos de uso dos mesmos.

Os curingas (*wildcards*) "?" e "*" podem ser utilizados na descrição de uma palavra-chave. O curinga "?" permite a detecção de palavras-chave que se casam com até uma substituição de caractere. Por exemplo, a palavra-chave "te?t" permite a identificação de requisitos cuja descrição contém palavras como "test" e "text". Analogamente, o curinga "*" permite a detecção de palavras-chave que contenham zero ou mais caracteres na posição em que o curinga se encontra. Por exemplo, requisitos com as palavras "tests" ou "tester" em sua descrição podem ser identificados pela palavra-chave "test*".

Palavras-chave similares podem ser encontradas a partir de uma busca por similaridade. Esse tipo de busca utiliza o algoritmo de distância de edição ou "Levenshtein Distance"[6], que retorna um valor entre 0 (zero) e 1 (um), sendo que, quanto mais próximo de 1 (um), mais similares são as palavras comparadas. Para utilizar esse recurso, o pesquisador deve acrescentar o símbolo "~" no final da descrição da palavra-chave desejada, juntamente com o *threshold* exigido para se aceitar uma palavra-chave como similar a outra. Por exemplo, "roam~0.5" irá identificar requisitos que contenham palavras-chave cujo retorno do algoritmo de distância de edição seja maior ou igual a 0.5.

É importante salientar que o pesquisador deve utilizar os recursos de busca com curingas e busca por similaridade com parcimônia para evitar a incidência de muitos falsos positivos.

Os operadores lógicos "AND", "OR" e "NOT" também são aceitos para construção de uma palavra-chave ou do agrupamento de palavras-chave. Por exemplo, "(cancelamento OR cancelar) AND reserva" recupera todos os requisitos que contenham as palavras-chave "cancelamento" OU "cancelar" E a palavra-chave "reserva", simultaneamente. Operadores lógicos e curingas podem ser utilizados em conjunto, assim, a palavra-chave anterior poderia ser reescrita como: "cancela* AND reserva".

Uma vez cadastrada a palavra-chave, ela aparecerá na lista de palavras-chave do interesse ativo. Uma vez que o conjunto de palavras-chave para um determinado interesse pode ser numeroso, faz-se necessário criar estratégias para aumentar a produtividade do pesquisador enquanto o mesmo cadastra/atualiza tais palavras. Para isso, foi implementada uma função de importação/exportação de palavras-chave a partir de arquivos de texto. Esse arquivo de texto deve conter as palavras-chave especificadas da seguinte forma: `<palavra-chave1>, <palavra-chave2>,..., <palavra-chaveN>`. Analogamente, o conjunto de palavras-chave de um determinado interesse pode ser exportado para arquivo de texto, de acordo com o formato especificado anteriormente. Durante a importação de um conjunto de palavras-chave, caso haja problemas relacionados à tentativa de se incluir palavras duplicadas ou que estejam na lista de *stopwords*, as palavras-chave envolvidas nesses problemas serão ignoradas.

Na Figura 5.19 é exibida a tela que apresenta o conjunto de palavras-chave do interesse "Segurança", segundo o catálogo da Figura 5.4. As palavras "autenticação" e "autorização" foram substituídas por "aut*ção". Caso haja mais do que uma palavra-chave

[6] Este algoritmo baseia-se na quantidade de operações de modificação (substituição, adição ou remoção) de caracteres necessária para converter uma palavra em outra. Quanto menos modificações forem necessárias, mais similares são as palavras comparadas (http://www.levenshtein.net/).

cadastrada para um determinado interesse, elas serão concatenadas por meio do operador lógico “OR” durante a busca por interesses do software.

Figura 5.19. Lista de palavras-chave para o interesse “Segurança”.

Gerenciamento das fontes de um interesse. Conforme pode ser visto na Figura 5.20, para cadastrar uma fonte para um interesse, o pesquisador deve informar, obrigatoriamente, seu nome (que deve ser único para o interesse ativo no momento) e seu tipo, que pode ser um “Catálogo”, “*Stakeholder*” ou um “Documento de Negócio”. Opcionalmente, o pesquisador pode informar uma descrição para essa fonte, que pode ser útil para facilitar a identificação dessa fonte. Por exemplo, se a fonte for um catálogo de requisitos não funcionais, o pesquisador pode informar a referência onde esse catálogo foi obtido, que são seus autores, entre outras informações.

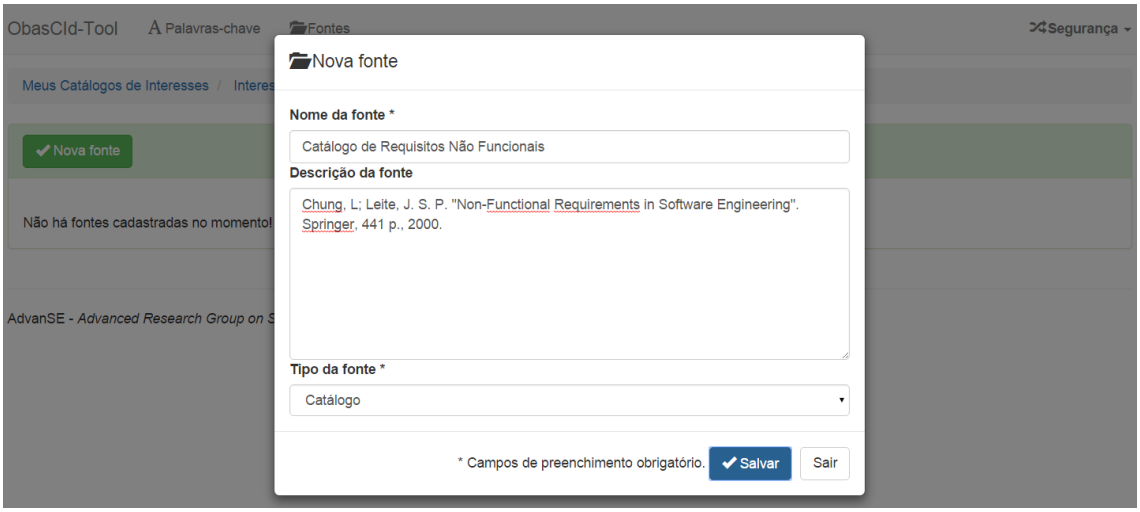


Figura 5.20. Cadastro de fontes de um interesse de software.

5.6.3 Gerenciamento dos relacionamentos entre interesses

Relacionamentos existentes entre diferentes interesses cadastrados para um mesmo catálogo de interesse de software podem ser criados e mantidos na ferramenta *ObasCId-Tool*. Os tipos possíveis de relacionamentos são “Composição”, “Dependência”, “Contribuição Negativa” e “Contribuição Positiva” e todos eles ocorrem entre um interesse fonte e um interesse alvo. Portanto, para que um relacionamento possa ser cadastrado para

um catálogo de interesses, deve haver pelo menos dois interesses pré-cadastrados para o mesmo.

Para cadastrar um novo relacionamento, as seguintes informações devem ser obrigatoriamente fornecidas (Figura 5.21): (i) o interesse fonte do relacionamento; (ii) o interesse alvo do relacionamento; e (iii) o tipo do relacionamento. Conforme foi explicado no Capítulo 3, a interpretação do que é um interesse alvo ou fonte depende do tipo do relacionamento instanciado. Por exemplo, em um relacionamento do tipo dependência, diz-se que o interesse fonte depende do interesse alvo. No caso de uma contribuição negativa, diz-se que o interesse fonte contribui negativamente para o interesse alvo. Para facilitar o entendimento por parte do pesquisador, a ferramenta *ObasCId-Tool* exibe o significado de cada relacionamento, juntamente com um exemplo prático de uso do mesmo, conforme pode ser visto na parte inferior da Figura 5.21. Essa mensagem se altera automaticamente, à medida que o pesquisador altera o tipo de relacionamento a ser criado.

Figura 5.21. Cadastro de relacionamentos entre interesses.

Uma vez cadastrado, o relacionamento é incluído na lista de relacionamentos, que pode ser vista na Figura 5.22. Nesta listagem, o nome do relacionamento aparece de acordo com a seguinte regra: `nome_do_interesse_fonte >> nome_do_interesse_alvo`. O tipo do relacionamento é representado por um ícone localizado entre o nome do relacionamento e o ícone de exclusão do relacionamento.

Na Figura 5.22 é apresentada a tela com os relacionamentos especificados no catálogo de interesses da Figura 4.4.

5.6.4 Relatórios de um catálogo de interesses de software

É possível gerar relatórios de um catálogo de interesses de software previamente cadastrado. Há dois tipos de relatórios disponíveis: (i) um relatório geral, que contém informações como o nome do catálogo, sua descrição e a lista de interesses desse catálogo,

com suas respectivas palavras-chave e relacionamentos, caso existam; e (ii) um relatório de relacionamentos, que contém um conjunto de matrizes de relacionamento que evidenciam os relacionamentos existentes entre os diferentes interesses de um catálogo de interesses.

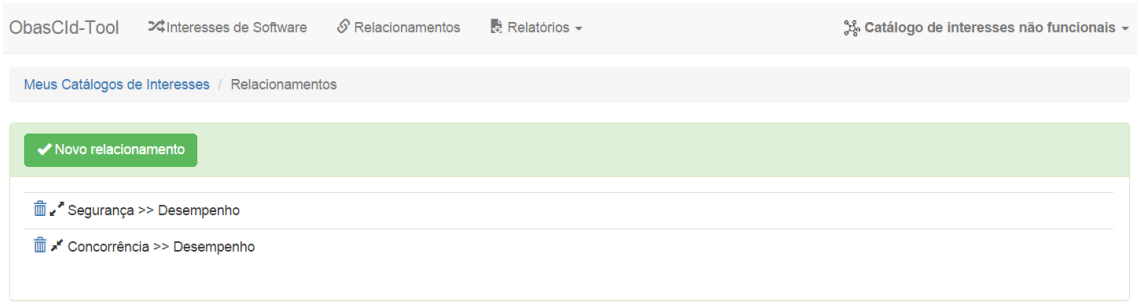


Figura 5.22. Listagem de relacionamentos entre interesses.

Na Figura 5.23 é exibida a tela com o relatório geral para o catálogo da Figura 4.4. Nele, cada interesse é representado em uma caixa na qual a parte superior contém seu nome, tipo e prioridade e na parte inferior estão as palavras-chave e fontes desse interesse, bem como os relacionamentos do mesmo com os demais interesses do software.

Figura 5.23. Exemplo de relatório geral de um catálogo de interesses de software.

Na Figura 5.24, por sua vez, é apresentada a tela com o relatório de relacionamentos desse catálogo de interesses. Nele, apenas a matriz de contribuição é apresentada, pois não há relacionamentos de dependência ou composição cadastrados para o catálogo em

questão. Essas matrizes são do tipo “Interesse vs. Interesse”, sendo que os interesses das linhas representam as fontes do relacionamento e os interesses das colunas, os alvos. Cada célula de uma matriz indica se há ou não um relacionamento entre os interesses correspondentes a essa célula. Na Figura 5.24 há dois relacionamentos de contribuição, um deles corresponde à contribuição negativa entre “Segurança” e “Desempenho” e o outro corresponde à contribuição positiva entre “Concorrência” e “Desempenho”.

Os símbolos que representam os relacionamentos de contribuição negativa ou positiva podem ser consultados na lista de ícones disponíveis na ferramenta (Figura 5.10 - B).

Figura 5.24. Exemplo de matrizes de relacionamentos entre interesses de um catálogo.

5.7 Módulo de Gerenciamento de Documentos de Requisitos

O enfoque da ferramenta *ObasCId-Tool* está na identificação de interesses a partir de requisitos de software. Sendo assim, uma importante função dessa ferramenta é o gerenciamento de documentos de requisitos. Essa funcionalidade pode ser utilizada por qualquer pesquisador cadastrado e autenticado na ferramenta.

Como pode ser visto na Figura 5.25, para adicionar um novo documento de requisitos, o pesquisador deve informar, obrigatoriamente, o nome do documento (que deve ser único) e o seu tipo de licença, que pode ser “Pública” ou “Privada”. O uso de tipos de licença para documentos de requisitos segue os mesmos princípios do uso de tipos de licença para catálogos de interesses de software e por isso não será comentado novamente.

Opcionalmente, o usuário poderá cadastrar uma descrição para seu documento de requisitos.

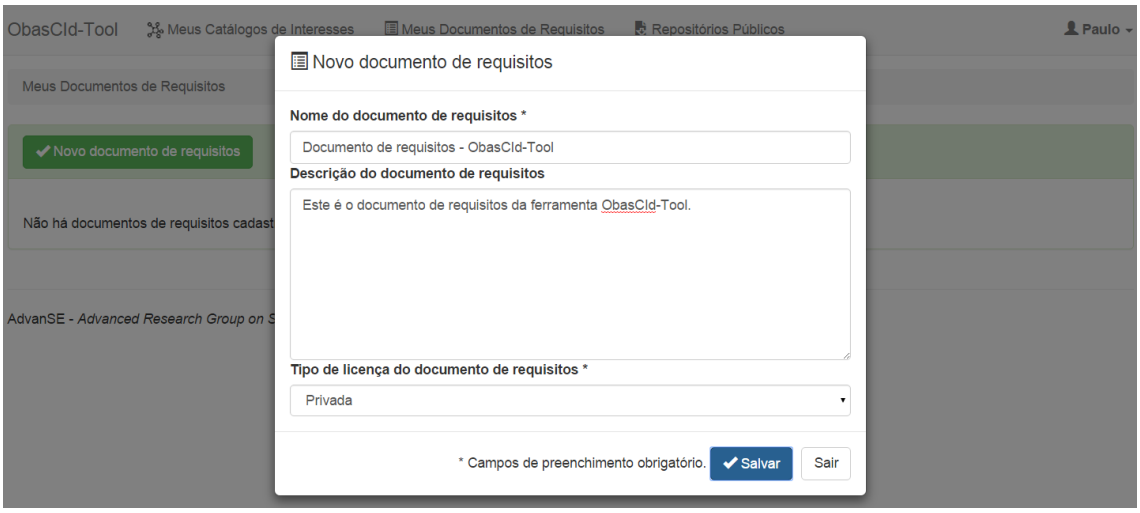


Figura 5.25. Cadastro de documentos de requisitos.

A listagem de documentos de requisitos segue a mesma estrutura da listagem de catálogos de interesses (Figura 5.8) e por isso não será apresentada. Os elementos que podem ser gerenciados para um documento de requisitos são “Requisitos”, “Dependências entre requisitos”, “Unidades de Identificação” e “Relatório”, que correspondem respectivamente à: (i) tela para gerenciamento de requisitos relacionados com o documento de requisitos ativo no momento; (ii) tela para gerenciamento dos relacionamentos de dependência existentes entre diferentes requisitos do documento de requisitos ativo no momento; (iii) tela para gerenciamento de unidades de identificação relacionadas ao documento de requisitos ativo no momento; e (iv) tela com o relatório geral do documento de requisitos ativo no momento.

5.7.1 Gerenciamento de requisitos

Para cadastrar um requisito, o pesquisador deve informar, obrigatoriamente, o nome do requisito, a descrição e o tipo do mesmo, que pode ser “Funcional” ou “Não funcional” (Figura 5.26). O nome do requisito é útil para facilitar sua identificação nas demais telas da ferramenta.

Uma vez cadastrado, o requisito aparecerá na listagem de requisitos do documento ativo. A Figura 5.27 apresenta os requisitos “RF-01” e “RNF-01”, do Quadro 5.1, cadastrados na ferramenta *ObasCId-Tool*.

De forma análoga ao que foi feito para as palavras-chave, a ferramenta *ObasCId-Tool* oferece apoio para importação/exportação de requisitos a partir de arquivos de texto.

No caso dos requisitos, esse arquivo de texto deve conter a descrição dos requisitos e de seus tipos, conforme ilustrado no Quadro 5.2.

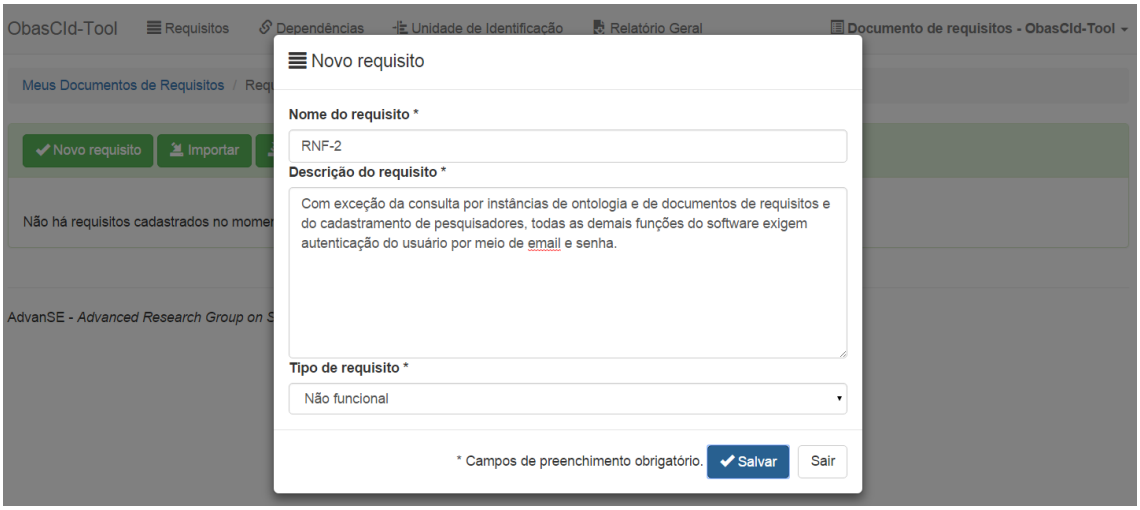


Figura 5.26. Cadastro de um requisito de software.

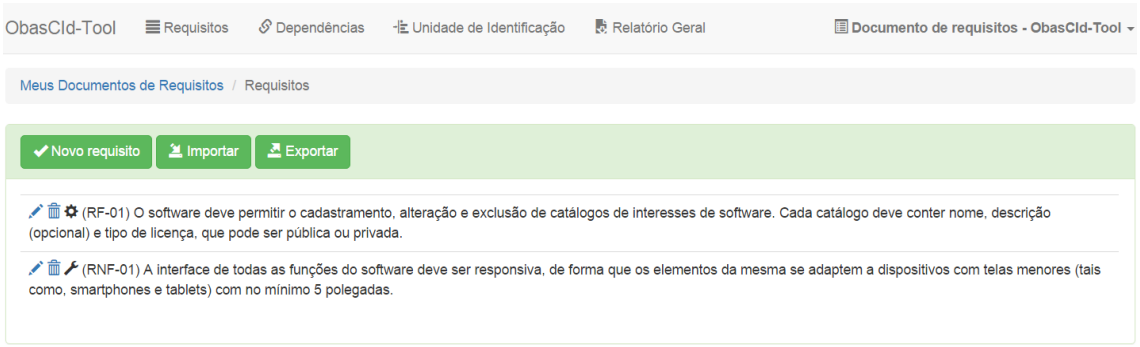


Figura 5.27. Listagem de requisitos do software.

Quadro 5.2. Modelo de entrada para importação de requisitos.

```
<tipo_do_requisito1: FR ou NFR>, <nome do requisito>
<descrição_do_requisito1>

<tipo_do_requisito2: FR ou NFR>, <nome do requisito>
<descrição_do_requisito2>

...

<tipo_do_requisitoN: FR ou NFR>, <nome do requisito>
<descrição_do_requisitoN>
```

Cada requisito é composto por duas ou mais linhas. A primeira linha refere-se ao tipo do requisito, representado pela sigla FR (*Functional Requirement*) ou NFR (*Non-Functional Requirement*) e pelo seu nome. O tipo e o nome do requisito devem ser separados por uma vírgula (,). As próximas *N* linhas não nulas correspondem à descrição desse requisito. Assim que for encontrada uma linha nula (em branco), considera-se que a descrição desse requisito terminou. O exemplo do Quadro 5.3 apresenta um arquivo com a descrição dos dois requisitos apresentados na Figura 5.27.

Quadro 5.3. Exemplo de arquivo para importação de requisitos.

```
FR, RF-01
O software deve permitir o cadastramento, alteração e exclusão de catálogos de
interesses de software. Cada catálogo deve conter nome, descrição (opcional) e
tipo de licença, que pode ser pública ou privada.

NFR, RNF-01
A interface de todas as funções do software deve ser responsiva, de forma que os
elementos da mesma se adaptem a dispositivos com telas menores (tais como,
smartphones e tablets) com no mínimo 5 polegadas.
```

Opcionalmente, o pesquisador pode especificar a dependência entre requisitos previamente cadastrados na ferramenta. Para isso, ele deve especificar obrigatoriamente o requisito fonte e o requisito alvo do relacionamento, sendo que o requisito fonte depende do requisito alvo. Por exemplo, de acordo com o Quadro 5.1, o requisito “RF-01” depende do requisito “RNF-01”. A Figura 5.28 apresenta a especificação dessa dependência.

Figura 5.28. Gerenciamento dos relacionamentos de dependência entre requisitos do software.

Analogamente aos relacionamentos entre interesses, cada dependência entre requisitos é apresentada na listagem de dependências da seguinte forma: `nome_do_requisito_fonte >> nome_do_requisito_alvo`. Para simplificar o processo de gerenciamento de dependências entre requisitos, o pesquisador também pode importar/exportar tais relacionamentos para arquivos de texto. Neste caso, o formato desse arquivo segue o que está ilustrado no Quadro 5.4.

Quadro 5.4. Modelo de entrada para importação de relacionamentos de dependência entre requisitos.

```
<nome_requisito_fonte1>, <nome_requisito_alvo1>, ..., <nome_requisito_alvo2>
...
<nome_requisito_fonteN>, <nome_requisito_alvo1>, ..., <nome_requisito_alvo2>
```

Cada linha desse arquivo representa um conjunto de relacionamentos de dependência entre um requisito fonte e um ou mais requisitos alvo, cujos nomes são separados por vírgula.

Com a finalidade de evidenciar um tipo de ocorrência que pode ocorrer durante o processo de identificação de interesses com a ferramenta *ObasCId-Tool*, o relacionamento de dependência entre os requisitos “RF-01” e “RNF-01” não será efetivado neste momento.

5.7.2 Relatório de um documento de requisitos

Um relatório geral de um documento de requisitos previamente cadastrado pode ser gerado. Na Figura 5.29 pode ser visualizado um relatório que contém as seguintes informações: (i) o nome e a descrição do documento de requisitos; (ii) a lista de requisitos com seus respectivos nome, descrições e tipo; (iii) a lista de requisitos dos quais ele depende.

Figura 5.29. Exemplo de um relatório geral de um documento de requisitos.

5.8 Módulo de Identificação e Classificação de Interesses

Apesar de ser tratado como um módulo distinto do módulo de gerenciamento de documentos de requisitos, as telas do módulo de identificação de interesses estão co-localizadas às telas desse módulo. Optou-se por isso, uma vez que a identificação de interesses de um software está intimamente relacionada ao seu documento de requisitos.

Este módulo implementa as quatro atividades propostas para a fase de identificação e classificação de interesses da abordagem *ObasCId*. Para implementar essas atividades, criou-se o conceito de “Unidade de Identificação”. Uma unidade de identificação consiste em um registro com as informações necessárias para que a identificação de interesses ocorra. Essas informações são (Figura 5.30): (i) nome da unidade de identificação (que deve ser único); (ii) um documento de requisitos; e (iii) um catálogo de interesses de software. Para que uma unidade de identificação seja criada, o pesquisador deve ter selecionado, a priori, um documento de requisitos. Sendo assim, essa informação não é apresentada na Figura 5.30.

A ideia de organizar o processo de identificação de interesses em unidades de identificação permite que o pesquisador teste diferentes cenários para identificação dos

interesses de software, como por exemplo, criando unidades de identificação para testar a efetividade de catálogos de interesses de software diferentes. Um exemplo prático dessa situação pode ocorrer quando um catálogo de interesses de software está sendo evoluído e o pesquisador quer ter certeza de que a cobertura e precisão da nova versão do catálogo são superiores às da versão antiga.

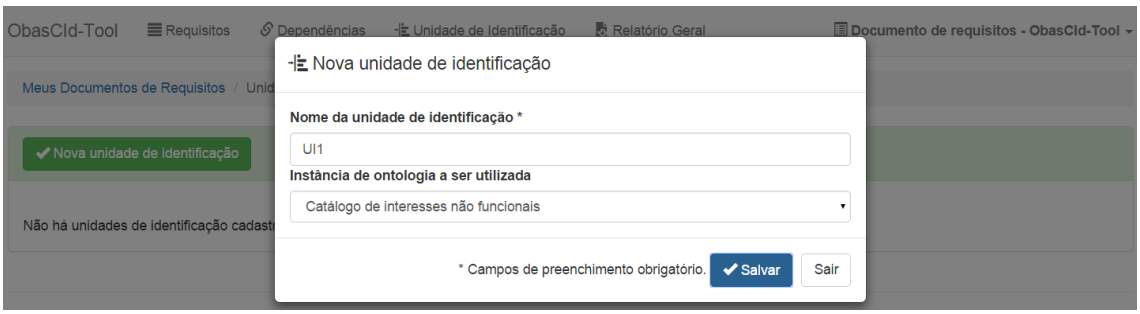


Figura 5.30. Cadastro de unidades de identificação.

O cadastro de unidades de identificação, entretanto, não garante a identificação e classificação dos interesses do software. Para isso, o pesquisador deve "executar" a unidade de identificação (ícone destacado na Figura 5.31).

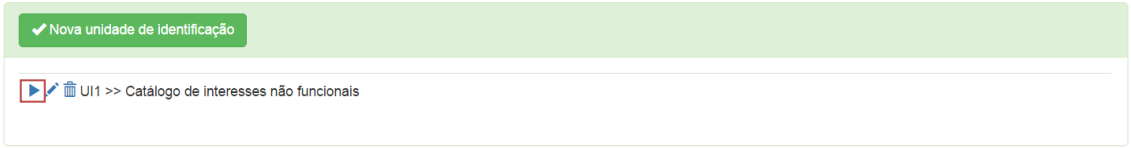


Figura 5.31. Listagem de unidades de identificação.

Para ilustrar os resultados do processo de identificação e classificação de interesses com a ferramenta *ObasCId-Tool*, o trecho do documento de requisitos da mesma (Quadro 5.1), bem como o catálogo de interesses não funcionais da Figura 4.4 foram utilizados. Além disso, esse catálogo de interesses não funcionais foi complementado com instâncias de interesses funcionais, criadas a partir do conhecimento de especialistas sobre a ferramenta *ObasCId-Tool*.

Essas instâncias, apresentadas na Figura 5.32, correspondem a cinco interesses funcionais, referentes aos cinco módulos da ferramenta *ObasCId-Tool*. Além disso, os relacionamentos de dependências entre os módulos da ferramenta, conforme ilustrado na Figura 5.1, foram transformados em quatro relacionamentos de dependência entre interesses desse catálogo. Um conjunto preliminar de 9 (nove) palavras-chave foi elaborado e relacionado aos interesses pré-existentes.

Inicialmente, o catálogo descrito acima foi executado apenas sobre os requisitos "RF-01" e "RNF-01" do Quadro 5.1. A saída retornada pela ferramenta *ObasCId-Tool* pode ser visualizada na Figura 5.33. Alguns elementos da tela da Figura 5.33 são comuns a qualquer processo de identificação e classificação de interesse na ferramenta *ObasCId-Tool*, e por isso precisam ser explicados.

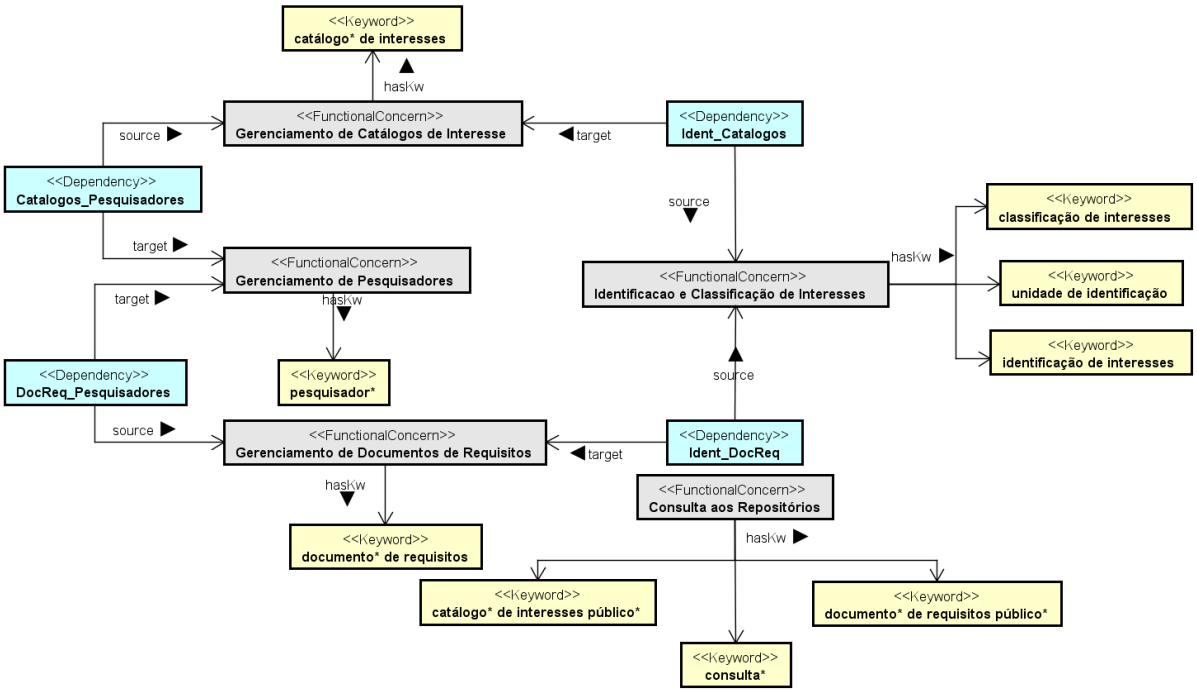


Figura 5.32. Interesses funcionais relacionados à ferramenta *ObasCId-Tool*.

Figura 5.33. Saída do processo de identificação de interesses de software com a ferramenta *ObasCId-Tool*.

A Figura 5.33 (A) apresenta o *menu* principal do módulo de identificação e classificação de interesses da ferramenta *ObasCId-Tool*. A primeira opção desse *menu*

“Interesses identificados” apresenta os dados que podem ser vistos nas partes C, D e E desta figura. A opção de *menu* “Interesse principal” permite a especificação dos interesses principais de cada requisito do software. Por fim, a opção “Matrizes” permite a visualização das matrizes de entrelaçamentos e contribuição entre interesses identificados no software.

A Figura 5.33 (B) apresenta o nome da unidade de identificação executada pelo pesquisador. A Figura 5.33 (C) apresenta as ocorrências encontradas durante o processo de identificação de interesses do software. A Figura 5.33 (D) apresenta um resumo do processo de identificação, destacando-se a lista de interesses advindos do catálogo utilizado, a lista de interesses identificados, a quantidade de requisitos de software existentes e a quantidade de requisitos afetados por algum interesse do catálogo, entre outros. A Figura 5.33 (E) destaca a funcionalidade de filtragem de requisitos, que será explicada mais adiante neste texto. Por fim, a Figura 5.33 (F) apresenta a listagem de requisitos do software, juntamente com os interesses identificados para os mesmos. Além do nome dos interesses, apresenta-se também o trecho da descrição do requisito que permitiu a identificação daquele interesse.

Como pode ser visto na Figura 5.33 (D, F), apenas dois dos doze interesses foram identificados, a saber, “Persistência” e “Gerenciamento de Catálogos de Interesses”, estando eles relacionados a um único requisito, o “RF-01”. Pode-se notar ainda que as palavras-chave “cadastramento”, “atualização” e “exclusão” foram responsáveis pela identificação do interesse “Persistência”, enquanto que a palavra-chave “catálogo de interesses” permitiu que o interesse “Gerenciamento de Catálogos de Interesses” fosse identificado.

A partir da Figura 5.33 (C), é possível notar que três ocorrências apareceram durante o processo de identificação de interesses. Essas ocorrências estão em conformidade com a abordagem *ObasCId*, conforme explicado no Capítulo 4 desta tese, por isso elas aparecem identificadas como ocorrências do tipo I, II, III ou IV. Quanto ao tipo das ocorrências, ERRO ou ALERTA, optou-se por apresentar as ocorrências de ERRO em uma caixa de mensagens vermelha e as ocorrências de ALERTA em uma caixa de mensagens amarela (um exemplo de mensagem de alerta é apresentado mais adiante nesta seção).

É importante salientar que, por questões de simplicidade, ocorrências de um mesmo tipo são aglutinadas em uma mesma mensagem. Por exemplo, as duas primeiras ocorrências de erro dizem respeito à não-identificação do interesse principal dos requisitos “RF-01” e “RNF-01”. No caso do requisito “RF-01”, há dois interesses identificados, mas não está explícito qual é o seu interesse principal. Para o requisito “RNF-01” não houve interesses identificados, pois, como pode ser visto na descrição desse requisito, trata-se de um requisito referente ao interesse não funcional “Responsividade”, que não é contemplado no catálogo de interesses utilizado.

Conforme explicado no Capítulo 4, para cada ocorrência há uma justificativa e várias recomendações para correção do problema identificado, caso necessário. A ferramenta *ObasCId-Tool* disponibiliza essas informações para que o pesquisador possa se orientar durante a resolução das ocorrências identificadas. Clicando-se no *link* disponível acima da lista de ocorrências, a tela da Figura 5.34 será apresentada ao pesquisador com os possíveis tipos de ocorrência e as explicações sobre eles.

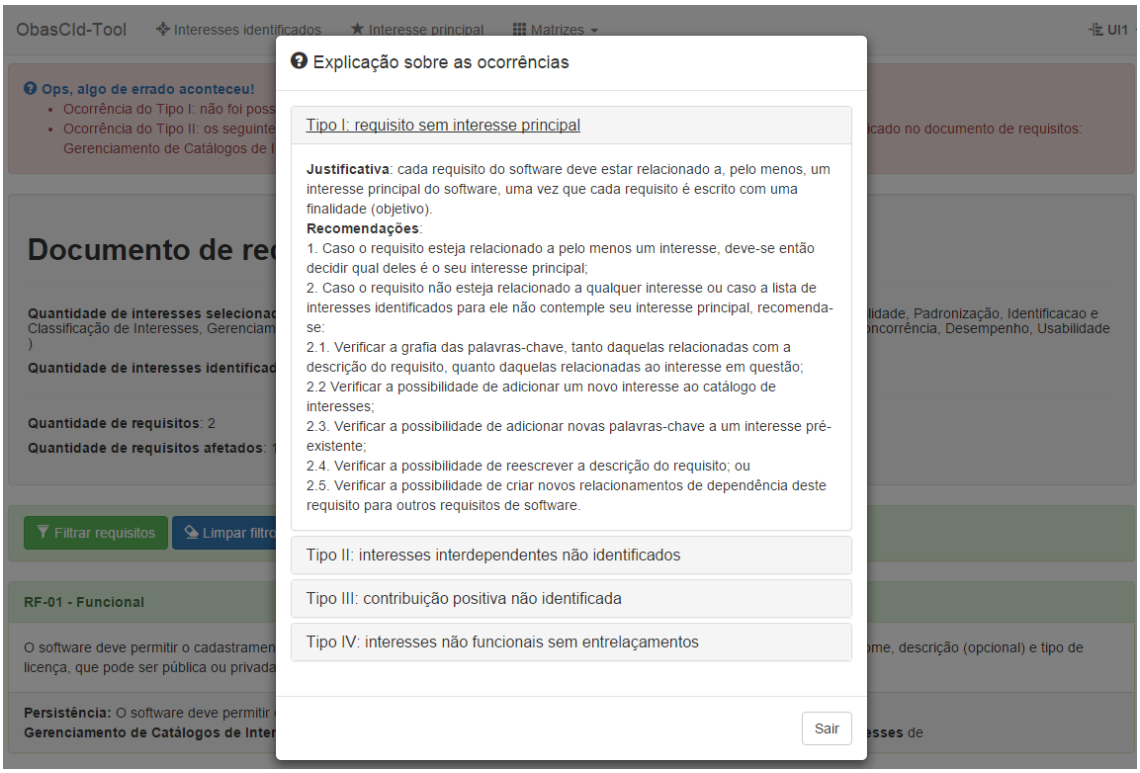


Figura 5.34. Explicações sobre as ocorrências do processo de identificação de interesses da abordagem *ObasCId*.

Para o caso do requisito “RNF-01”, a recomendação mais adequada é adicionar o interesse “Responsividade” ao catálogo de interesses em uso. Para isso, catálogos, documentos de negócios e a ajuda de especialistas sobre esse interesse podem ser utilizados. Nesta tese, consultou-se a documentação sobre “responsividade”, oferecida pelo W3C (*World Wide Web Consortium*), um consórcio de empresas responsáveis pela criação de padrões para melhor utilização da Web (W3C, 2015). Segundo o W3C, responsividade significa a adaptação da interface do software a diferentes tamanhos de telas (*screens*), como de *desktops*, *tablets*, *smartphones*, entre outros. Ainda segundo o W3C, um dos objetivos da responsividade é aprimorar a experiência do usuário enquanto o mesmo interage com o software.

Com base nestas informações, o catálogo de interesses utilizado foi adaptado, incorporando-se o interesse “Responsividade”, conforme ilustra o diagrama da Figura 5.35. Além disso, uma vez que a responsividade visa a aprimorar a experiência do usuário

enquanto o mesmo interage com o software, foi criado também um relacionamento de contribuição positiva do interesse “Responsividade” para o interesse “Usabilidade”.

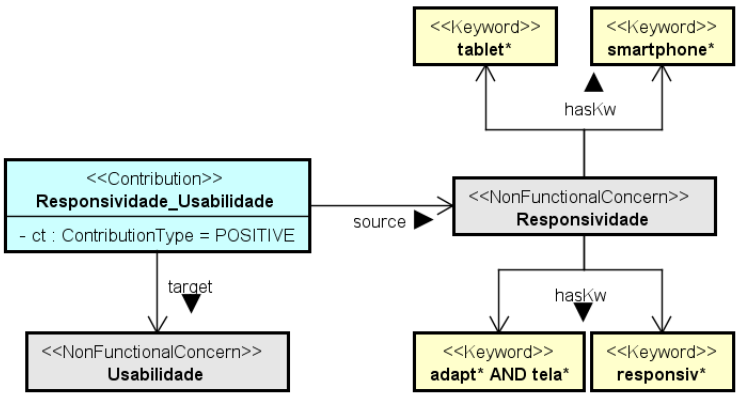


Figura 5.35. Elementos relacionados ao interesse “Responsividade”.

Uma vez tendo feito isso, o requisito “RNF-01” passou a ser contemplado pelo interesse “Responsividade”, contudo uma nova ocorrência apareceu (Figura 5.36). O interesse “Responsividade” é do tipo não funcional e nenhum requisito funcional do software está sendo contemplado por ele. Essa é uma situação de alerta, pois interesses não funcionais são fortes candidatos a apresentarem comportamento transversal (Sampaio *et al*., 2007). Assim, a ferramenta *ObasCId-Tool* gera uma mensagem de alerta ao engenheiro de software, comunicando-o sobre essa situação.

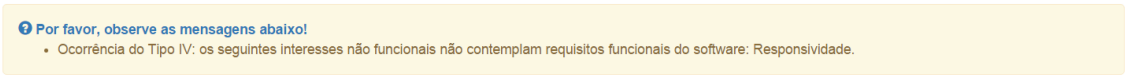


Figura 5.36. Mensagem com uma ocorrência do tipo IV.

Para resolver essa situação, uma das recomendações (Quadro 4.12 do Capítulo 4) é verificar a dependência entre requisitos do software. Neste caso, a característica de responsividade deve estar presente em todas as telas da ferramenta. Assim, vários outros requisitos do software, tais como o “RF-01” dependem do requisito relacionado a esse interesse. Uma vez especificada a dependência entre esses requisitos, conforme explicado na Seção 5.7.1, essa mensagem de alerta é removida e o interesse “Responsividade” passa a contemplar o requisito funcional “RF-01” também.

Voltando-se à ocorrência relacionada à inexistência de um interesse principal para os requisitos “RF-01” e “RNF-01”, tem-se que, conforme explicado no Capítulo 4, cada requisito do software deve ser escrito com uma finalidade (propósito, interesse). Para especificar qual é o interesse principal de um determinado requisito, o pesquisador pode acessar o *menu* “Interesse principal” da Figura 5.33. Para cadastrar um interesse principal, o pesquisador deve informar, obrigatoriamente, o requisito desejado e o seu interesse principal (Figura 5.37).

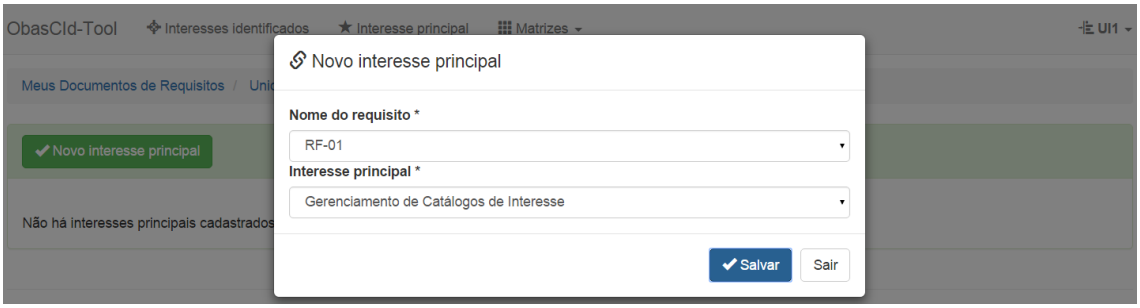


Figura 5.37. Cadastro do interesse principal de um requisito.

No caso do requisito “RF-01”, seu interesse principal é “Gerenciamento de Catálogos de Interesses”, uma vez que ele foi escrito com o intuito de indicar quais são as funções de gerenciamento disponíveis para um catálogo de interesses, bem como os dados necessários para gerenciá-los. No caso do requisito “RNF-01” o único interesse relacionado a ele, a saber, “Responsividade”, é também o seu interesse principal, uma vez que tal requisito foi escrito para especificar o comportamento de responsividade da ferramenta *ObasCId-Tool*.

Uma vez especificados os interesses principais destes requisitos, as ocorrências do tipo I são removidas da lista de ocorrências e a lista de requisitos e interesses identificados é atualizada, destacando-se os interesses principais de cada requisito (Figura 5.38).

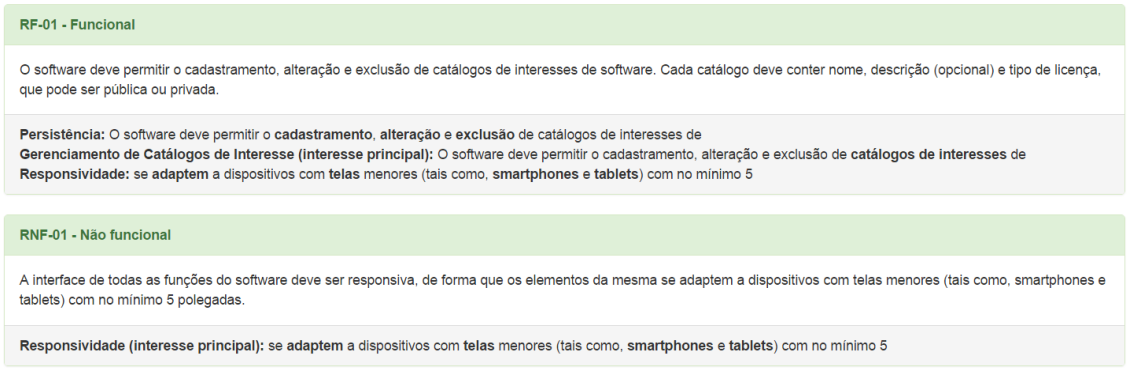


Figura 5.38. Lista de requisitos e interesses identificados atualizada após a especificação dos interesses principais de cada requisito.

A última ocorrência a ser analisada da lista de ocorrências da Figura 5.33 diz respeito à não-identificação de interesses interdependentes (ocorrência do tipo II). Como pode ser visto no catálogo de interesses da Figura 5.32, o interesse “Gerenciamento de Catálogos de Interesses” depende do interesse “Gerenciamento de Pesquisadores”. Isso porque, para se gerenciar um catálogo de interesses de software, o pesquisador deve estar devidamente cadastrado no software. O interesse “Gerenciamento de Catálogos de Interesses” foi identificado no documento de requisitos, mas o interesse “Gerenciamento de Pesquisadores” não. Por esse motivo, essa última ocorrência foi gerada. No caso desse exemplo, isso ocorreu, pois nem todos os requisitos do software foram cadastrados na ferramenta *ObasCId-Tool*.

Para resolver esse problema, cadastrou-se o requisito “RF-02” e sua dependência com o requisito “RNF-01” do Quadro 5.1 na ferramenta. Ao reexecutar a unidade de identificação, o requisito “RF-02” foi relacionado aos interesses “Gerenciamento de Pesquisadores” e “Persistência”, devido à ocorrência das palavras-chave “pesquisador” e “cadastramento” na descrição do requisito. A partir da especificação do interesse “Gerenciamento de Pesquisadores” como interesse principal desse requisito, a lista de ocorrências do processo de identificação ficou vazia.

Considerando-se que o processo de identificação está completo, o pesquisador poderá verificar a situação do entrelaçamento entre os interesses identificados. Para isso, ele pode acessar o *menu* “Matrizes” e escolher a opção “Matriz de entrelaçamentos”. A Figura 5.39 exibe a matriz de entrelaçamentos gerada com os dados do exemplo anterior. É importante salientar que o engenheiro de software não será impedido de gerar a matriz de entrelaçamentos caso haja ocorrências de erro durante a atividade de identificação de interesses, contudo, esta matriz pode apresentar resultados incorretos ou insatisfatórios caso as ocorrências de erro não sejam verificadas e tratadas anteriormente.

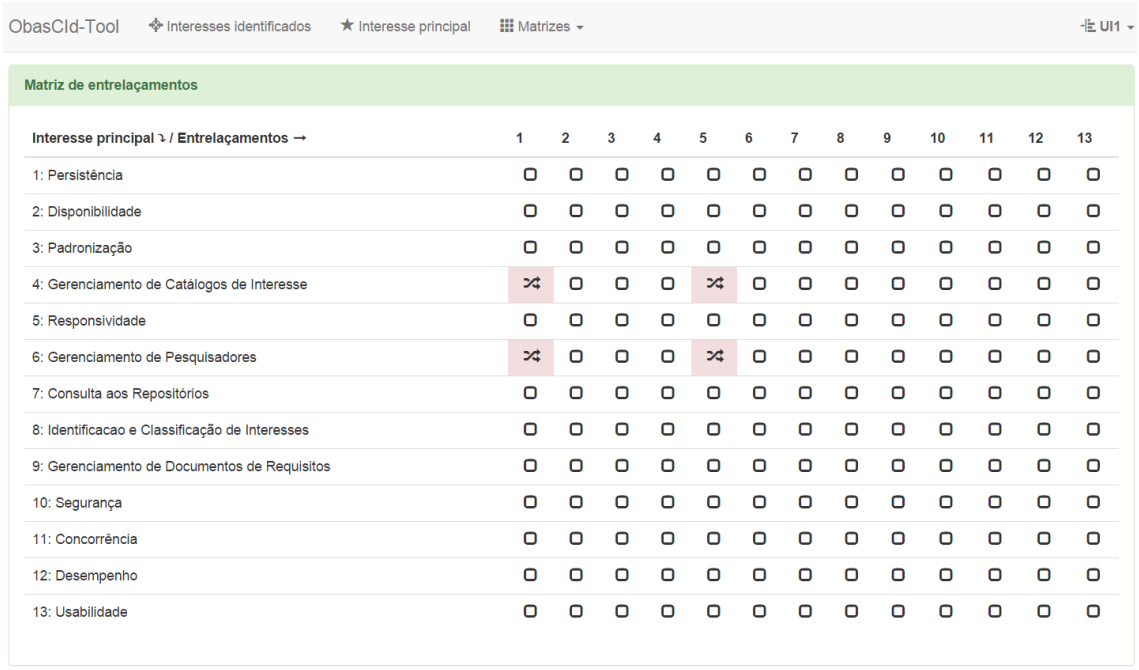


Figura 5.39. Exemplo de uma matriz de entrelaçamentos entre interesses.

Conforme comentado no Capítulo 4, uma matriz de entrelaçamentos é do tipo “Interesse vs. Interesse”, na qual cada célula “[C1, C2]” dessa matriz, quando marcada, representa que o interesse “C2” entrecorta o comportamento do interesse “C1”. No caso da Figura 5.39, nota-se que os interesses “1: Persistência” e “5: Responsividade” entrecortam os interesses “4: Gerenciamento de Catálogos de Interesses” e “6: Gerenciamento de Pesquisadores” (células destacadas em vermelho da Figura 5.39). Isso porque “Persistência” e “Responsividade” estão associados a requisitos para os quais os interesses

principais são “4: Gerenciamento de Catálogos de Interesses” e “6: Gerenciamento de Pesquisadores”.

Caso o pesquisador deseje conhecer os requisitos nos quais esses interesses estão entrelaçados, basta voltar à lista de requisitos e interesses identificados e aplicar o “filtro de requisitos”. Por exemplo, para saber em quais requisitos o interesse “Persistência” está entrecortando o comportamento do interesse “Gerenciamento de Pesquisadores”, o pesquisador inicialmente deve filtrar os requisitos escolhendo “Gerenciamento de Pesquisadores” como interesse principal e “Persistência” como interesse transversal, conforme pode ser visto na Figura 5.40. Assim, apenas os requisitos compatíveis com esse filtro serão apresentados (Figura 5.41).

O “filtro de requisitos” oferece também a opção de se informar apenas o interesse principal ou transversal. Isso permite ao pesquisador descobrir: (i) quais são os requisitos entrecortados por determinado interesse, independentemente do interesse principal dos mesmos; e (ii) quais são os requisitos de um determinado interesse principal, independentemente de quem os entrecorta.

Figura 5.40. Exemplo de aplicação do “filtro de requisitos”.

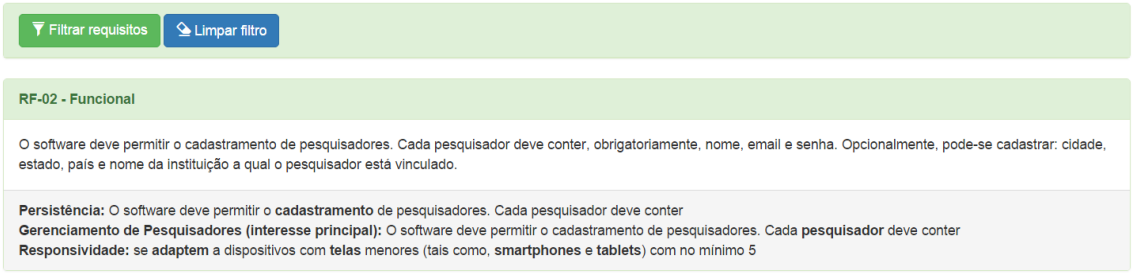


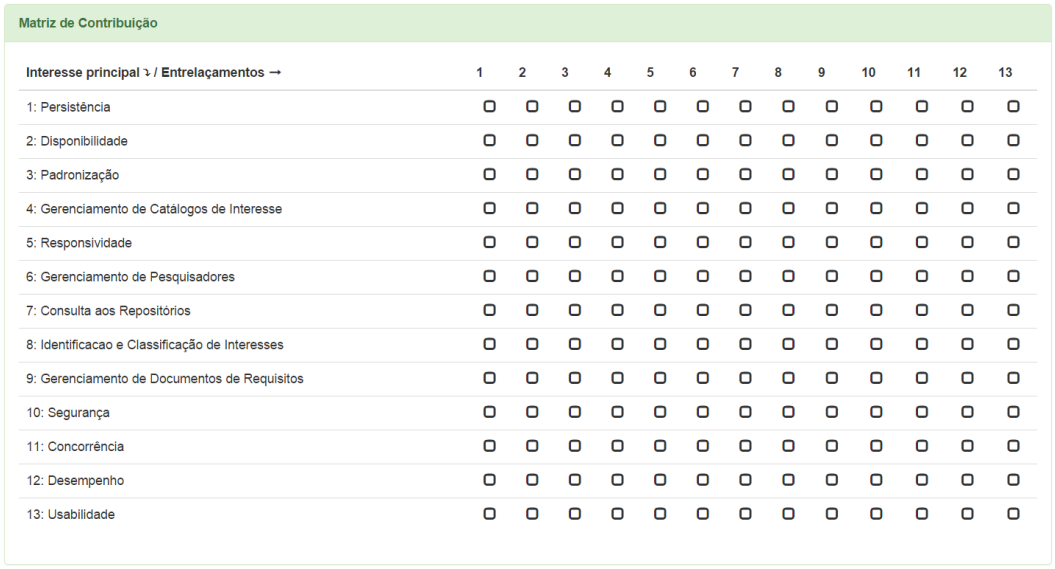
Figura 5.41. Lista de requisitos e interesses identificados após a aplicação do “filtro de requisitos”.

O último recurso do “Módulo de Identificação e Classificação de Interesses” a ser apresentado diz respeito à matriz de contribuição. Essa também é uma matriz “Interesse vs. Interesse”, em que cada célula “[C1, C2]” descreve os relacionamentos de contribuição para as quais “C2” é o interesse alvo e cujo interesse fonte desse relacionamento também entrecorta o interesse “C1”.

É importante salientar que o fato de haver relacionamentos de contribuição documentados no catálogo de interesses de software utilizado não significa que esses

relacionamentos aparecerão nesta matriz de contribuição. A matriz que apresenta os relacionamentos de contribuição entre interesses existentes no catálogo de interesses é aquela apresentada na Seção 5.6.4 deste capítulo. A matriz de contribuição comentada nesta seção tem por finalidade mostrar ao engenheiro de software os relacionamentos de contribuição entre interesses identificados no software e que precisam de sua atenção.

Conforme pode ser visto na Figura 5.42, não há relacionamentos de contribuição entre os interesses identificados no exemplo anterior. Isso ocorreu porque não há relacionamentos de contribuição entre os interesses “Persistência” e “Responsividade” no catálogo de interesses utilizado.


Matriz de Contribuição

Interesse principal ↘ / Entrelaçamentos →	1	2	3	4	5	6	7	8	9	10	11	12	13
1: Persistência	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢
2: Disponibilidade	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢
3: Padronização	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢
4: Gerenciamento de Catálogos de Interesse	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢
5: Responsividade	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢
6: Gerenciamento de Pesquisadores	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢
7: Consulta aos Repositórios	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢
8: Identificacao e Classificação de Interesses	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢
9: Gerenciamento de Documentos de Requisitos	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢
10: Segurança	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢
11: Concorrência	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢
12: Desempenho	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢
13: Usabilidade	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢

Figura 5.42. Exemplo de uma matriz de contribuição.

Em outras palavras, durante o desenvolvimento dos interesses “Gerenciamento de Catálogos de Interesse” e “Gerenciamento de Pesquisadores” o engenheiro de software pode raciocinar sobre “Persistência” e “Responsividade” de forma isolada, uma vez que não há influência mútua entre esses interesses.

Considerando que o requisito “RNF-03” do Quadro 5.1 seja acrescentado ao software do exemplo anterior e que os relacionamentos de dependência entre os requisitos “RF-01” e “RNF-03” e “RF-02” e “RNF-03” também sejam especificados, as matrizes de entrelaçamentos e de contribuição serão atualizadas, conforme apresentam a Figura 5.43 (A) e a Figura 5.43 (B), respectivamente.

Como pode ser visto, o interesse “13: Usabilidade” também passa a afetar os interesses “4: Gerenciamento de Catálogos de Interesse” e “6: Gerenciamento de Pesquisadores”, devido ao relacionamento de dependência entre os requisitos “RF-01” e “RF-02” com “RNF-03”. No caso da matriz de contribuição, pode-se notar a existência de uma contribuição positiva (+) do interesse “5: Responsividade” para o interesse “13:

Usabilidade". Além disso, essa contribuição ocorre sobre os interesses "4: Gerenciamento de Catálogos de Interesse" e "6: Gerenciamento de Pesquisadores".

Matriz de entrelaçamentos

Interesse principal ↴ / Entrelaçamentos →	1	2	3	4	5	6	7	8	9	10	11	12	13
1: Persistência	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
2: Disponibilidade	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
3: Padronização	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
4: Gerenciamento de Catálogos de Interesse	⤮	☐	☐	☐	⤮	☐	☐	☐	☐	☐	☐	☐	⤮
5: Responsividade	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
6: Gerenciamento de Pesquisadores	⤮	☐	☐	☐	⤮	☐	☐	☐	☐	☐	☐	☐	⤮
7: Consulta aos Repositórios	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
8: Identificacao e Classificação de Interesses	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
9: Gerenciamento de Documentos de Requisitos	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
10: Segurança	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
11: Concorrência	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
12: Desempenho	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
13: Usabilidade	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

(A)

Matriz de Contribuição

Interesse principal ↴ / Entrelaçamentos →	1	2	3	4	5	6	7	8	9	10	11	12	13
1: Persistência	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
2: Disponibilidade	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
3: Padronização	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
4: Gerenciamento de Catálogos de Interesse	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	5 (+)
5: Responsividade	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
6: Gerenciamento de Pesquisadores	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	5 (+)
7: Consulta aos Repositórios	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
8: Identificacao e Classificação de Interesses	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
9: Gerenciamento de Documentos de Requisitos	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
10: Segurança	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
11: Concorrência	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
12: Desempenho	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
13: Usabilidade	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

(B)

Figura 5.43. Matrizes de entrelaçamentos e contribuição atualizadas após a inclusão do requisito "RNF-03".

5.9 Identificação e Classificação dos Interesses da ferramenta *ObasCId-Tool*

A partir dos resultados apresentados na Seção 5.8, todos os demais requisitos da ferramenta *ObasCId-Tool*, disponíveis no Quadro A.8.1 do Apêndice A desta tese, foram cadastrados.

Conforme explicado na Seção 5.8, para executar uma unidade de identificação, o pesquisador deve ter, no mínimo, um catálogo de interesses de software e um documento de requisitos cadastrados. Contudo, a maneira como o pesquisador realiza o cadastramento dos elementos desses artefatos pode facilitar seu entendimento sobre possíveis situações de erro durante o processo de identificação e classificação de interesses. Assim, recomenda-se não realizar o cadastramento dos relacionamentos de dependência entre requisitos na primeira execução da unidade de identificação.

De acordo com a descrição dos elementos e axiomas da ontologia *O4C* (Capítulo 3) e com a descrição da abordagem *ObasCId* (Capítulo 4), os relacionamentos de dependência entre requisitos permitem com que um requisito herde os interesses identificados para outro requisito como seus interesses transversais. Assim, cadastrando esses relacionamentos logo no início do processo de identificação de interesses, pode-se omitir as situações em que não foram identificados interesses para um determinado requisito do software, por meio de palavras-chave. Essas situações são importantes pois podem evidenciar a necessidade de um refinamento da descrição do documento de requisitos ou do catálogo de interesses.

Dessa forma, na primeira execução da unidade de identificação sobre o documento de requisitos completo da ferramenta *ObasCId-Tool*, não foram incluídos os relacionamentos de dependência existentes entre esses requisitos. A Figura 5.44 apresenta o resumo do processo de identificação, bem como a lista de ocorrências gerada.

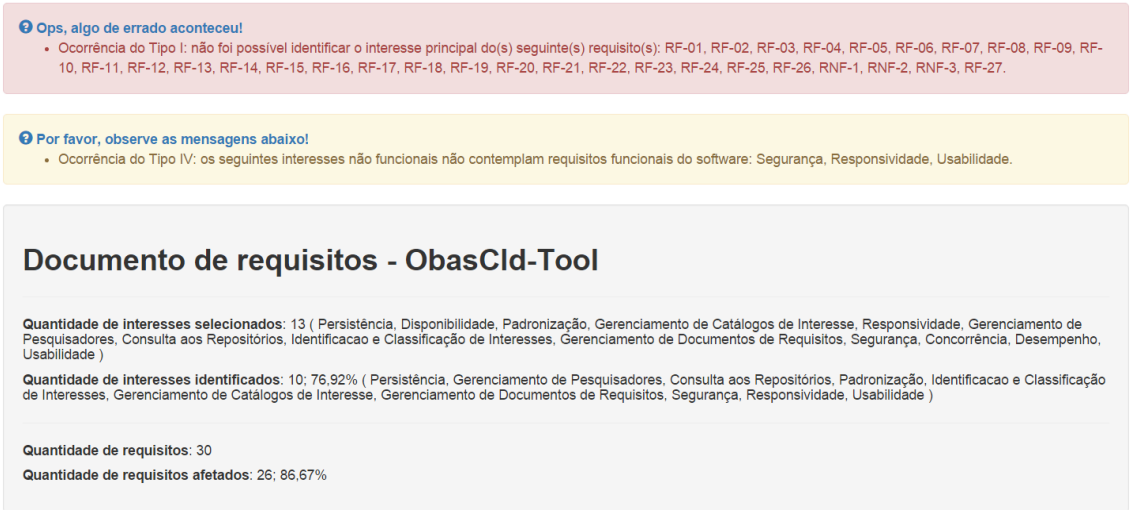


Figura 5.44. Saída do processo de identificação de interesses com os requisitos da ferramenta *ObasCId-Tool*.

Como pode ser visto, uma cobertura satisfatória dos requisitos da ferramenta foi obtida, pois aproximadamente 87% dos requisitos foram contemplados por algum tipo de interesse. Além disso, dos 13 (treze) interesses existentes no catálogo de interesses utilizado, 10 (dez) foram identificados. Com relação às ocorrências, a maioria está relacionada à não-especificação dos interesses principais dos requisitos do software (ocorrência do tipo I). Há ainda três ocorrências do tipo IV, que diz respeito à identificação

de interesses não funcionais que não entrecortam requisitos funcionais do software. A resolução destas ocorrências será discutida mais adiante nesta seção.

Quanto à lista de requisitos e interesses identificados, há algumas situações interessantes a serem discutidas. Os quatro requisitos para os quais não houve interesses identificados são “RF-10”, “RF-15”, “RF-16” e “RF-27”. O requisito “RF-27” diz respeito à filtragem de requisitos da lista de requisitos e interesses identificados. Esse requisito está relacionado ao interesse “Identificação e Classificação de Interesses”, contudo, não havia palavras no catálogo de interesse que fosse capaz de capturá-lo. Para resolver essa situação, a palavra-chave “lista de requisitos e interesses identificados”, que é um artefato da fase de identificação e classificação de interesses, foi adicionada ao catálogo.

Os requisitos “RF-15” e “RF-16” estão relacionados ao interesse “Gerenciamento de Catálogos de Interesses”. No catálogo de interesses utilizado, a palavra-chave “catálogos de interesse” foi cadastrada, porém na descrição destes requisitos, a expressão “catálogo de interesses” aparece. Esse erro poderia ter sido evitado por meio do uso de recursos como *wildcards* ou “busca por similaridade”, durante o cadastramento das palavras-chave do catálogo. Para resolver esse problema, a palavra-chave anterior foi substituída por “catálogo* AND interesses”.

O último requisito a ser considerado é o “RF-10”, que diz respeito à importação/exportação de palavras-chave de um interesse, ou seja, também está relacionado ao “Gerenciamento de Catálogos de Interesse”. Esse requisito não foi contemplado por esse interesse, pois faltam palavras-chave no catálogo de interesses capazes de identificar outros tipos de elementos relacionados aos catálogos de interesses de software. Para resolver esse problema, novas palavras-chave foram cadastradas para o interesse “Gerenciamento de Catálogos de Interesse”, tais como “palavras-chave”, “fontes”, entre outros.

Quanto às ocorrências do tipo IV, os três interesses funcionais do catálogo, a saber, “Responsividade”, “Usabilidade” e “Segurança” foram identificados, porém eles estão relacionados apenas a requisitos não funcionais do software. Esse problema foi solucionado após o cadastramento dos relacionamentos de dependências entre os requisitos do software, conforme especificado no Quadro A.8.1. Assim, restaram apenas as ocorrências que dizem respeito à não-especificação do interesse principal de cada requisito.

Com o auxílio de especialistas sobre o domínio da ferramenta *ObasCId-Tool*, os requisitos dessa ferramenta foram classificados quanto ao seu interesse principal, conforme pode ser visto no Quadro 5.5. Para isso, tais especialistas tiveram acesso à lista de requisitos e interesses identificados pela ferramenta *ObasCId-Tool*.

Quadro 5.5. Lista de requisitos da ferramenta *ObasCId-Tool* e seus interesses principais.

Requisito do software	Interesse Principal
RF-01	Gerenciamento de Catálogos de Interesses
RF-02	Gerenciamento de Pesquisadores
RF-03	Gerenciamento de Pesquisadores
RF-04	Gerenciamento de Pesquisadores
RF-05	Gerenciamento de Pesquisadores
RF-06	Gerenciamento de Documentos de Requisitos
RF-07	Gerenciamento de Catálogos de Interesses
RF-08	Gerenciamento de Catálogos de Interesses
RF-09	Gerenciamento de Catálogos de Interesses
RF-10	Gerenciamento de Catálogos de Interesses
RF-11	Gerenciamento de Catálogos de Interesses
RF-12	Gerenciamento de Documentos de Requisitos
RF-13	Gerenciamento de Documentos de Requisitos
RF-14	Identificação e Classificação de Interesses
RF-15	Gerenciamento de Catálogos de Interesses
RF-16	Gerenciamento de Catálogos de Interesses
RF-17	Gerenciamento de Documentos de Requisitos
RF-18	Identificação e Classificação de Interesses
RF-19	Identificação e Classificação de Interesses
RF-20	Identificação e Classificação de Interesses
RF-21	Identificação e Classificação de Interesses
RF-22	Consulta aos Repositórios
RF-23	Consulta aos Repositórios
RF-24	Consulta aos Repositórios
RF-25	Consulta aos Repositórios
RF-26	Gerenciamento de Catálogos de Interesses
RF-27	Identificação e Classificação de Interesses
RNF-01	Responsividade
RNF-02	Segurança
RNF-03	Usabilidade

Uma vez cadastrados os interesses principais de cada requisito do software, as ocorrência do tipo I foram todas resolvidas e como resultado, tem-se a matriz de entrelaçamentos da Figura 5.45.

Algumas observações a respeito da matriz da Figura 5.45 são:

i) a natureza transversal dos interesses não funcionais existentes no software foi evidenciada, uma vez que dos quatro interesses não funcionais identificados (“Persistência”, “Segurança”, “Responsividade” e “Usabilidade”), apenas “Segurança” não está entrelaçado com todos os interesses funcionais identificados no software. Isso ocorreu porque as funções especificadas pelos requisitos do módulo “Consulta aos Repositórios” não exigem que os usuários estejam autenticado na ferramenta, tornando os interesse “Consulta aos Repositórios” e “Segurança” não entrelaçados;

ii) a existência de interesses funcionais transversais também é evidenciada. Dos cinco interesses funcionais identificados, apenas os interesses “Identificação e

Classificação de Interesses” e “Consulta aos Repositórios” não entrecortam outros interesses do software. Conforme pode ser visto no diagrama de arquitetura da Figura 5.1, esses são os únicos módulos dos quais nenhum outro módulo depende, ou seja, a chance de haver entrelaçamentos de outros módulos com esses é realmente pequena; e

iii) dos nove interesses identificados, sete podem ser considerados candidatos a interesses transversais. Apenas os interesses não funcionais “Identificação e Classificação de Interesses” e “Consulta aos Repositórios” podem ser considerados interesses base.

Matriz de entrelaçamentos

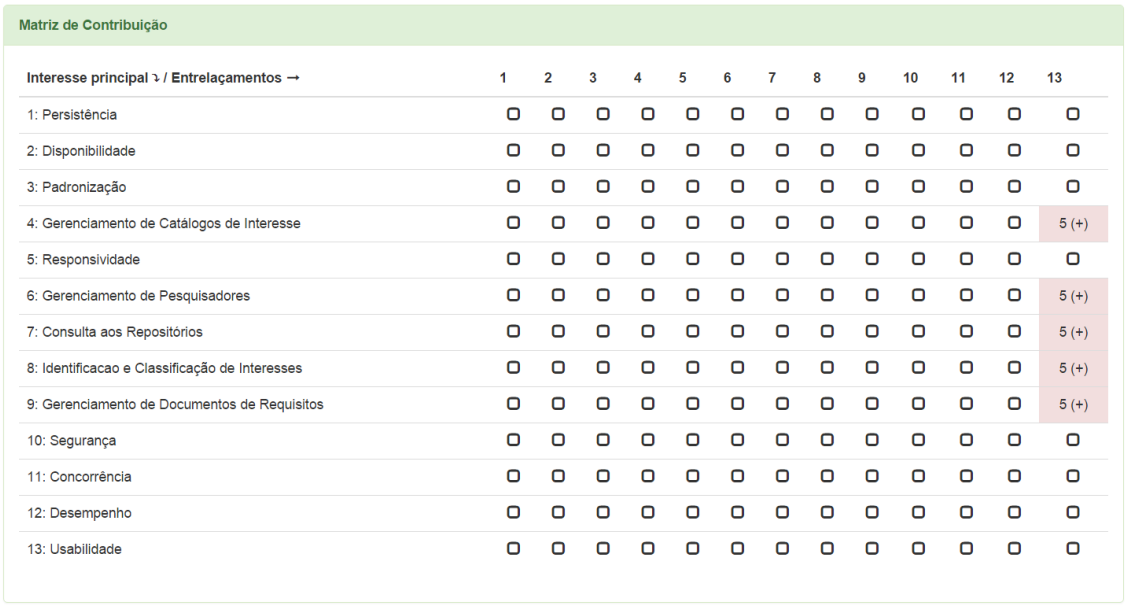
Interesse principal ↴ / Entrelaçamentos →	1	2	3	4	5	6	7	8	9	10	11	12	13
1: Persistência	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢
2: Disponibilidade	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢
3: Padronização	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢
4: Gerenciamento de Catálogos de Interesse	⤮	▢	▢	▢	⤮	⤮	⤮	▢	▢	⤮	▢	▢	⤮
5: Responsividade	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢
6: Gerenciamento de Pesquisadores	⤮	▢	⤮	⤮	⤮	▢	▢	▢	⤮	⤮	▢	▢	⤮
7: Consulta aos Repositórios	⤮	▢	▢	⤮	⤮	▢	▢	▢	⤮	▢	▢	▢	⤮
8: Identificacao e Classificação de Interesses	⤮	▢	▢	⤮	⤮	▢	▢	▢	⤮	⤮	▢	▢	⤮
9: Gerenciamento de Documentos de Requisitos	⤮	▢	▢	▢	⤮	▢	⤮	▢	▢	⤮	▢	▢	⤮
10: Segurança	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢
11: Concorrência	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢
12: Desempenho	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢
13: Usabilidade	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢	▢

Figura 5.45. Matriz de entrelaçamentos do processo de identificação e classificação de interesses da ferramenta *ObasCId-Tool*.

Contudo, há um ponto que merece a atenção do pesquisador, que é o entrelaçamento do interesse “3: Padronização” com o interesse “6: Gerenciamento de Pesquisadores”. Utilizando o “filtro de requisitos” com a opção “Gerenciamento de Pesquisadores” como interesse principal e “Padronização” como interesse transversal, o pesquisador terá acesso ao requisito onde tal entrelaçamento ocorre. Trata-se do requisito “RF-05”, que diz respeito ao cadastramento de opções padrão (*default*) para entrada de dados na ferramenta *ObasCId-Tool*. A palavra-chave “padr*” do interesse “Padronização” foi a responsável pela identificação desse interesse. Contudo, trata-se de um falso positivo, pois o termo “opções padrão” desse requisito possui um significado diferente do que se entende por “padronização”. Para remover esse falso positivo, o engenheiro de software pode refinar a lista de palavras-chave desse interesse ou reescrever o requisito em questão. Neste caso, optou-se por remover o termo “padrão” da descrição dos requisitos, deixando apenas o seu equivalente em inglês “*default*”. Outra opção seria especificar melhor o termo

"padrão" no catálogo de interesses, informando palavras-chave como "padrões AND codificação" e "padrões AND projeto", entre outras.

A matriz de contribuição gerada para a ferramenta *ObasCId-Tool* pode ser visualizada na Figura 5.46. O único ponto de atenção é a contribuição positiva exercida pelo interesse "5: Responsividade" sobre o interesse "13: Usabilidade", no contexto dos interesses 4, 6, 7, 8 e 9, ou seja, todos os cinco interesses funcionais do software. Isso quer dizer que, ao implementar mecanismos de responsividade para as telas relacionadas com essas funcionalidades, a usabilidade do software estará sendo aprimorada.



Matriz de Contribuição

Interesse principal ↴ / Entrelaçamentos →	1	2	3	4	5	6	7	8	9	10	11	12	13
1: Persistência	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
2: Disponibilidade	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
3: Padronização	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
4: Gerenciamento de Catálogos de Interesse	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	5 (+)
5: Responsividade	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
6: Gerenciamento de Pesquisadores	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	5 (+)
7: Consulta aos Repositórios	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	5 (+)
8: Identificacao e Classificação de Interesses	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	5 (+)
9: Gerenciamento de Documentos de Requisitos	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	5 (+)
10: Segurança	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
11: Concorrência	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
12: Desempenho	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
13: Usabilidade	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

Figura 5.46. Matriz de contribuição do processo de identificação e classificação de interesses da ferramenta *ObasCId-Tool*.

5.10 Características Técnicas e Mecanismos de Extensão

Um dos objetivos deste trabalho é o estabelecimento de uma compreensão mais ampla a respeito do domínio de interesses de software. Para que isso seja possível, acredita-se que é necessário facilitar o acesso e o compartilhamento de informações a respeito desse domínio. Dessa forma optou-se pelo desenvolvimento de uma aplicação *web* que pudesse ser executada em um servidor de aplicações e que o seu acesso fosse possível a partir de qualquer ponto com Internet. Isso se justifica pela impossibilidade de se reutilizar outras ferramentas para EROA existentes na literatura, uma vez que nenhum delas possuía esse enfoque.

ObasCId-Tool é uma ferramenta livre[7] e gratuita. Ela foi desenvolvida com base nas seguintes tecnologias livres: (i) a especificação *Java Server Faces* (JSF) e o *framework* para desenvolvimento de aplicações *web* denominado Mojarra[8], que implementa a especificação *JSF*: utilizado para desenvolvimento tanto do *backend* quanto do *frontend* da ferramenta *ObasCId-Tool*; (ii) o *framework* CSS para desenvolvimento de interfaces *web* responsivas, denominado *Bootstrap*[9]: fundamental para implementação do requisito não funcional "RNF-01" desta ferramenta; (iii) o Sistema Gerenciador de Bancos de Dados (SGBD) MySQL: utilizado como repositório para os catálogos de interesses e documentos de requisitos do software; e (iv) o motor de busca *Apache Lucene*[10]: tecnologia utilizada para implementação dos algoritmos de identificação de interesses a partir de palavras-chave. Ele oferece suporte à busca com *wildcards*, busca por similaridade e busca com operadores lógico, três mecanismos oferecidos pela ferramenta *ObasCId-Tool*.

Uma característica importante da *ObasCId-Tool* quanto à sua aplicabilidade a diferentes contextos linguísticos é a internacionalização (em inglês, *internationalization* ou *i18n*). A internacionalização permite que a ferramenta *ObasCId-Tool* seja utilizada em países que fazem uso da língua portuguesa (do Brasil) ou inglesa, uma vez que todos os rótulos e mensagens disponíveis na ferramenta encontram-se traduzidos para esses dois idiomas. O *framework* Mojarra possui suporte nativo à internacionalização, o que permite que a ferramenta seja facilmente estendida para outros idiomas.

Na ferramenta *ObasCId-Tool*, todas as cadeias de caracteres (*strings*) são lidas a partir de arquivos de propriedades específicos para cada idioma. Por exemplo, o arquivo com todas as *strings* utilizadas na interface da ferramenta *ObasCId-Tool* para o idioma português é denominado "Messages_pt_BR.properties". Analogamente, o arquivo com as *strings* para o idioma inglês denomina-se "Messages_en_US.properties".

O conteúdo desses arquivos segue o formato: `<chave> = <valor>`, onde `<chave>` refere-se a uma *string* única do arquivo de propriedades, que servirá para permitir o acesso a uma determinada *string* internacionalizada; essa chave deve ser a mesma em todos os arquivos de internacionalização. `<valor>`, por sua vez, refere-se à própria palavra que será internacionalizada de acordo com o idioma escolhido pelo usuário; assim, seu conteúdo varia de um arquivo de internacionalização para outro. Por exemplo, dada a chave

[7] O código-fonte da ferramenta encontra-se disponível em: https://gitlab.com/advanse/obascid-tool

[8] https://javaserverfaces.java.net/

[9] http://getbootstrap.com/

[10] https://lucene.apache.org/core/

"*word_yes*", em um arquivo de internacionalização para o idioma inglês teríamos a entrada "*word_yes = Yes*" e para um arquivo para o português, "*word_yes = Sim*".

Sendo assim, caso necessário, o usuário poderá realizar o download do código fonte da ferramenta *ObasCId-Tool* e criar uma versão da ferramenta em seu idioma, bastando que ele elabore um arquivo de internacionalização.

Conforme comentado na Seção 5.6.2, a ferramenta *ObasCId-Tool* carrega automaticamente um conjunto de *stopwords* para cada novo pesquisador cadastrado. Esse conjunto de *stopwords* inclui palavras dos idiomas inglês e português. Para estendê-lo, incluindo, alterando ou removendo alguma *stopword* previamente cadastrada, basta que o usuário faça o download da ferramenta *ObasCId-Tool* e altere o valor da entrada "*stopwords*" nos arquivos de internacionalização da ferramenta. Para criar um conjunto de *stopwords* específico para outro idioma, basta que o usuário crie um novo arquivo de internacionalização e especifique um novo valor traduzido para seu idioma para a chave "*stopwords*".

Uma última opção de extensão da ferramenta *ObasCId-Tool* consiste em alterar o endereço do formulário de avaliação da ferramenta, que pode ser acessado pelo pesquisador por meio do item de *menu* da Figura 5.5 (C). Esse formulário é um questionário eletrônico desenvolvido no Google Forms[11], no qual são feitas algumas afirmações sobre a utilidade e facilidade de uso da ferramenta *ObasCId-Tool*. O questionário disponível na versão oficial da ferramenta[12] foi desenvolvido com base no modelo TAM (*Technology Acceptance Model*) (Davis, 1993) e é explicado no Capítulo 6 desta tese. Caso o usuário instale sua própria versão da ferramenta e queria torná-la pública, bem como alterar o formulário de avaliação da ferramenta, basta que ele altere o valor da chave "evaluation_form" nos arquivos de internacionalização. Como o *link* de acesso ao formulário de avaliação está nos arquivos de internacionalização, é possível elaborar e oferecer aos usuários da ferramenta formulários específicos em sua língua nativa.

5.11 Considerações Finais

No Quadro 5.6 estão descritas as ferramentas propostas para automatização das atividades das abordagens para EROA descritas no Capítulo 2 (Parreira Júnior e Penteado, 2013). Nota-se que a abordagem mais completa em termos de apoio computacional é a *EA*-

[11] https://www.google.com/forms/about/

[12] goo.gl/OEi18U

Miner (Chitchyan *et al*., 2005; Sampaio *et al*., 2005), pois todas as suas atividades são automatizadas em partes ou por completo. A atividade de composição de interesses dessa abordagem é totalmente automatizada pela ferramenta ARCADE. O usuário precisa apenas selecionar os interesses a serem compostos e toda regra de composição é gerada automaticamente. ARCADE automatiza também algumas atividades das abordagens *MDSoC* (Moreira *et al*., 2005) e EROA/Arcade (Rashid *et al*., 2003; Rashid *et al*., 2002).

Nota-se ainda que as atividades melhor contempladas com recursos computacionais são “Representação” e “Composição de Interesses”. Dessa forma, as atividades para EROA que exigem maior atenção da comunidade científica para confecção de apoios computacionais são “Identificação e Classificação de Interesses” e “Detecção e Análise de Conflitos”.

Quadro 5.6. Abordagens para EROA e ferramentas de apoio (Parreira Júnior e Penteado, 2013).

Abordagem	Atividade	Apoio Computacional
MDSoC	Representação e Composição de Interesses e Análise e Resolução de Conflitos	ARCADE
Theme	Representação de Interesses	*Plug-in* Eclipse Theme/UML
EA-Miner	Identificação e Classificação de Interesses	*EA-Miner*, WMATRIX, RATSY (*Requirement Analysis Tool with Synthesis*) e KWIP (*Key Word In Phrase*)
	Representação e Composição de Interesses e Análise e Resolução de Conflitos	ARCADE
EROA/XML	Representação e Composição de Interesses	APOR (*AsPect-Oriented Requirements tool*)
EROA/Arcade	Representação e Composição de Interesses e Análise e Resolução de Conflitos	ARCADE

Além de se conhecer os apoios computacionais oferecidos pelas abordagens, faz-se necessário conhecer também a situação desses apoios com relação aos seguintes aspectos (Quadro 5.7): (i) *disponibilidade*: visa a conhecer e divulgar os possíveis *links* para download do apoio computacional proposto, bem como do seu código-fonte, caso seja um software livre; (ii) *suporte e atualização*: visa a relatar à comunidade científica a data da última atualização do apoio computacional apresentado; e (iii) *documentação disponível*: visa a descrever os tipos de documentação oferecidos pelo apoio computacional, tais como manuais de uso, guias para desenvolvedores, entre outros, e onde eles podem ser encontrados. Esses aspectos são importantes, uma vez que versões obsoletas e/ou com documentação falha podem prejudicar o uso e a evolução dos apoios computacionais, bem como das abordagens que os utilizam.

Observa-se pelo Quadro 5.7 que a maioria dos apoios computacionais não está disponível, o que é um problema, pois dificulta a condução de pesquisas sobre as abordagens que fazem uso destes apoios computacionais. Muitas vezes esses apoios

computacionais existem e sua obtenção ocorre por meio de trocas de mensagens privativas (via *email*) com os desenvolvedores das ferramentas.

Quadro 5.7. Disponibilidade e situação das ferramentas computacionais propostas para EROA.

Apoio Computacional	*Download*	Licença de Uso	Documentação Disponível	
			Uso	Desenvolvimento
Plug-in Eclipse Theme/UML	Executável: https://www.scss.tcd.ie/~otooleea/lero/plugins/ThemeUMLPlugin_1.0.7.jar Código: https://www.scss.tcd.ie/~otooleea/lero/ThemeUMLPluginSrc.zip	*GNU Free Documentation License, Version 1.2*	*Manual, disponível em: https://www.scss.tcd.ie/~otooleea/lero/UserManual.pdf (última atualização: Fev/2009)*	*Guia para desenvolvedores, disponível em: https://www.scss.tcd.ie/~otooleea/lero/handbook_dev.pdf (última atualização: Fev/2009)*
EA-Miner	Indisponível	Indisponível	Indisponível	Indisponível
WMATRIX	Execução online apenas: http://ucrel.lancs.ac.uk/wmatrix3.html Código: indisponível	Proprietária	Manual disponível em: http://ucrel.lancs.ac.uk/wmatrix/ (última atualização: Fev/2010)	Indisponível
RATSY	Executável e código fonte: http://rat.fbk.eu/ratsy/index.php/Main/Download	GNU LESSER GENERAL PUBLIC LICENSE Version 2.1, February 1999 (LGPL)	Manual do usuário disponível em: http://rat.fbk.eu/ratsy/uploads/Main/RatsyManual_v2.1.pdf (última atualização: 2010)	Manual do usuário disponível em: http://rat.fbk.eu/ratsy/uploads/Main/RatsyManual_v2.1.pdf (última atualização: 2010)
KWIP	Indisponível	Indisponível	Indisponível	Indisponível
ARCADE	Indisponível	Indisponível	Indisponível	Indisponível
APOR	Indisponível	Indisponível	Indisponível	Indisponível

A documentação dos apoios computacionais disponíveis, isto é, o *plug-in Eclipse Theme/UML*, *WMATRIX* e *RATSY*, é obsoleta, o que pode ser um indício de que esses apoios computacionais foram descontinuados. No caso do plug-in *Eclipse Theme/UML*, por se tratar de um apoio computacional que depende de outra plataforma para ser executado, isto é, da *IDE Eclipse*, o mesmo funciona apenas em versões mais antigas desta IDE. Essas situações também dificultam a condução de pesquisas relacionadas a essas abordagens. Um ponto positivo desse *plug-in* é o fato de ele ser software livre, permitindo assim, que outros pesquisadores/desenvolvedores possam adaptar e/ou atualizar o software para suas necessidades, bem como para as novas versões das plataformas de desenvolvimento.

O último ponto a ser destacado é que não foram encontradas ferramentas disponíveis e funcionais para identificação e classificação de interesses de software a partir

de documentos de requisitos, o que prejudicou a avaliação das abordagens analisadas no Capítulo 2 com relação à ferramenta *ObasCId-Tool*.

Com base no que foi apresentado anteriormente, *ObasCId-Tool* consiste em uma importante iniciativa no contexto da identificação e classificação de interesses. De acordo com o Quadro 5.7, apenas *EA-Miner* possui um módulo de apoio à identificação de interesses, contudo a mesma não estava disponível para download e uso. Além disso, ela baseia-se na ferramenta *WMATRIX* (2015), que é responsável pela execução de rotinas de processamento de linguagem natural para textos em inglês apenas.

ObasCId-Tool é voltada para identificação de interesses com base em palavras-chave e no relacionamento de dependência existentes entre requisitos de software e interesses. Para isso, catálogos de interesses, confeccionados com base nos conceitos da ontologia *O4C*, são utilizados como base de conhecimento. Isso torna a ferramenta *ObasCId-Tool* independente do idioma utilizado pelo pesquisador, bastando que o mesmo tenha os artefatos necessários para aplicação da ferramenta escritos em seu idioma, tais como o catálogo de interesses e o documento de requisitos. Além disso, conforme comentado na Seção 5.10, toda interface da ferramenta *ObasCId-Tool* é internacionalizada para os idiomas inglês e português (do Brasil) e ela oferece suporte facilitado para extensão dessa internacionalização para outros idiomas.

O Capítulo 6 desta tese apresenta um conjunto de dois estudos experimentais realizados com o intuito de avaliar os produtos deste doutorado, a saber, a abordagem *ObasCId* e a ferramenta *ObasCId-Tool*, ambos baseados na ontologia *O4C*, quanto à sua efetividade e eficiência no contexto da EROA.

Capítulo 6

ESTUDOS EXPERIMENTAIS

6.1 Considerações Iniciais

O sucesso do desenvolvimento de um software depende da interação entre diversas variáveis, como ambiente de trabalho, experiência pessoal, utilização de processos e procedimentos que visem à melhoria da qualidade do produto final e apoio ferramental (Wohlin *et al*., 2012; Mafra e Travassos, 2006). Kitchenham *et al*. (2004) complementam essa afirmação, dizendo que, a partir desse cenário com diversas variáveis, uma questão importante é "como identificar, isolar e avaliar a contribuição individual da aplicação de uma determinada tecnologia (processo, técnica, ferramenta, entre outros) para a qualidade final do produto de software?".

A utilização de experimentação em Engenharia de Software pode atender satisfatoriamente a essa necessidade, uma vez que ela fornece mecanismos adequados para a identificação e o entendimento do relacionamento entre as diferentes variáveis envolvidas em um determinado contexto (Mafra e Travassos, 2006). Neste sentido, alguns tipos de investigação podem ser conduzidos, tais como *surveys*, estudos de caso e experimentos controlados.

Contudo, para que a avaliação de qualquer tecnologia seja efetiva, torna-se necessário dar enfoque aos objetivos a serem alcançados com tais medições (Basili e Rombach, 1994). Soligen e Berghout (1999) afirmam que pode ser interessante utilizar abordagens para guiar a definição de modelos de avaliação. Nesse sentido, uma abordagem comumente utilizada é a GQM (*Goal-Question-Metric*) (Basili e Rombach, 1994).

Na Seção 6.2 deste capítulo, apresentam-se os modelos de avaliação criados para a abordagem proposta neste trabalho, *ObasCId*, bem como para a ferramenta computacional *ObasCId-Tool*, com base na abordagem GQM. Nas Seções 6.3 e 6.4 estão os estudos

experimentais necessários para se atingir os objetos de avaliação definidos na Seção 6.2. Para cada estudo, apresenta-se o planejamento, a execução, os resultados obtidos com o mesmo, bem como as ameaças à validade do estudo, de acordo com o roteiro de experimentação para engenharia de software proposto por Wohlin *et al*. (2012). Por fim, a Seção 6.5 apresenta as considerações finais deste capítulo.

6.2 Modelo GQM dos Estudos Experimentais

A abordagem GQM baseia-se na premissa de que para avaliar qualquer tecnologia, primeiramente, os próprios objetivos da avaliação devem ser conhecidos. GQM é dividida em três níveis (Basili e Rombach, 1994):

- **Conceitual**: no qual são definidos os objetivos da avaliação. O objetivo é definido para um objeto, que pode ser um produto, um processo, um serviço, entre outros. No contexto deste trabalho, os objetos são a abordagem *ObasCId* e a ferramenta *ObasCId-Tool*;
- **Operacional**: no qual são definidas questões que caracterizam um caminho para se alcançar um determinado objetivo de avaliação; e
- **Quantitativo**: no qual são definidas as métricas que definem um conjunto de dados (objetivos ou subjetivos), que quando interpretados, permitirão ao pesquisador responder às questões estabelecidas no nível operacional.

6.2.1 Nível conceitual: definição dos objetivos de avaliação

Para este modelo de avaliação, foram criados três objetivos, os quais são apresentados no Quadro 6.1, no Quadro 6.2 e no Quadro 6.3.

Quadro 6.1. Avaliação da abordagem *ObasCId* quanto à sua efetividade e eficiência.

Objetivo 1 (O1)
Analisar a: o uso da abordagem *ObasCId* para identificação e classificação de interesses de software **Com o propósito de**: avaliar **Com respeito à**: efetividade (cobertura e precisão) e eficiência (tempo) **Do ponto de vista de**: engenheiros de software **No contexto de**: um grupo de alunos de graduação e pós-graduação em Ciência da Computação

O primeiro objetivo diz respeito à avaliação da efetividade e da eficiência da abordagem *ObasCId*; o segundo e o terceiro estão relacionados, respetivamente, à

avaliação da facilidade de uso e da utilidade percebidas pelos usuários da ferramenta *ObasCId-Tool*.

Quadro 6.2. Avaliação da ferramenta *ObasCId-Tool* quanto à facilidade de uso percebida por seus usuários.

Objetivo 2 (O2)
Analisar o: o uso da ferramenta *ObasCId-Tool* para identificação e classificação de interesses de software **Com o propósito de**: avaliar **Com respeito à**: facilidade de uso percebida **Do ponto de vista de**: engenheiros de software **No contexto de**: um grupo de alunos de graduação e pós-graduação em Ciência da Computação.

Quadro 6.3. Avaliação da ferramenta *ObasCId-Tool* quanto à utilidade percebida por seus usuários.

Objetivo 3 (O3)
Analisar o: o uso da ferramenta *ObasCId-Tool* para identificação e classificação de interesses de software **Com o propósito de**: avaliar **Com respeito à**: utilidade percebida **Do ponto de vista de**: engenheiros de software **No contexto de**: um grupo de alunos de graduação e pós-graduação em Ciência da Computação.

6.2.2 Nível operacional: definição das questões

A partir dos objetivos apresentados na Seção 6.2.1, algumas questões foram elencadas a fim de que, uma vez respondidas, possam indicar se os objetivos de avaliação foram atingidos ou não. As questões definidas para o objetivo O1 são apresentadas no Quadro 6.4. Esse quadro apresenta: (i) a sigla da questão na primeira coluna – esta sigla será utilizada mais a frente neste capítulo; e (ii) a descrição da questão na segunda coluna.

Quadro 6.4. Questões para o objetivo 1.

Sigla	Descrição
Q1O1	Quão efetiva (em termos de cobertura e precisão) é a identificação e classificação de interesses de software com o apoio da abordagem *ObasCId*?
Q2O1	Quão eficiente (em termos de tempo de execução) é a identificação e classificação de interesses de software com o apoio da abordagem *ObasCId*?

Para elencar as questões para os objetivos O2 e O3, respectivamente correspondentes à facilidade de uso e à utilidade percebidas pelos usuários da ferramenta *ObasCId-Tool*, o modelo de aceitação de tecnologia TAM (*Technology Acceptance Model*) (Davis, 1993) foi utilizado. Esse modelo possui como objetivo explicar o comportamento das pessoas em relação à aceitação de uma tecnologia e tem sido utilizado em estudos recentes (Hernandes, 2014) para avaliação de produtos de software. O modelo TAM define dois constructos básicos (Davis, 1993): (i) *utilidade percebida*, que mede o quanto uma pessoa

acredita que usar determinada tecnologia aumenta sua produtividade; e (ii) *facilidade de uso percebida*, que mede o quanto uma pessoa acredita que o uso de determinada tecnologia é fácil. Sugere-se ainda a criação de questionários, aos quais são atribuídas afirmações relacionadas à facilidade de uso e à utilidade percebidas pelos usuários da tecnologia em análise. Para cada afirmação, o respondente deve escolher uma dentre as seguintes opções "Discordo totalmente", "Discordo em grande parte", "Discordo parcialmente", "Neutro", "Concordo parcialmente", "Concordo em grande parte" e Concordo totalmente", conforme sua opinião sobre essa afirmação.

As questões referentes aos objetivos de avaliação O2 e O3, que correspondem aos constructos *facilidade de uso percebida* e *utilidade percebida* são apresentadas, respectivamente, no Quadro 6.5 e no Quadro 6.6.

Quadro 6.5. Questões para o objetivo O2.

Sigla	Descrição
Q1O2	Eu gostei de trabalhar com a *ObasCId-Tool*.
Q2O2	O acesso à *ObasCId-Tool* é simples.
Q3O2	Usar a *ObasCId-Tool* é uma boa ideia.
Q4O2	Na *ObasCId-Tool* eu sempre sei onde estou e como chegar aonde quero.
Q5O2	Os recursos de manipulação da *ObasCId-Tool* estão claros e fáceis de achar.
Q6O2	Minha interação com a *ObasCId-Tool* é clara e compreensível.
Q7O2	Na *ObasCId-Tool*, é fácil encontrar a informação que desejo.
Q8O2	A *ObasCId-Tool* possui visual/interface atraente.
Q9O2	Quão efetivo (em termos da proporção de atividades concluídas com sucesso) é o gerenciamento dos recursos necessários para identificação e classificação de interesses de software com apoio da ferramenta *ObasCId-Tool*?

Quadro 6.6. Questões para o objetivo O3.

Sigla	Descrição
Q1O3	Utilizar a *ObasCId-Tool* é importante e adiciona valor ao meu trabalho.
Q2O3	A *ObasCId-Tool* é útil no processo de identificação e classificação de interesses de software.
Q3O3	Usar a *ObasCId-Tool* pode aumentar meu desempenho durante a identificação e classificação de interesses a partir de requisitos de software.
Q4O3	A *ObasCId-Tool* pode facilitar a realização do meu trabalho.
Q5O3	A *ObasCId-Tool* produz os resultados que espero de uma ferramenta de suporte à identificação e classificação de interesses de software.
Q6O3	Eu pretendo integrar a *ObasCId-Tool* à minha rotina de trabalho.
Q7O3	Eu recomendarei o uso da *ObasCId-Tool*.
Q8O3	Os conceitos da área de identificação e classificação de interesses a partir de requisitos de software foram abordados por completo na *ObasCId-Tool*.

Para o objetivo O2, correspondente à facilidade de uso percebida pelos usuários da ferramenta *ObasCId-Tool*, além das questões elencadas com base no modelo TAM, criou-se

ainda mais uma questão, referente ao gerenciamento dos recursos necessários para identificação e classificação de interesses de software nessa ferramenta (questão “Q9O2” do Quadro 6.5).

No contexto da ferramenta *ObasCId-Tool*, tais recursos são “catálogos de interesses de software”, “documentos de requisitos de software” e “unidades de identificação de interesses”, conforme discutido no Capítulo 5. Esse tipo de questão garante maior objetividade ao processo de avaliação da característica de facilidade de uso da ferramenta *ObasCId-Tool*. Contudo, o uso do modelo TAM ainda faz-se necessário, pois sua aplicação pode oferecer informações úteis sobre a ferramenta, que não seriam observadas somente com os resultados da questão comentada anteriormente.

6.2.3 Nível quantitativo: definição das métricas

Após definidas as questões para cada objetivo de avaliação, deve-se especificar um conjunto de métricas capazes de indicar valores que, quando interpretados, fornecerão as informações necessárias para que o avaliador responda a essas questões. No Quadro 6.7 e no Quadro 6.8 estão as métricas utilizadas para avaliação da efetividade e da eficiência da abordagem *ObasCId* (Objetivos O1 e O2).

Quadro 6.7. Métricas diretas para avaliação da abordagem proposta.

Sigla	Descrição
M1	Quantidade de interesses base ou transversais existentes no software em análise (obtidos por meio de um oráculo).
M1.1	Quantidade de interesses base existentes no software em análise (obtidos por meio de um oráculo).
M1.2	Quantidade de interesses transversais existentes no software em análise (obtidos por meio de um oráculo).
M2	Quantidade de interesses base ou transversais identificados com auxílio da abordagem em análise (inclui os falsos positivos).
M2.1	Quantidade de interesses base identificados com auxílio da abordagem em análise (inclui os falsos positivos).
M2.2	Quantidade de interesses transversais identificados com auxílio da abordagem em análise (inclui os falsos positivos).
M3	Quantidade de interesses base ou transversais corretamente identificados com auxílio da abordagem em análise (não inclui os falsos positivos).
M3.1	Quantidade de interesses base corretamente identificados com auxílio da abordagem em análise (não inclui os falsos positivos).
M3.2	Quantidade de interesses transversais corretamente identificados com auxílio da abordagem em análise (não inclui os falsos positivos).
M4	Tempo gasto (em minutos) para identificação e classificação dos interesses do software com auxílio da abordagem em análise.

Muitas dessas métricas são diretas (Quadro 6.7), ou seja, referem-se à contagem de elementos do produto de software, como quantidade de interesses base e de interesses

transversais. Outras métricas são indiretas, isto é, são obtidas a partir de expressões matemáticas simples envolvendo duas ou mais métricas diretas (Quadro 6.8).

Quadro 6.8. Métricas indiretas para avaliação da abordagem proposta.

Sigla	Descrição	Fórmula
M5	Cobertura global: porcentagem de interesses base ou transversais identificados corretamente em relação ao total de interesses existentes no software.	$\left(\frac{M2}{M1}\right)*100$
M6	Cobertura para interesses base: porcentagem de interesses base identificados corretamente em relação ao total de interesses base existentes no software.	$\left(\frac{M2.1}{M1.1}\right)*100$
M7	Cobertura para interesses transversais: porcentagem de interesses transversais identificados corretamente em relação ao total de interesses transversais existentes no software.	$\left(\frac{M2.2}{M1.2}\right)*100$
M8	Precisão global: porcentagem de interesses base ou transversais identificados corretamente em relação ao total de interesses base ou transversais identificados com auxílio da abordagem em análise.	$\left(\frac{M3}{M2}\right)*100$
M9	Precisão para interesses base: porcentagem de interesses base identificados corretamente em relação ao total de interesses base identificados com auxílio da abordagem em análise.	$\left(\frac{M3.1}{M2.1}\right)*100$
M10	Precisão para interesses transversais: porcentagem de interesses transversais identificados corretamente em relação ao total de interesses transversais identificados com auxílio da abordagem em análise.	$\left(\frac{M3.2}{M2.2}\right)*100$

As métricas Cobertura (*Recall*) e Precisão (*Precision*), descritas no Quadro 6.8, geralmente são utilizadas para medição da efetividade de produtos e processos em diversas áreas de pesquisa, tais como recuperação da informação e processamento de linguagem natural. Além disso, elas são amplamente utilizadas em trabalhos relacionados à identificação e classificação de interesses de software, tanto em nível de código (Kellens *et al*., 2007), quanto em nível de requisitos (Herrera *et al*., 2012; Sampaio *et al*., 2007).

Na Figura 6.1 sumarizam-se os relacionamentos entre as métricas descritas anteriormente e as questões especificadas para o objetivo de avaliação O1. Como pode ser visto nesta figura, cada questão está relacionada diretamente com as métricas que são efetivamente utilizadas para respondê-la. Algumas métricas, por sua vez, estão relacionadas a outras métricas, pois dependem delas para o cálculo do seu valor.

No Quadro 6.9 é apresentado o conjunto de métricas utilizadas para a avaliação da utilidade e da facilidade de uso percebidas pelos usuários da ferramenta *ObasCId-Tool* (objetivos O2 e O3). Analogamente ao que foi feito para o objetivo O1, a Figura 6.2 apresenta o relacionamento existente entre os objetivos O2 e O3 e suas respectivas questões e métricas.

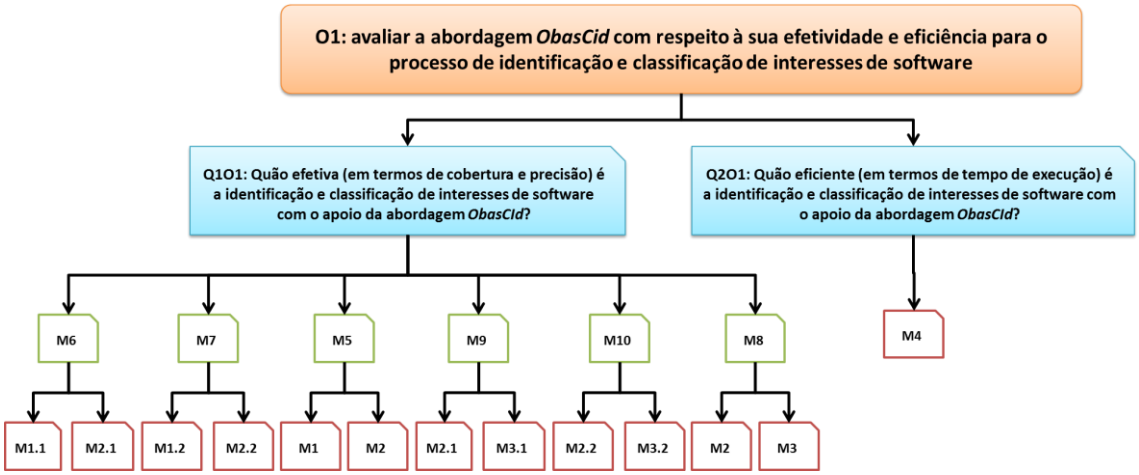


Figura 6.1. Relacionamento entre o objetivo O1 e suas respectivas questões e métricas.

Quadro 6.9. Métricas para avaliação da utilidade e facilidade de uso da ferramenta proposta.

Sigla	Descrição
M11	Porcentagem de usuários que escolheram a opção “Discordo totalmente”.
M12	Porcentagem de usuários que escolheram a opção “Discordo em grande parte”.
M13	Porcentagem de usuários que escolheram a opção “Discordo parcialmente”.
M14	Porcentagem de usuários que escolheram a opção “Neutro”.
M15	Porcentagem de usuários que escolheram a opção “Concordo parcialmente”.
M16	Porcentagem de usuários que escolheram a opção “Concordo em grande parte”.
M17	Porcentagem de usuários que escolheram a opção “Concordo totalmente”.
M18	Porcentagem de atividades concluídas com sucesso pelo usuário durante o gerenciamento de recursos para identificação e classificação de interesses na ferramenta *ObasCId-Tool*.

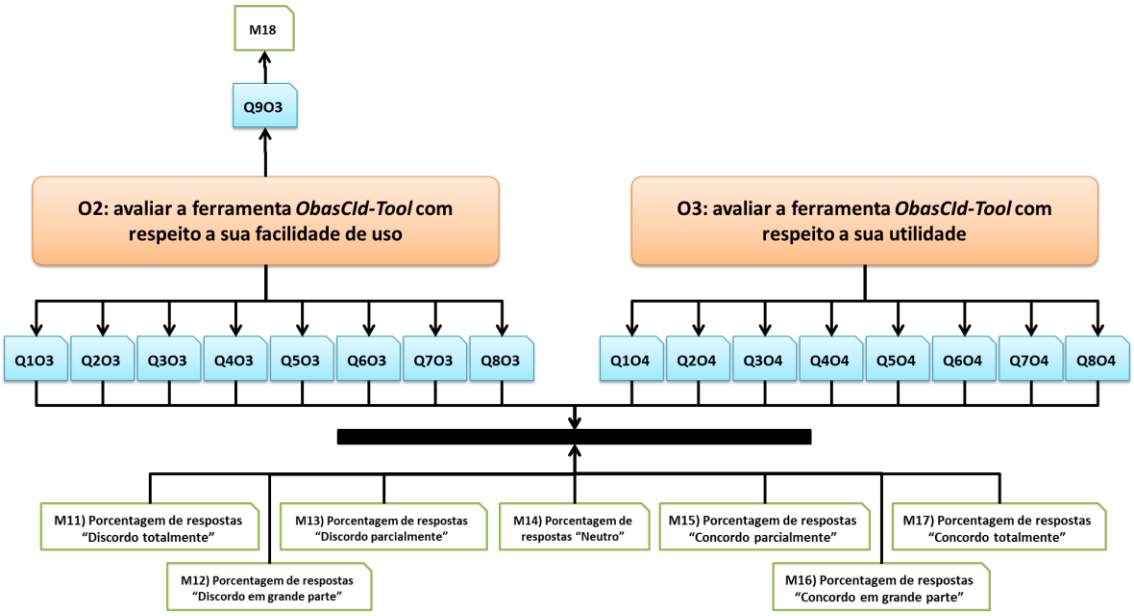


Figura 6.2. Relacionamento entre os objetivos O2 e O3 e suas respectivas questões e métricas.

Uma etapa importante do nível quantitativo do modelo GQM é especificar a interpretação das métricas de cada questão, pois é por meio dela que o avaliador poderá

estabelecer hipóteses, bem como extrair conclusões dos resultados obtidos após a execução de seu modelo avaliativo. Dessa forma, no Quadro 6.10 é apresentado como se deve interpretar o resultado das principais métricas elencadas para este modelo de avaliação.

Quadro 6.10. Interpretação das métricas do modelo de avaliação proposto.

Métrica	Interpretação
M5, M6, M7, M8, M9 e M10	Quanto maior o valor dessas métricas, mais efetiva é a abordagem para identificação e classificação de interesses de software.
M4	Quanto menor o valor dessa métrica, mais eficiente é a abordagem para identificação e classificação de interesses de software.
M18	Quanto maior o valor dessa métrica, mais efetiva é a ferramenta em análise, com relação ao gerenciamento dos recursos necessários para identificação e classificação de interesses de software. Mais especificamente, caso M18 seja menor ou igual 75%, então a ferramenta não pode ser considerada efetiva para o gerenciamento de recursos para identificação e classificação de interesses. Caso contrário, ou seja, se M18 > 75%, então a ferramenta pode ser considerada efetiva para o gerenciamento de tais recursos. O valor de 75% como *threshold* foi escolhido de forma *ad-hoc* por pesquisadores com experiência de uso sobre a ferramenta *ObasCId-Tool*.
M11, M12, M13, M14, M15, M16 e M17	Para o construto utilidade percebida, se M11 + M12 + M13 + M14 ≥ M15 + M16 + M17, isto é, se a porcentagem de opiniões negativas/neutra a respeito desse constructo for maior ou igual à porcentagem de opiniões positivas, então a característica representada por esse constructo deve ser revista na ferramenta em análise. Caso M11 + M12 + M13 + M14 < M15 + M16 + M17, então a ferramenta em análise pode ser considerada satisfatória com relação à característica representada pelo constructo. No caso do construto facilidade de uso percebida, vale o que foi exposto acima para o constructo utilidade percebida; além disso, deve-se levar em consideração também a efetividade proporcionada pela ferramenta em análise (métrica M18). Ou seja, a facilidade de uso percebida pelos usuários da ferramenta pode ser considerada satisfatória, caso (M11 + M12 + M13 + M14 < M15 + M16 + M17) e (M18 > 75%). Caso contrário, isto é, se (M11 + M12 + M13 + M14 ≥ M15 + M16 + M17) ou (M18 ≤ 75%), então a facilidade de uso da ferramenta em análise é insatisfatória.

6.3 Estudo Experimental I – *ObasCId* vs. *Theme/Doc*

O objetivo deste estudo experimental foi prover dados que, quando tratados e interpretados, possam permitir aos pesquisadores responder às duas questões apresentadas na Seção 6.2.2 para o objetivo de avaliação O1. Para isso, um grupo de participantes foi orientado a identificar e classificar os interesses de um software real utilizando como apoio as abordagens *ObasCId* e *Theme/Doc*. A abordagem *Theme/Doc* foi escolhida por ser uma abordagem robusta e que tem sido avaliada em estudos experimentais recentes (Herrera *et al*., 2012).

6.3.1 Planejamento

O planejamento deste experimento foi realizado de acordo com o modelo proposto por Wohlin *et al*. (2012), que envolve as fases descritas a seguir.

1. Seleção do contexto. O estudo foi realizado com quatorze alunos de graduação e pós-graduação em Ciência da Computação, no contexto da disciplina Engenharia de Software. Cabe ressaltar que esse estudo não teve impacto na nota dos alunos, para que isso não influenciasse nos resultados do experimento.

Os softwares cujos documentos de requisitos foram utilizados referem-se ao sistema de informação para registro de reclamações na área da saúde, *Health Watcher* (2015), e ao sistema de informação para locação de DVDs apresentado em (Viana, 2009), que será referenciado daqui em diante como *LocaDVD*.

Health Watcher foi escolhido por apresentar um documento de requisitos favorável à identificação e classificação de interesses de software, uma vez que vários interesses transversais estão presentes neste software, tais como “Segurança”, “Persistência”, entre outros. Além disso, os interesses do software *Health Watcher* já foram devidamente identificados e catalogados por especialistas em EROA (Health Watcher, 2015), servindo como oráculo para verificação das respostas dadas pelos participantes deste estudo experimental. Já o software *LocaDVD* foi escolhido por se tratar de um sistema de gestão de recursos de negócio, apropriado para ser utilizado com catálogos de interesses criados a partir da linguagem de padrões GRN (Braga, 2002), conforme apresentado na Seção 4.3 desta tese.

2. Formulação de hipóteses. A partir das questões e das métricas levantadas nas Seções 6.2.2 e 6.2.3, seis hipóteses foram elaboradas, duas relacionadas à cobertura, duas à precisão e duas ao tempo de execução das abordagens *ObasCId* e *Theme/Doc* (Quadro 6.11). É importante salientar que, devido à pequena quantidade de interesses de um mesmo tipo existentes nestes softwares, isto é, funcional ou não funcional, apenas hipóteses para as métricas globais de cobertura e precisão foram elaboradas. Contudo, no caso do experimento com o software *LocaDVD*, os valores das métricas cobertura e precisão específicas para cada tipo de interesse são apresentados e discutidos de forma descritiva.

3. Seleção de variáveis e seleção dos participantes. Variáveis independentes são aquelas manipuladas e controladas durante o estudo. Neste estudo, a variável independente consiste nas abordagens para identificação e classificação de interesses *ObasCId* e *Theme/Doc*. As variáveis dependentes são aquelas sob análise e cujas variações, com base nas mudanças feitas nas variáveis independentes, devem ser observadas. Neste experimento, a cobertura (métricas M5, M6 e M7), a precisão (métricas M8, M9 e M10) e o tempo (M4) são considerados como variáveis dependentes.

Os participantes do experimento foram selecionados por meio de amostragem não probabilística por conveniência. Dos quatorze alunos participantes do experimento, sete eram de graduação e sete de pós-graduação. De acordo com o questionário de caracterização de perfil disponível no Apêndice B, todos os participantes afirmaram não terem trabalhado com EROA, nem com identificação de interesses de software em nível de código. Quando perguntados sobre seu nível de conhecimento sobre Engenharia de Requisitos, todos os participantes afirmaram estar em um nível intermediário, cujo conhecimento nesse assunto se deu por meio de disciplinas em Engenharia de Software.

Quadro 6.11. Hipóteses para o Estudo Experimental I.

Hipóteses referentes à cobertura proporcionada pelas abordagens	
H_{0M5}	Não há diferença com relação à cobertura global média proporcionada pelas abordagens *ObasCId* e *Theme/*Doc, isto é, H_{0M5}: $M5_{ObasCId} = M5_{Theme/Doc}$
H_{1M5}	Há diferença com relação à cobertura global média proporcionada pelas abordagens *ObasCId* e *Theme/*Doc, isto é, H_{1M5}: $M5_{ObasCId} \neq M5_{Theme/Doc}$
Hipóteses referentes à precisão proporcionada pelas abordagens	
H_{0M8}	Não há diferença com relação à precisão global média proporcionada pelas abordagens *ObasCId* e *Theme/*Doc, isto é, H_{0M8}: $M8_{ObasCId} = M8_{Theme/Doc}$
H_{1M8}	Há diferença com relação à precisão global média proporcionada pelas abordagens *ObasCId* e *Theme/*Doc, isto é, H_{1M8}: $M8_{ObasCId} \neq M8_{Theme/Doc}$
Hipóteses referentes ao tempo de execução das abordagens	
H_{0M4}	Não há diferença com relação ao tempo médio para execução das abordagens *ObasCId* e *Theme/*Doc, isto é, H_{0M4}: $M4_{ObasCId} = M4_{Theme/Doc}$
H_{1M4}	Há diferença com relação ao tempo médio para execução das abordagens *ObasCId* e *Theme/*Doc, isto é, H_{1M4}: $M4_{ObasCId} \neq M4_{Theme/Doc}$

4. Projeto e execução do experimento realizado. A distribuição dos participantes nos grupos foi realizada com o intuito de formar dois grupos homogêneos, com respeito ao nível de experiência dos participantes, e com sete participantes em cada. A experiência dos participantes foi verificada pela aplicação de um questionário de caracterização de perfil, que considerou os conhecimentos dos mesmos sobre EROA. Além disso, o experimento foi planejado em fases (treinamento e execução) para minimizar ainda mais o efeito do conhecimento dos participantes sobre as variáveis dependentes.

Antes do início do experimento, foi realizado um treinamento que teve como objetivo homogeneizar o conhecimento dos participantes sobre EROA e sobre as abordagens *ObasCId* e *Theme/Doc*. Durante o treinamento, não foi informado aos participantes qual das abordagens foi desenvolvida pelo autor deste trabalho. Também houve a preocupação de que o treinamento não beneficiasse uma das abordagens em detrimento da outra.

A execução do experimento ocorreu em duas fases. Na primeira fase, os participantes deveriam identificar os interesses não funcionais presentes no documento de requisitos do software *Health Watcher* e classificá-los como transversais ou não. Para isso, o Grupo 1 utilizou a abordagem *Theme/Doc* e o Grupo 2 a *ObasCId*. Na segunda fase, os

participantes deveriam identificar os interesses funcionais e não funcionais do software *LocaDVD* e classificá-los como transversais ou não. Para isso, o Grupo 1 utilizou a abordagem *ObasCId* e o Grupo 2, a *Theme/Doc*. A configuração do projeto do estudo experimental I pode ser visualizada no Quadro 6.12. Com essa configuração, minimiza-se a interferência de variáveis que não são de interesse desse estudo, tais como a experiência dos participantes e o tipo de software utilizado, sobre as variáveis dependentes.

Quadro 6.12. Projeto do estudo experimental I.

Fases do estudo	Grupo 1	Grupo 2
Treinamento	*ObasCId* e *Theme/Doc*	*ObasCId* e *Theme/Doc*
Fase 1	*Theme/Doc + Health Watcher*	*ObasCId + Health Watcher*
Fase 2	*ObasCId + LocaDVD*	*Theme/Doc + LocaDVD*

O trecho do documento de requisitos do *Health Watcher* disponibilizado aos participantes apresentava seis interesses não funcionais, todos classificados como transversais no oráculo utilizado como referência (Health Watcher, 2015): “Segurança”, “Concorrência”, “Usabilidade”, “Performance”, “Disponibilidade” e “Persistência”. Quanto ao documento de requisitos do software *LocaDVD*, o mesmo apresentava quatro interesses funcionais (“Pagamento”, “Transação”, “Recurso” e “Destino”) e dois não funcionais (“*Logging*” e “Persistência”), sendo “Transação”, “*Logging*” e “Persistência” os interesses transversais desse software. Para contagem dos acertos dos participantes, bem como para calcular os valores das métricas cobertura e precisão, considerou-se a quantidade de interesses identificados e classificados corretamente pelos participantes do experimento.

Na primeira fase do experimento, além do documento de requisitos do software *Health Watcher*, os participantes do Grupo 2 receberam também um catálogo para interesses não funcionais de software, representado em um diagrama de classes UML, igual àquele apresentado no Capítulo 4. Tal catálogo foi criado a partir dos catálogos de requisitos não funcionais propostos por Cysneiro (2015), Chung e Leite (2000) e Boehm e In (1996). É importante salientar que, como não havia catálogos específicos para interesses funcionais referentes ao domínio de reclamações na área da saúde, decidiu-se fornecer os interesses funcionais do software *Health Watcher* previamente identificados para ambos os grupos. Quanto aos participantes do Grupo 1, eles tiveram acesso aos catálogos de requisitos não funcionais utilizados para confecção do catálogo de interesses utilizado pelo Grupo 2.

Na segunda fase, além do documento de requisitos do software *LocaDVD*, os participantes do Grupo 1 receberam também um catálogo para interesses não funcionais, similar ao utilizado na primeira fase do experimento, porém com a inclusão de novos interesses. Tais interesses são do tipo funcional e correspondem a alguns padrões da linguagem GRN, conforme apresentado na Seção 4.3.

6.3.2 Análise dos Resultados

Na Tabela 6.1 são apresentados os resultados obtidos para os dois grupos de participantes, com relação ao software *Health Watcher*. A primeira parte desta tabela (colunas 1 a 4) apresenta os resultados do Grupo 1, que utilizou a abordagem *Theme/Doc*. Já na segunda parte (colunas 5 a 8), os resultados referem-se aos participantes do Grupo 2, que utilizaram a abordagem *ObasCId*.

Tabela 6.1. Resultados do estudo experimental I para o software *Health Watcher*.

Abordagem *Theme/Doc* (Grupo 1)				**Abordagem *ObasCId* (Grupo 2)**			
Partic.	**Cobertura Global (M5)**	**Precisão Global (M8)**	**Tempo (min) – M4**	**Partic.**	**Cobertura Global (M5)**	**Precisão Global (M8)**	**Tempo (min) – M4**
P1	42,85	75,00	43	P8	71,42	71,00	62
P2	42,85	100,00	48	P9	85,71	100,00	39
P3	42,85	100,00	49	P10	85,71	100,00	54
P4	28,57	66,00	48	P11	71,42	100,00	37
P5	57,14	80,00	36	P12	57,14	75,00	43
P6	42,85	100,00	31	P13	71,42	80,00	42
P7	28,57	100,00	34	P14	71,42	100,00	42
Média	**40,81**	**88,71**	**41,28**	**Média**	**73,46**	**89,42**	**45,57**

Com relação à precisão, nota-se na Tabela 6.1 que não há diferença significativa entre as duas abordagens. Além disso, pelo alto valor dessa métrica, pode-se dizer que não houve grande incidência de falsos positivos durante a identificação dos interesses do software.

Ainda a partir da Tabela 6.1, é possível notar que, quanto ao tempo para aplicação das abordagens, a abordagem *ObasCId* consumiu mais tempo. Foram, em média, 45 (quarenta e cinco) minutos gastos pelos participantes que utilizaram *ObasCId* contra 41 (quarenta e um) minutos gastos pelos participantes que utilizaram a abordagem *Theme/Doc*. Isso faz sentido, uma vez que os participantes que utilizaram a abordagem *ObasCId* tinham mais um artefato para consultar, isto é, o catálogo de interesses de software, bem como algumas atividades novas a serem realizadas, que não são contempladas na abordagem *Theme/Doc*. Contudo, nota-se que a diferença de tempo não é significativa. Entende-se que, apesar de os participantes que utilizaram a abordagem *ObasCId* terem que realizar tarefas extras, o trabalho de identificação e classificação de interesses fica mais focado e mais direcionado com o uso do catálogo de interesses e do processo proposto, minimizando assim, o impacto sobre o tempo de aplicação da abordagem.

Quanto à cobertura, os participantes que utilizaram a abordagem *ObasCId* (Grupo 2) obtiveram resultados mais promissores, apresentando uma cobertura média de aproximadamente 74%, contra 41% daqueles que utilizaram a abordagem *Theme/Doc* (Grupo 1). Para complementar a discussão sobre a métrica cobertura, na Tabela 6.2 são

apresentados: (i) a listagem de interesses existentes na aplicação - primeira coluna; (ii) os interesses identificados por cada participante que utilizou a abordagem *Theme/Doc* - da segunda até a oitava colunas; (iii) a porcentagem de participantes que identificaram cada interesse - nona coluna; e (iv) as mesmas informações apresentadas anteriormente para a abordagem *ObasCId* - da décima até a décima sétima coluna.

Tabela 6.2. Interesses identificados no software *Health Watcher*.

Interesses	**Participantes *Theme/Doc* (Grupo 1)**							**%**	**Participantes *ObasCId* (Grupo 2)**							**%**
	1	**2**	**3**	**4**	**5**	**6**	**7**		**8**	**9**	**10**	**11**	**12**	**13**	**14**	
Persistência	X	X						28	X	X	X	X	X			71
Segurança	X	X	X	X	X	X	X	100	X	X	X	X	X	X	X	100
Concorrência		X						14		X	X	X		X	X	71
Usabilidade			X	X	X	X		57	X	X	X	X		X	X	85
Desempenho					X	X	X	43	X	X	X			X	X	71
Disponibilidade	X		X		X			43	X	X	X	X	X			71
Média								**47,5**	**Média**							**75,83**
# Falsos positivos	1	0	0	1	1	0	0	-	2	0	0	0	1	1	0	-

Por meio dessa tabela, é possível notar que para cada interesse, a porcentagem de participantes da abordagem *ObasCId* que o identificou é sempre maior ou igual à porcentagem de participantes da abordagem *Theme/Doc* que o identificou. O interesse mais identificado pelos participantes das duas abordagens é o interesse “Segurança” e os menos identificados são “Persistência” e “Concorrência”. “Segurança” é um interesse que geralmente possui requisitos bem claros a seu respeito, já no caso da “Persistência”, por exemplo, em geral não há requisitos escritos especificamente para esse interesse, o que pode ter comprometido a identificação do mesmo. Quanto ao interesse “Concorrência”, o baixo percentual de participantes do Grupo 1 (*Theme/Doc*) que o identificaram pode ser justificado por se tratar de um interesse não tão conhecido pelos participantes, quanto os demais. Contudo, as informações oferecidas no catálogo de interesses podem ter influenciado positivamente os resultados, fazendo com que a porcentagem de participantes do Grupo 2 (*ObasCId*) que identificou o interesse “Concorrência” fosse maior.

Quanto aos falsos positivos, tanto na abordagem *Theme/Doc* como na *ObasCId*, a maioria deles apareceu em razão das ambiguidades existentes nas palavras dos catálogos fornecidos aos participantes do experimento e da falta de experiência dos participantes sobre alguns tipos de interesses não funcionais. Por exemplo, a palavra-chave “ao mesmo tempo”, que é um sinônimo de “simultaneamente”, fez com que fosse identificado o interesse de “Distribuição”, em invés de “Concorrência”.

Os mesmos tipos de informações apresentadas na Tabela 6.1 e Tabela 6.2 são também apresentadas para o software *LocaDVD*, conforme pode ser visto na Tabela 6.3 e na Tabela 6.4. Entretanto, no caso do software *LocaDVD*, as métricas cobertura e precisão foram especificadas com relação ao tipo de interesse de software (base ou transversal).

Alguns pontos interessantes sobre os resultados obtidos para o software *LocaDVD* são:

1) Conforme já comentado, Sampaio *et al*. (2007) constataram que a precisão das abordagens para EROA para interesses transversais é satisfatória, mas a cobertura não. Essa situação foi constatada no caso da abordagem *Theme/Doc*, mas não no caso da abordagem *ObasCId*. O valor de cobertura global proporcionada pela abordagem *ObasCId* se aproximou bastante do valor da precisão global. Isso pode ter ocorrido devido ao apoio oferecido pela abordagem *ObasCId* ao engenheiros de software durante a identificação e classificação de interesses.

Tabela 6.3. Resultados do estudo experimental I para o software *LocaDVD*.

Abordagem *ObasCId* (Grupo 1)							
Partic.	**Cobertura IB (M6)**	**Cobertura IT (M7)**	**Cobertura Global (M5)**	**Precisão IB (M9)**	**Precisão IT (M10)**	**Precisão Global (M8)**	**Tempo (min) – M4**
P1	100	66	83	75	100	83	32
P2	100	66	83	60	100	71	22
P3	100	100	100	60	100	75	18
P4	66	66	66	100	100	100	42
P5	33	100	66	75	100	80	37
P6	100	100	100	75	100	86	22
P7	100	66	83	60	100	71	25
Média	**85,57**	**80,57**	**83,00**	**72,14**	**100,00**	**80,85**	**28,28**
Abordagem *Theme/Doc* (Grupo 2)							
Partic.	**Cobertura IB (M6)**	**Cobertura IT (M7)**	**Cobertura Global (M5)**	**Precisão IB (M9)**	**Precisão IT (M10)**	**Precisão Global (M8)**	**Tempo (min) – M4**
P8	0	66	33	0	100	66	18
P9	66	66	66	66	100	80	29
P10	100	33	66	75	100	80	15
P11	0	66	33	###	100	100	32
P12	100	66	71	60	100	71	13
P13	33	66	50	33	100	60	18
P14	66	33	50	50	100	75	21
Média	**52,14**	**56,57**	**52,71**	**47,43**	**100,00**	**76,00**	**20,85**

Legenda: IB – Interesse Base; IT – Interesse Transversal

Tabela 6.4. Interesses identificados no software *LocaDVD*.

Interesses	**Participantes *ObasCId***							**%**	**Participantes *Theme/Doc***							**%**
	1	**2**	**3**	**4**	**5**	**6**	**7**		**8**	**9**	**10**	**11**	**12**	**13**	**14**	
Transação	X	X	X	X	X	X	X	100		X	X		X		X	57
Pagamento	X	X	X			X	X	71			X		X	X		28
Recurso	X	X	X	X		X	X	86		X	X		X		X	57
Destino	X	X	X	X	X	X	X	100		X	X		X		X	57
Logging	X	X	X	X	X	X	X	100	X	X		X	X	X		72
Persistência			X		X	X		43	X			X		X		43
Média								**83,33**	**Média**							**52,33**
# Falsos positivos IB	1	2	2	0	1	1	2	-	1	1	1	0	2	2	1	-
# Falsos positivos IT	0	0	0	0	0	0	0	-	0	0	0	0	0	0	0	-

Legenda: IB – Interesse Base; IT – Interesse Transversal

2) O tempo para execução das duas abordagens reduziu, em comparação ao tempo gasto para identificação dos interesses do software *Health Watcher*, porém a diferença entre as abordagens *ObasCId* e *Theme/Doc* continuou, isto é, menos tempo foi necessário para a execução da abordagem *Theme/Doc*. A redução no tempo pode ser explicada pela característica dos softwares utilizados. Apesar de ambos apresentarem uma quantidade de interesses e requisitos similar, o software *LocaDVD* pertence a um domínio mais comum do que o software *Health Watcher*. Isso pode ter facilitado o processo de leitura e entendimento do documento de requisitos do software *LocaDVD* por parte dos participantes do experimento. A diferença do tempo de execução entre as duas abordagens aumentou de 4 (quatro) minutos (para o software *Health Watcher*) para 8 (oito) minutos (para o *LocaDVD*). Isso pode ser explicado pelo fato de que, para o experimento com o software *LocaDVD*, o catálogo de interesses utilizado era mais amplo, pois incluía também um conjunto de interesses funcionais.

3) A cobertura global proporciona pela abordagem *ObasCId* continua sendo maior do que a cobertura global proporcionada pela abordagem *Theme/Doc*, mesmo com um software e um conjunto de participantes diferentes; a precisão global proporcionada pela abordagem *ObasCId* também continua sendo maior do que a precisão proporcionada pela abordagem *Theme/Doc*, mas a diferença não é significativa.

4) A cobertura de interesses base proporcionada pela abordagem *ObasCId* é maior do que a cobertura de interesses transversais, situação observada também em Sampaio *et al*. (2007). Contudo, isso não aconteceu para a abordagem *Theme/Doc*. Uma possível explicação é o fato de os interesses transversais não funcionais existentes no software *LocaDVD* (“Persistência” e “*Logging*”) serem mais bem conhecidos do que aqueles existentes no software *Health Watcher*.

Na Tabela 6.5 encontra-se o resumo dos resultados do estudo experimental I, destacando-se os valores médios das métricas globais de cobertura e precisão e do tempo de execução de cada abordagem. Além disso, é possível observar o aumento proporcional do valor de cada métrica, da abordagem *Theme/Doc* para a abordagem *ObasCId*.

Tabela 6.5. Resumo dos resultados do estudo experimental I.

Aplicação	Cobertura Global (M5)		Precisão Global (M8)		Tempo (min) – M4	
	Theme/Doc	***ObasCId***	***Theme/Doc***	***ObasCId***	***Theme/Doc***	***ObasCId***
Health Watcher	40,81 (G1)	73,46 (G2) (↑ 1,80)	88,71 (G1)	89,42 (G2) (↑ 1,006)	41,28 (G1)	45,57 (G2) (↑ 1,103)
LocaDVD	52,71 (G2)	83,00 (G1) (↑ 1,57)	76,00 (G2)	80,85 (G1) (↑ 1,063)	20,85 (G2)	28,28 (G1) (↑ 1,356)

Legenda: G1 – Grupo 1; G2 – Grupo 2

Conforme pode ser visto nessa tabela, independentemente do software utilizado ou do grupo que desempenhou as atividades de identificação e classificação de interesses, a

abordagem *ObasCId* apresentou valores para cobertura e precisão maiores do que as da abordagem *Theme/Doc*. Porém, os aumentos mais significativos incidem sobre a métrica cobertura; no caso do experimento com o software *Health Watcher*, o valor da cobertura proporcionada pela abordagem *ObasCId* foi quase o dobro da cobertura da *Theme/Doc*, e para o software *LocaDVD*, a diferença chegou a quase 60%, em favor da *ObasCId*. Analogamente, o tempo para execução da abordagem *ObasCId* é sempre maior do que o tempo para execução da abordagem *Theme/Doc*, independentemente do software/grupos de participantes envolvido no estudo. Esse aumento variou de 10% à aproximadamente 36%.

6.3.3 Teste de Hipóteses

Apesar de os valores apresentados anteriormente indicarem que a utilização da abordagem *ObasCId* proporciona melhores valores de cobertura, com relação à identificação e classificação de interesses de software, faz-se necessário realizar a análise estatística dos dados, por meio de testes de hipótese, com o intuito de garantir maior confiabilidade às afirmações feitas.

O objetivo de um teste de hipótese é verificar se a hipótese nula (H_0) pode ser rejeitada, com algum grau de significância (confiança de que está sendo tomada a decisão correta), chegando assim, à aceitação da hipótese alternativa H_1. Antes de se aplicar um teste de hipóteses, entretanto, é necessário conhecer sobre qual tipo de distribuição de probabilidade os dados coletados no estudo se encontram organizados. Isto porque muitos testes de hipóteses, tais como o *t-test* (Montgomery, 2000), possuem como pré-requisito a necessidade de que os dados estejam distribuídos normalmente.

Para verificar a normalidade dos dados, aplicou-se o teste de normalidade conhecido como *Shapiro-Wilk* (Montgomery, 2000). A hipótese nula do teste *Shapiro-Wilk* é que os dados estão normalmente distribuídos. Caso o valor da probabilidade de se rejeitar incorretamente a hipótese nula em favor da hipótese alternativa (denominada, *W*) for maior do que a probabilidade de se aceitar corretamente a hipótese nula, para um determinado grau de significância (*p-value*), então esta hipótese é aceita.

Teste de hipóteses para o experimento com o software *Health Watcher*. Os dados referentes à cobertura proporcionada pela abordagem *Theme/Doc* = {42,85; 42,85; 42,85; 28,57; 57,14; 42,85; 28,57}, foram considerados normalizados com grau de significância p = 0,01, pois W = 0,84 e *Threshold* (p = 0,01) = 0,73. Para o conjunto de dados referente à cobertura proporcionada pela abordagem *ObasCId* = {71,42; 85,71; 85,71; 71,42; 57,14; 71,42; 71,42}, os dados também foram considerados normalizados com grau de significância p = 0,01, pois W = 0,84 e *Threshold* (p = 0,01) = 0,73. Isso significa que,

para os dois conjuntos de dados, é possível afirmar com 99% de confiança que estes dados estão distribuídos normalmente. A mesma situação de normalidade pode ser verificada para os conjuntos de dados relativos ao tempo gasto pelos usuários para execução das duas abordagens. Já o conjunto de dados referente à precisão proporcionada pelas abordagens não foi considerado normalizado.

Uma vez que os dados relacionados à cobertura e ao tempo de execução das abordagens foram considerados normalizados, aplicou-se o *t-test* para verificar as hipóteses do Quadro 6.11, relacionadas com as métricas M5 (cobertura global) e M4 (tempo). *T-test* é um teste estatístico paramétrico utilizado para comparação entre médias de duas amostras distintas. A hipótese nula deste método afirma que as duas médias comparadas são iguais. Comparando-se os valores médios da cobertura proporcionada pelas abordagens *Theme/Doc* (média = 40,81) e *ObasCId* (média = 73,46), apresentadas na Tabela 6.1, a hipótese nula H_{0M5} pode ser rejeitada com grau de significância p = 0,0004. Ou seja, com aproximadamente 99,9% de confiança, é possível afirmar que a cobertura proporcionada pela abordagem *ObasCId* é diferente da cobertura proporcionada pela abordagem *Theme/Doc*. Como o valor da cobertura proporcionada pela abordagem *ObasCId* é maior do que o da abordagem *Theme/Doc*, pode-se afirmar que a abordagem *ObasCId* foi mais efetiva, em termos de cobertura, para a identificação e classificação de interesses no software *Health Watcher*.

Com relação à média dos tempos gastos para realização das atividades das duas abordagens, *Theme/Doc* (média = 41 min) e *ObasCId* (média = 45 min), não foi possível obter indícios estatísticos, com nível de significância maior ou igual a 95%, de que estes tempos são diferentes, ou seja, não foi possível rejeitar a hipótese H_{0M4}.

O fato de o conjunto de dados referente à precisão não ter sido considerado normalizado restringe o uso do *t-test*. Sendo assim, para testar a hipótese H_{0M8}, aplicou-se o teste *Mann-Whitney* (Montgomery, 2000). *Mann-Whitney* é um teste não-paramétrico que permite que duas médias sejam comparadas sem a necessidade de os dados que as geraram estarem distribuídos normalmente. A hipótese nula deste teste afirma que as médias dos dois conjuntos de dados são idênticas. Comparando-se os valores médios da precisão proporcionada pelas abordagens *Theme/Doc* (média = 88,71) e *ObasCId* (média = 89,42), a hipótese nula H_{0M8} não pode ser rejeitada com nível de significância maior ou igual a 95%.

Em resumo, os testes de hipótese para o experimento com o software *Health Watcher* permitiram afirmar que há diferenças significativas entre a cobertura medida para as duas abordagens em análise, sendo que a abordagem *ObasCId* apresentou resultados melhores do que os da abordagem *Theme/Doc*. Porém, não é possível afirmar que há

diferenças estatisticamente significantes entre o tempo de execução e a precisão proporcionada pelas duas abordagens.

Teste de hipóteses para o experimento com o software *LocaDVD*. A partir da aplicação do teste *Shapiro-Wilk* (Montgomery, 2000), os conjuntos de dados que deram origem aos valores médios das métricas cobertura global (M5), precisão global (M8) e tempo (M4) para o experimento com o software *LocaDVD* foram considerados normalizados. Assim, as três hipóteses nulas descritas no Quadro 6.11 podem ser testadas por meio do *t-test*. Os valores das métricas M6, M7, M9 e M10 não foram analisados estatisticamente, pois a quantidade de interesses de cada tipo (base ou transversal) era reduzida (entre três e quatro interesses de cada tipo), o que comprometia a análise desses dados. Assim, apesar de os valores da métrica M6 dos participantes P8 e P11 serem considerados *outliers*, esse fato não prejudicou a análise dos valores de cobertura e precisão globais.

No Quadro 6.13, são apresentados os resultados dos testes de hipóteses realizados para o experimento com o software *LocaDVD*, descrevendo, para cada hipótese nula, a decisão tomada sobre ela (rejeitar ou não rejeitar) e o valor do *p-value* obtido com o teste. Cabe ressaltar que o menor valor de *p-value* aceito para rejeitar a hipótese nula neste trabalho é 0,05.

Como pode ser observado, apenas a hipótese nula H_{0M5}, referente à cobertura global proporcionada pelas duas abordagens analisadas, pode ser rejeitada, aceitando-se então a hipótese alternativa H_{1M5}: $M5_{ObasCId}$ {83,00} ≠ $M5_{Theme/Doc}$ {52,71}. Como o valor $M5_{ObasCId}$ > $M5_{Theme/Doc}$, pode-se afirmar que a efetividade da abordagem *ObasCId*, em termos de cobertura, é maior do que a da abordagem *Theme/Doc*.

Quadro 6.13. Teste de hipóteses para o experimento com o software *LocaDVD*.

Hipótese Nula	Decisão Tomada	*p-value*
H_{0M5}: $M5_{ObasCId}$ {83,00} = $M5_{Theme/Doc}$ {52,71}	Rejeitar H_{0M5}	0,001212
H_{0M8}: $M8_{ObasCId}$ {80,85} = $M8_{Theme/Doc}$ {76,00}	Não rejeitar H_{0M8}	0,224449
H_{0M4}: $M4_{ObasCId}$ {28,28} = $M4_{Theme/Doc}$ {20,85}	Não rejeitar H_{0M4}	0,054813

De modo análogo ao que ocorreu para o experimento com o software *Health Watcher*, as hipóteses nulas referentes à precisão global e ao tempo de execução das abordagens não puderam ser rejeitadas com o mínimo de 95% de nível de significância. Isso indica que não há diferenças significativas entre as duas abordagens analisadas quanto à precisão e ao tempo de execução das mesmas.

6.3.4 Ameaças à Validade do Estudo

Wohlin *et al*. (2012) afirmam que um estudo experimental está sujeito a situações que podem ameaçar a validade dos resultados obtidos com ele. As principais ameaças tratadas neste estudo são:

1) Ameaças à validade de conclusão. Refere-se às questões que afetam a habilidade de se tirar conclusões corretas a respeito do objeto de estudo do experimento. Um exemplo de ameaça deste tipo diz respeito à escolha do método estatístico adequado para análise dos dados. Neste estudo, os testes estatísticos adotados para comparação dos valores médios das métricas foram *t-test* e *Mann-Whitney*. O *t-test* requer que os dados das amostras estejam normalmente distribuídos, dessa forma, antes de aplicá-lo verificou-se esse pressuposto por meio do teste *Shapiro-Wilk*. Para os casos em que o resultado do teste *Shapiro-Wilk* foi negativo, aplicou-se o teste *Mann-Whitney*.

2) Ameaças à validade interna. Refere-se às questões que afetam a habilidade de se assegurar que os resultados foram, de fato, obtidos em decorrência dos tratamentos (isto é, das abordagens *ObasCId* e *Theme/Doc*) e não por uma coincidência. Uma ameaça desse tipo pode ser o modo como os participantes foram selecionados e agrupados. Outro ponto que pode ter influenciado os resultados foi a utilização de alunos de graduação e pós-graduação como participantes do estudo. Como formas de mitigar essas ameaças, não foram apresentadas expectativas a favor ou contra qualquer abordagem analisada, para que os participantes não fossem influenciados. Além disso, os estudantes foram agrupados adequadamente, de acordo com seus níveis de experiência para que os grupos ficassem homogêneos, evitando assim, discrepâncias com relação à experiência dos participantes. Outra ameaça a ser destacada está relacionada ao aprendizado que os participantes obtiveram sobre EROA entre a primeira e a segunda fases do estudo experimental. Para minimizar esse viés, utilizou-se aplicações diferentes, com tipos de interesses distintos, nas duas fases do experimento. Por exemplo, há apenas um interesse em comum, a saber, “Persisitência”, nos softwares *Health Watcher* e *LocaDVD*.

3) Ameaças à validade externa. Refere-se às questões que afetam a habilidade de se generalizar os resultados do experimento para um contexto mais amplo do que aquele selecionado para o estudo. Sendo assim, os fatores que podem ter influenciado os resultados deste experimento são: (i) os softwares utilizados no experimento, ou seja, *Health Watcher* e *LocaDVD*; (ii) a qualidade dos formulários apresentados aos participantes; e (iii) a quantidade de amostras (participantes) utilizadas. Um ponto importante a ser destacado é que os softwares utilizados no experimento possuíam poucos requisitos e interesses, o que pode ter influenciado a qualidade dos resultados obtidos para cobertura e precisão. Contudo, a escolha de softwares com muitos requisitos também poderia influenciar

negativamente os resultados, em razão do esforço que seria necessário por parte dos participantes do experimento para a identificação e classificação dos interesses do software. Com o intuito de mitigar essas possíveis ameaças, é necessário pesquisar estratégias mais adequadas para condução de experimentos com softwares de grande porte.

6.4 Estudo Experimental II – Facilidade de uso e utilidade da ferramenta *ObasCId-Tool*

Este estudo experimental refere-se aos objetivos de avaliação O2 e O3 (Seção 6.2.1), que visam a avaliar a ferramenta *ObasCId-Tool*, com respeito à utilidade e à facilidade de uso percebidas por seus usuários.

6.4.1 Planejamento

O planejamento deste experimento também foi realizado de acordo com o modelo proposto por Wohlin *et al*. (2012).

1. Seleção do contexto. O contexto deste estudo consiste em uma situação de uso da ferramenta *ObasCId-Tool*, visando a gerenciar os recursos necessários para identificação e classificação de interesses a partir de requisitos de software. O estudo foi realizado com 11 (onze) alunos de graduação e pós-graduação em Ciência da Computação, na disciplina Engenharia de Software. Tais alunos são diferentes daqueles utilizados no primeiro estudo experimental, de forma que não houve interferência nos resultados do experimento, por conta do aprendizado prévio desses participantes. Este estudo também não gerou impacto na nota dos alunos, para que isso não influenciasse nos resultados.

2. Formulação de hipóteses. Quatro hipóteses foram elaboradas para este estudo, duas relacionadas ao constructo utilidade percebida e duas relacionadas ao constructo facilidade de uso percebida (Quadro 6.14).

3. Seleção de variáveis e seleção dos participantes. As variáveis sob análise neste experimento são as porcentagens de respostas para cada questão do questionário elaborado com base no modelo TAM (Davis, 1993), bem como a porcentagem de atividades concluídas com sucesso pelos participantes do estudo.

Os participantes foram selecionados por meio de amostragem não probabilística por conveniência. Dos onze alunos participantes do experimento, seis eram de graduação e cinco de pós-graduação. Contudo, analogamente ao que ocorreu no primeiro estudo

experimental, em termos de experiência sobre EROA, de acordo com o questionário de caracterização de perfil, todos os participantes afirmaram não terem trabalhado com EROA, nem com identificação de interesses de software em nível de código. Quando perguntados sobre seu nível de conhecimento sobre Engenharia de Requisitos, dez participantes afirmaram estar em um nível intermediário, cujo contato com esse assunto se deu por meio de disciplinas de Engenharia de Software. Um participante afirmou estar em nível avançado, por já ter trabalhado com engenharia de requisitos, seja na área acadêmica ou na prática profissional.

Quadro 6.14. Hipóteses para o estudo experimental II.

Hipóteses referentes à utilidade percebida	
H_{0UP}	Não há consenso sobre o constructo utilidade percebida, com relação ao uso da ferramenta *ObasCId-Tool*, isto é, H_{0UP}: M11 + M12 + M13 + M14 = M15 + M16 + M16.
H_{1UP}	Há consenso sobre o constructo utilidade percebida, com relação ao uso da ferramenta *ObasCId-Tool*, isto é, H_{0UP}: M11 + M12 + M13 + M14 ≠ M15 + M16 + M16.
Hipóteses referentes à facilidade de uso percebida	
H_{0FU}	Não há consenso sobre o constructo facilidade de uso percebida, com relação ao uso da ferramenta *ObasCId-Tool*, isto é, H_{0FU}: M11 + M12 + M13 + M14 = M15 + M16 + M16 ou M18 = 75% (ver Quadro 6.10).
H_{1FU}	Há consenso sobre o constructo facilidade de uso percebida, com relação ao uso da ferramenta *ObasCId-Tool*, isto é, H_{0FU}: M11 + M12 + M13 + M14 ≠ M15 + M16 + M16 e M18 ≠ 75% (ver Quadro 6.10).

4. Projeto e execução do experimento realizado. É importante salientar que nenhum dos participantes do experimento tinha utilizado a ferramenta *ObasCId-Tool* anteriormente. Assim, antes do início do experimento, foi realizado um treinamento com duração de 40 (quarenta) minutos, que teve como objetivo apresentar a ferramenta *ObasCId-Tool* aos participantes.

Na fase de execução do experimento, os participantes deveriam realizar uma série de atividades na ferramenta *ObasCId-Tool*, que consistiam em gerenciar os recursos necessários para identificação e classificação de interesses a partir de requisitos de software. Alguns exemplos de atividades são “cadastrar um novo catálogo de interesses de software”, “cadastrar interesses para um catálogo existente”, “cadastrar palavras-chave e relacionamentos para interesses existentes”, “cadastrar um novo documento de requisitos”, entre outras. Além disso, após a conclusão de todas as atividades, solicitou-se aos participantes do estudo que respondessem a um questionário eletrônico que continha as questões Q1O3 à Q8O3 do Quadro 6.5 e todas as questões do Quadro 6.6.

6.4.2 Análise dos Resultados

Na Tabela 6.6 e na Tabela 6.7 são apresentados, respectivamente, os resultados obtidos por meio do formulário eletrônico respondido pelos participantes, com relação aos

constructos utilidade percebida e facilidade de uso percebida. A primeira coluna dessas tabelas contém as afirmações dadas aos participantes; as colunas 2 a 8 apresentam a porcentagem de participantes que escolheram as opções 1 a 7, respectivamente; por fim, a nona e décima colunas apresentam, respectivamente, a porcentagem de participantes que escolheram opções negativas ou neutras (1, 2, 3 ou 4) e positivas (5, 6 ou 7). A última linha dessas tabelas apresenta a média da porcentagem de participantes que escolheram uma determinada opção.

Tabela 6.6. Resultados para o constructo facilidade de uso percebida.

Questão	Opções (% participantes)							-	+
	1	2	3	4	5	6	7		
Eu gostei de trabalhar com a *ObasCId-Tool*.	0	0	8	0	17	33	42	**8**	**92**
O acesso à *ObasCId-Tool* é simples.	0	8	0	0	8	33	50	**8**	**92**
Usar a *ObasCId-Tool* é uma boa ideia.	0	0	0	0	33	33	34	**0**	**100**
Na *ObasCId-Tool* eu sempre sei onde estou e como chegar aonde quero.	0	0	17	17	17	25	25	**33**	**67**
Os recursos de manipulação da *ObasCId-Tool* estão claros e fáceis de achar.	0	0	0	8	25	42	25	**8**	**92**
Minha interação com a *ObasCId-Tool* é clara e compreensível.	0	0	8	0	17	50	25	**8**	**92**
Na *ObasCId-Tool*, é fácil encontrar a informação que desejo.	0	8	0	8	17	25	42	**17**	**83**
A *ObasCId-Tool* possui visual/interface atraente.	0	0	0	8	17	17	58	**8**	**92**
Média	**0**	**2**	**4**	**5**	**19**	**32**	**38**	**11**	**89**

Legenda: 1 (discordo totalmente); 2 (discordo em grande parte) ; 3 (discordo parcialmente); 4 (neutro); 5 (concordo parcialmente); 6 (concordo em grande parte) ; 7 (concordo totalmente).

Tabela 6.7. Resultados para o constructo utilidade percebida.

Questão	Opções (% participantes)							-	+
	1	2	3	4	5	6	7		
Utilizar a *ObasCId-Tool* é importante e adiciona valor ao meu trabalho.	0	0	0	25	17	25	33	**25**	**75**
A *ObasCId-Tool* é útil no processo de identificação e classificação de ITs.	0	0	0	0	17	33	50	**0**	**100**
Usar a *ObasCId-Tool* pode aumentar meu desempenho durante a identificação e classificação de ITs em requisitos de software.	0	0	0	8	17	25	50	**8**	**92**
A *ObasCId-Tool* pode facilitar a realização do meu trabalho.	0	0	0	25	17	17	42	**25**	**75**
A *ObasCId-Tool* produz os resultados que espero de uma ferramenta de suporte à identificação e classificação de ITs.	0	0	0	33	8	33	25	**33**	**67**
Eu pretendo integrar a *ObasCId-Tool* à minha rotina de trabalho.	0	0	0	42	25	25	8	**42**	**58**
Eu recomendarei o uso da *ObasCId-Tool*.	0	0	0	0	25	50	25	**0**	**100**
Os conceitos da área de identificação e classificação de ITs foram abordados por completo na *ObasCId-Tool*.	0	0	0	17	17	25	42	**17**	**83**
Média	**0**	**0**	**0**	**19**	**18**	**29**	**34**	**19**	**81**

Legenda: 1 (discordo totalmente); 2 (discordo em grande parte) ; 3 (discordo parcialmente); 4 (neutro); 5 (concordo parcialmente); 6 (concordo em grande parte) ; 7 (concordo totalmente).

De modo geral, pode-se notar que, tanto para a utilidade percebida, quanto para a facilidade de uso percebida, a ferramenta *ObasCId-Tool* obteve um percentual maior de opções positivas do que negativas ou neutras. Quanto à facilidade de uso, a afirmação mais bem aceita pelos participantes foi a "*Usar a ObasCId-Tool é uma boa ideia*", enquanto que a afirmação que obteve o pior resultado foi "*Na ObasCId-Tool eu sempre sei onde estou e como chegar aonde quero.*", acompanhada de "*Na ObasCId-Tool, é fácil encontrar a informação que desejo.*". Isso indica que os recursos de navegação e *feedback* sobre a localização do usuário na ferramenta devem ser reavaliados.

Quanto à utilidade percebida, as duas afirmações com 100% de opções positivas foram "*A ObasCId-Tool é útil no processo de identificação e classificação de interesses de software.*" e "*Eu recomendarei o uso da ObasCId-Tool.*", o que indica que os participantes encontraram na ferramenta *ObasCId-Tool* recursos potencialmente positivos para o processo de identificação e classificação de interesses a partir de requisitos de software. É importante salientar que os participantes do experimento passaram por um treinamento sobre EROA, que incluiu exposições teóricas sobre a área, bem como a realização de exercícios práticos de identificação e classificação de interesses a partir de requisitos de software. Isso foi feito para que os participantes pudessem opinar de forma mais consciente sobre essas afirmações. A afirmação com menor porcentagem de aprovação foi "*Eu pretendo integrar a ObasCId-Tool à minha rotina de trabalho.*". Isso se justifica pelo fato de os participantes do experimento pertencerem a áreas de pesquisa diversas dentro da engenharia de software.

Conforme mencionado na Seção 6.2.2, com o intuito de complementar os dados para análise da característica de facilidade de uso da ferramenta *ObasCId-Tool*, os participantes tiveram que realizaram um conjunto de atividades relacionadas ao gerenciamento de recursos para identificação e classificação de interesses. A primeira atividade teve como finalidade verificar as habilidades dos usuários nas seguintes funções da ferramenta: (i) adicionar um novo interesse a um catálogo de interesses existente; (ii) cadastrar um relacionamento de dependência entre interesses; (iii) criar uma unidade de identificação; (iv) executar a unidade de identificação e descrever os interesses identificados para cada requisito do software, bem como listar as ocorrências descritas na ferramenta; e (v) corrigir o documento de requisitos/catálogo de interesses para que as ocorrências fossem solucionadas.

Já a segunda atividade tinha como objetivo verificar a habilidade dos usuários com relação ao entendimento do diagrama de classes UML utilizado para representar um catálogo de interesses, conforme apresentado na Seção 4.3. Assim, dado um catálogo de interesses representado nesse tipo de diagrama, os usuários tiveram que realizar o cadastro

dos seguintes elementos na ferramenta *ObasCId-Tool*: (i) interesses de software; (ii) palavras-chave; e (iii) relacionamentos existentes entre interesses.

Na Tabela 6.8 são apresentados os resultados da execução das atividades descritas acima para os onze participantes deste estudo experimental. A coluna 1 apresenta o código da atividade; a atividade 1 foi desmembrada em 5 subatividades e a atividade 2, em 3 subatividades, de acordo com as funções que os usuários deviam desempenhar na ferramenta. As colunas 2 à 12 representam quais atividades foram concluídas com sucesso (X) ou não pelos participantes do estudo. A coluna 13 apresenta a porcentagem de participantes que concluíram determinada atividade; já a última linha da tabela apresenta a porcentagem de atividades concluídas com sucesso por participante.

Tabela 6.8. Atividades concluídas pelos usuários da ferramenta *ObasCId-Tool*.

Atividade	Atividades concluídas por participante											% Concluintes
	1	2	3	4	5	6	7	8	9	10	11	
1.1	X	X	X	X	X	X	X	X	X	X	X	**100**
1.2	X	X	X	X	X	X	X	X	X	X	X	**100**
1.3	X	X	X	X	X	X	X	X	X	X	X	**100**
1.4	X	X	X	X	X		X	X	X	X	X	**90**
1.5	X	X	X	X	X	X	X	X	X	X	X	**100**
2.1	X	X	X	X	X	X	X	X	X	X	X	**100**
2.2	X		X		X	X	X	X	X	X	X	**81**
2.3	X	X	X	X	X	X	X	X	X	X	X	**100**
% Ativid. concluídas	**100**	**87**	**100**	**87**	**100**	**87**	**100**	**100**	**100**	**100**	**100**	**Média = 96,5**

Como pode ser visto, 8 (oito) dos 11 (onze) usuários conseguiram concluir com sucesso todas as oito atividades propostas a eles. Além disso, o menor desempenho foi de 87% das atividades concluídas, o que é um desempenho satisfatório. Seis das oito atividades foram concluídas por todos os usuários e a atividade com menor percentual de concluintes (81%) foi a atividade 2.2, que consistia no cadastramento de palavras-chave para os interesses de um catálogo. Esse pode ser um indício de que a representação de catálogos de interesses, por meio de diagramas de classes UML, pode confundir o entendimento dos usuários sobre quais palavras-chave estão vinculadas aos interesses do catálogo.

6.4.3 Teste de Hipóteses

Para verificar a validade ou não da hipótese nula referente à utilidade percebida, H_{0UP}, deve-se comparar a porcentagem média de opiniões positivas sobre essa característica com a porcentagem média de opiniões negativas. Antes, porém, deve-se verificar se os dados da amostra estão em conformidade com a distribuição normal de probabilidade. Assim, o teste *Shapiro-Wilk* foi aplicado ao conjunto de opiniões positivas e

negativas (ou neutras) da Tabela 6.7. Com relação ao conjunto de opiniões positivas {75; 100; 92; 75; 67; 58; 100; 83}, o mesmo foi considerado normalizado com grau de significância p = 0,01, pois W = 0,93 e Threshold (p = 0,01) = 0,749. O mesmo ocorreu para o conjunto de dados referente às opiniões negativas (ou neutras) {25; 0; 8; 25; 33; 42; 0; 17}.

Uma vez que os dois conjuntos de dados foram considerados normalizados, aplicou-se o *t-test* para verificar a validade ou não da hipótese H_{0UP}. Comparando-se o valor médio de opiniões positivas a respeito da utilidade da ferramenta *ObasCId-Tool* (81%) com o valor médio de opiniões negativas ou neutras (19%), a hipótese nula H_{0UP} pode ser rejeitada com grau de significância p = 0,00001. Ou seja, com aproximadamente 99,9% de confiança, é possível afirmar que a porcentagem média de opiniões positivas difere-se da porcentagem média de opiniões negativas ou neutras. Além disso, como a percentagem de opiniões negativas ou neutras é menor do que a de opiniões positivas, de acordo com a regra de interpretação do Quadro 6.10, a ferramenta *ObasCId-Tool* pode ser considerada satisfatória com relação ao constructo *utilidade percebida*.

Para a análise do constructo *facilidade de uso*, a hipótese H_{0FU} é composta de duas partes. Primeiramente, deve-se verificar se a porcentagem média de opiniões positivas dos usuários a respeito da facilidade de uso da ferramenta é diferente da porcentagem média de opiniões negativas ou neutras. Posteriormente, deve-se verificar se a porcentagem de tarefas concluídas pelos usuários da ferramenta é diferente de 75%.

Inicialmente, aplicou-se o teste para verificar a normalidade do conjunto de opiniões dos usuários a respeito dessa característica e verificou-se que, para os dois conjuntos (de opiniões negativas ou neutras e de opiniões positivas), os dados não encontram-se normalizados, o que restringe o uso do *t-test*. Sendo assim, para testar essa parte da hipótese H_{0FU}, aplicou-se novamente o teste *Mann-Whitney*. Comparando-se os valor médio de opiniões positivas a respeito da facilidade de uso da ferramenta *ObasCId-Tool* (89%) com o valor médio de opiniões negativas ou neutras (11%), apresentadas na Tabela 6.6, a primeira parte da hipótese nula H_{0FU} pode ser rejeitada com grau de significância p = 0,00094. Ou seja, é possível afirmar que a porcentagem média de opiniões positivas difere-se da porcentagem média de opiniões negativas ou neutras dos usuários a respeito da facilidade de uso da ferramenta *ObasCId-Tool*.

A porcentagem média de atividades concluídas com sucesso pelos participantes do estudo experimental também não está em conformidade com a distribuição normal de probabilidade. Assim, novamente o teste *Mann-Whitney* foi aplicado, comparando-se a porcentagem média de atividades concluídas com sucesso (96,5%) com o valor 75%. Como resultado, a segunda parte da hipótese nula H_{0FU} pode ser rejeitada com grau de significância $p = 8 * 10^{-5}$.

Assim, como as duas partes da hipótese H_{0FU} foram rejeitadas, então, pode-se considerar a hipótese alternativa H_{1FU}, isto é, que o valor médio de opiniões positivas a respeito da facilidade de uso da ferramenta *ObasCId-Tool* é diferente do valor médio de opiniões negativas ou neutras e que a porcentagem média de atividades concluídas pelos participantes do estudo também é diferente de 75%. Além disso, como a percentagem de opiniões negativas ou neutras é menor do que a de opiniões positivas e a porcentagem média de atividades concluídas é maior do que 75%, de acordo com a regra de interpretação do Quadro 6.10, a ferramenta *ObasCId-Tool* pode ser considerada satisfatória com relação ao constructo *facilidade de uso percebida*.

6.4.4 Ameaças à Validade do Estudo

1) Ameaças à validade de conclusão. Analogamente ao que foi dito para o estudo experimental I, um exemplo de ameaça deste tipo diz respeito à escolha do método estatístico adequado para análise dos dados. No caso deste estudo, dois testes estatísticos foram adotados *t-test* e *Mann-Whitney*. O *t-test* requer dados normalmente distribuídos, dessa forma, o teste de normalidade de *Shapiro-Wilk* foi aplicado para confirmar essa situação antes da aplicação do mesmo. Para os casos em que o teste *Shapiro-Wilk* não indicou normalidade do conjunto de dados, o teste *Mann-Whitney* foi aplicado, por ser um teste não-paramétrico aplicável a conjuntos de dados que não seguem uma distribuição de probabilidade específica.

2) Ameaças à validade interna. Novamente, um ponto que pode ter influenciado os resultados foi a utilização de alunos de graduação e pós-graduação como participantes do experimento. Contudo, não foram demonstradas expectativas a favor ou contra a ferramenta analisada, para que os participantes não fossem influenciados. Além disso, os estudantes passaram por um mesmo treinamento com duração fixa para que nenhum deles tivesse privilégios sobre os demais.

3) Ameaças à validade externa. Os fatores que podem ter influenciado nos resultados deste experimento são: (i) a qualidade dos formulários com as atividades apresentadas aos participantes; e (ii) a quantidade de amostras (participantes) estudadas. Com o intuito de mitigar essas possíveis ameaças, novos experimentos com outros grupos de participantes e com aplicações diferentes devem ser realizados.

6.5 Considerações Finais

Este capítulo apresentou o planejamento e a execução de dois estudos experimentais que objetivaram verificar: (i) a efetividade e eficiência da abordagem *ObasCId*; e (ii) a aceitação da ferramenta *ObasCId-Tool* por parte de seus usuários, com relação à sua utilidade e facilidade de uso.

Os estudos foram realizados com grupos de catorze e onze alunos de graduação e pós-graduação em Ciência da Computação, respectivamente, sendo que o segundo grupo de alunos era diferente do primeiro. Assim, vinte e cinco alunos no total foram utilizados nos dois estudos experimentais realizados.

No primeiro estudo, os participantes tiveram que utilizar as abordagens *ObasCId* e *Theme/Doc* para a identificação e classificação dos interesses de dois softwares pertencentes a domínios distintos, a saber, *Health Watcher* (2015) e *LocaDVD* (Viana, 2009). Como principais resultados, notou-se que a abordagem *ObasCId* proporcionou maior cobertura para a identificação e classificação de interesses do que a abordagem *Theme/Doc* nos dois softwares analisados. A cobertura média proporcionada pela abordagem *ObasCId* variou entre 73,46% e 83%, enquanto que a da *Theme/Doc* variou entre 40,81% e 52,71%, considerando softwares e grupos de participantes diferentes. Outra constatação interessante foi que não houve diferenças significativas quanto à precisão proporcionada por essas abordagens; além disso, os valores de precisão obtidos podem ser considerados altos para as duas abordagens.

O tempo de execução das duas abordagens é um fator importante a ser considerado, uma vez que ele está diretamente relacionado à produtividade proporcionada por essa abordagem aos seus usuários. Por meio dos resultados obtidos, notou-se que não há diferenças significativas em termos de tempo para execução das abordagens *ObasCId* e *Theme/Doc*. Essa conclusão reforça os benefícios trazidos pela abordagem *ObasCId*, pois há indícios de que os recursos introduzidos nessa abordagem não impactarão na produtividade da equipe de desenvolvimento de software.

As principais conclusões obtidas a partir do segundo estudo experimental é que a ferramenta *ObasCId-Tool* foi bem aceita por seus usuários, quanto às suas características de facilidade de uso e utilidade. Notou-se ainda que, com um treinamento de quarenta minutos, a maioria dos usuários da ferramenta conseguiu concluir todas as atividades sugeridas. Finalizando o conteúdo desta tese, o Capítulo 7 apresenta as considerações finais deste doutorado, destacando-se as principais contribuições trazidas pelo mesmo, bem como suas limitações. Além disso, são elencadas algumas propostas para continuidade deste trabalho.

Capítulo 7

CONSIDERAÇÕES FINAIS

Conforme já comentado, a EROA objetiva promover melhorias quanto à separação de interesses durante as fases iniciais do desenvolvimento de software, oferecendo estratégias mais adequadas para a identificação, modularização e composição de interesses transversais. Com a melhoria na separação dos interesses de um software, espera-se que, quando um interesse transversal for modificado, a influência dessas modificações nos demais requisitos do software não acarrete perda de modularização (Sampaio *et al*., 2007).

Dentre as várias abordagens para EROA propostas nos últimos anos, pode-se notar que tais abordagens foram concebidas basicamente de três maneiras: (i) evoluindo-se abordagens já conhecidas para ER tradicional, como por exemplo, a modelagem baseada em pontos de vista (Rashid *et al*., 2002) e cenários (Whittle e Araújo, 2004); (ii) propondo-se novas abordagens, como *Theme/Doc* (Baniassad e Clarke, 2004), *EA-Miner* (Sampaio *et al*., 2005), *MDSoC* (Moreira *et al*., 2005) e *EROA/XML* (Soeiro *et al*., 2006); ou (iii) adaptando-se outras abordagens para EROA já existentes, com o intuito de complementá-las ou minimizar algumas de suas limitações (Chitchyan *et al*., 2006; Clarke e Baniassad, 2005; Rashid *et al*., 2003). Algumas das abordagens citadas foram apresentadas no Capítulo 2 desta tese, destacando-se seus pontos fortes e limitações.

Apesar de existirem diversas abordagens para EROA na literatura, há escassez de estudos cujo intuito seja prover uma compreensão clara e consensual sobre o domínio de interesses de software, destacando-se as características de um interesse, como ele se relaciona com outros interesses e com os requisitos do software, entre outros. Com o intuito de mitigar essa deficiência, propôs-se uma ontologia para o domínio de interesses de software, denominada *O4C* (*Ontology for Concerns*). Os detalhes dessa ontologia, bem como da forma como ela foi obtida/desenvolvida foram apresentados no Capítulo 3.

Aliado ao problema relatado anteriormente, está o fato de que as abordagens para EROA existentes na literatura deixam a desejar quanto ao oferecimento de recursos apropriados para apoiar engenheiros de software durante a atividade de identificação e

classificação de interesses de software. Essa foi a segunda preocupação atacada neste doutorado e ela foi contemplada em duas partes: primeiramente, procurou-se oferecer ao engenheiro de software uma base de conhecimentos sobre tipos específicos de interesses de software, por meio da qual ele pudesse encontrar informações que o auxiliassem no processo de identificação e classificação de interesses a partir de requisitos de software.

Para isso, propôs-se o uso de catálogos para tipos específicos de interesses de software, que consistem em instâncias dos conceitos e relacionamentos existentes na ontologia *O4C*. As primeiras seções do Capítulo 4 apresentam alguns exemplos de catálogos de interesses elaborados a partir de fontes diversas, tais como uma linguagem de padrões para gestão de recursos de negócio, denominada GRN (Braga, 2002), e um projeto de software, denominado *Health Watcher* (2015), para o qual a identificação e classificação de interesses já havia sido concluída.

Contudo, a simples existência desses catálogos não seria suficiente para aprimorar a efetividade do processo de identificação e classificação de interesses. Assim, em segundo lugar, confeccionou-se a abordagem para EROA denominada *ObasCId* (*Ontologically-based Concern Identification*) (Capítulo 4), que oferece recursos para que o engenheiro de software possa utilizar o conteúdo disponível nesses catálogos, bem como nos requisitos do software, de forma sistemática e planejada, a fim de obter melhores resultados com a identificação e classificação de interesses.

Além da abordagem *ObasCId*, também foi desenvolvida uma ferramenta computacional denominada *ObasCId-Tool*, que automatiza algumas das atividades propostas para essa abordagem. Essa ferramenta foi apresentada no Capítulo 5 e os resultados da avaliação da mesma estão no Capítulo 6. A *ObasCId-Tool*, quando avaliada, foi bem aceita por seus usuários, quanto às suas características de utilidade e facilidade de uso.

Um estudo experimental também foi conduzido sobre a *ObasCId*, com o intuito de confrontar os resultados obtidos por meio dessa abordagem com os de outra abordagem para EROA existente na literatura (Capítulo 6), denominada *Theme/Doc* (Clarke e Baniassad, 2005; Baniassad e Clarke, 2004). Como resultados, notou-se que a abordagem *ObasCId* pode contribuir para o aprimoramento da cobertura do processo de identificação e classificação de interesse a partir de requisitos de software, sem prejudicar sua precisão e sua eficiência.

7.1 Contribuições e Limitações

Com base no que foi exposto anteriormente, bem como nos demais capítulos desta tese, ressalta-se como principais contribuições deste doutorado:

- O estabelecimento do estado da arte referente à EROA e ao uso de ontologias no contexto da Engenharia de Requisitos, por meio dos Mapeamentos Sistemáticos conduzidos neste trabalho (Parreira Júnior e Penteado, 2014; Parreira Júnior e Penteado, 2015a; Parreira Júnior e Penteado, 2015d);
- A ontologia para interesses de software *O4C*, que provê uma definição conceitual clara a respeito do domínio de interesses de software, destacando os principais conceitos e relacionamentos envolvidos neste domínio. A partir do entendimento e uso dos conceitos contidos nessa ontologia, pode-se elaborar abordagens, métodos e ferramentas para EROA compatíveis entre si, uma vez que são baseados em uma conceitualização comum. Além disso, essa ontologia pode ser utilizada como base para confecção de catálogos de interesses mais amplos e que podem ser úteis no processo de identificação e classificação de interesses;
- A abordagem *ObasCId*, que provê recursos apropriados aos engenheiros de software para desempenharem a atividade de identificação e classificação de interesses em requisitos de software de forma mais efetiva; e
- A ferramenta computacional *ObasCId-Tool*, que provê suporte adequado para geração e compartilhamento do conhecimento a respeito dos interesses de software, bem como para identificação e classificação de interesses a partir de requisitos de software.

Como principais limitações dos produtos gerados nesta pesquisa, pode-se citar:

- A dependência da qualidade dos resultados do processo de identificação e classificação de interesses proposto pela abordagem *ObasCId* da existência de bons catálogos de interesses de software. Há algumas propostas de catálogos para requisitos não funcionais, porém, conforme já comentado, eles não são totalmente adequados para aplicação no contexto da EROA;
- A dependência da qualidade dos resultados da atividade de especificação do interesse principal de cada requisito da experiência do engenheiro de software que aplica a abordagem *ObasCId*. Além disso, essa é uma atividade não automatizada pela ferramenta *ObasCId-Tool*, o que pode afetar a produtividade da equipe de desenvolvimento. Apesar de essa tomada de decisão ser sobre um conjunto menor de interesses, isto é, apenas sobre aqueles interesses que foram

previamente identificados para o requisito em análise, faz-se necessário propor estratégias para reduzir essa dependência, bem como melhorar a eficiência para execução dessa atividade;

- A qualidade dos resultados do processo de identificação e classificação de interesses com a abordagem *ObasCId* é afetada pelo correto entendimento do modelo gráfico (diagrama de classes UML) utilizado para representação dos catálogos de interesses. O diagrama de classes UML pode não ser a melhor forma de se apresentar um catálogo de interesses, principalmente quando esse é muito amplo. Assim, é necessário estudar outras estratégias para visualização das informações disponíveis em um catálogo, como por exemplo o uso de tabelas, grafos, entre outros modelos gráficos;
- A falta de estudos experimentais que comparem a ferramenta *ObasCId-Tool* com outras ferramentas para EROA propostas na literatura. Isso ocorreu, principalmente, devido à escassez de ferramentas que ofereçam apoio à atividade de identificação e classificação de interesses. Apenas a ferramenta *EA-Miner* oferecia esse suporte, mas ela não encontrava-se disponível. A maioria das ferramentas propostas são para outras atividades da EROA, como representação e composição de interesses. Além disso, conforme comentado na Seção 5.11, a maioria delas está indisponível ou obsoleta; e
- A falta de apoio à atividade de composição de interesses de software. O enfoque da abordagem *ObasCId* está na identificação e classificação de interesses de software, apresentando também alguns recursos para a detecção de conflitos entre interesses, como a "matriz de contribuição". Há recursos ainda para representação de interesses, como a "lista de requisitos e interesses identificados", contudo, faltam recursos para contemplar a atividade de composição de interesses da EROA. Isto é, na abordagem *ObasCId*, bem como na ferramenta *ObasCId-Tool*, não há como especificar a maneira como um interesse transversal se compõe aos interesses base afetados por ele. Esta atividade da EROA não foi abordada neste doutorado pelas seguintes razões: (i) a atividade de identificação e classificação de interesses é tida como um "gargalo" no processo de EROA (Sampaio *et al*., 2007), tornando-se assim, um problema de pesquisa interessante e desafiador; (ii) as demais atividades da EROA, como a atividade "composição de interesses", dependem dos resultados da atividade "identificação e classificação de interesses", sendo assim, optou-se por aprimorar os resultados dessa atividade primariamente.

7.2 Trabalhos Futuros

Com base nas contribuições e limitações apresentadas na Seção 7.1, algumas propostas para trabalhos futuros podem ser elencadas:

- Extensão da abordagem *ObasCId*, bem como da ferramenta *ObasCId-Tool*, para que elas contemplem a atividade de "composição de interesses" da EROA. A ideia seria implementar recursos como as "regras de composição", existentes em algumas das abordagens para EROA tratadas no Capítulo 2;
- Extensão da ontologia *O4C*, bem como da abordagem *ObasCId* e da ferramenta *ObasCId-Tool*, para permitir a utilização de outros tipos de relacionamentos entre requisitos no processo de identificação e classificação de interesses;
- Confecção de métodos para transformação automatizada de catálogos de requisitos não funcionais em catálogos de interesses de software baseados na ontologia *O4C*. Essa ideia consiste em tentar estabelecer uma relação entre os conceitos da ontologia *O4C* e os conceitos dos catálogos de requisitos não funcionais existentes na literatura, com o objetivo de permitir a construção de procedimentos automatizados para transformação das informações contidas nos catálogos de requisitos não funcionais em catálogos de interesses de software baseados na ontologia *O4C*;
- Criação de heurísticas para a especificação do interesse principal de cada requisito. Tais heurísticas poderiam servir como diretrizes para facilitar a tomada de decisão por parte do engenheiro de software, reduzindo assim, a dependência da qualidade dos resultados do processo de identificação e classificação de interesses da experiência do profissional que o aplica. Além disso, o uso de heurísticas poderia ser considerado como um ponto de partida para a automatização dessa atividade na ferramenta *ObasCId-Tool*;
- Elaboração de algoritmos para extração automática de palavras-chave para interesses de software, a partir de documentos de requisitos de projetos já concluídos. Muitas vezes, conforme apresentado no Capítulo 4, é possível construir catálogos de interesses de software a partir de dados históricos de projetos já finalizados. Contudo, em geral, algumas informações relevantes para esses interesses, tais como as palavras-chave que podem ser utilizadas para sua identificação não são explicitamente mencionadas nestes projetos. Uma ideia seria desenvolver algoritmos que, baseados em técnicas de processamento de

linguagem natural, extraísse candidatos a palavras-chave que poderiam ser utilizadas para identificação de determinados interesses de software;

- Criação de mecanismos para construção colaborativa de catálogos de software, bem como para o controle de diferentes versões desses catálogos. Esses recursos poderiam melhorar a produtividade da equipe envolvida no processo de construção de catálogos de interesses, bem como facilitar o rastreamento das modificações realizadas por cada membro dessa equipe; e
- Criação de estratégias que auxiliem engenheiros de software a adaptarem catálogos de interesses pré-existentes as suas necessidades. Tais necessidades podem advir do desejo de se identificar um novo tipo de interesse no software, da baixa efetividade proporcionada pelo catálogo de interesses original, entre outros;
- Criação de mecanismos para integração entre a ferramenta desenvolvida neste trabalho (*ObasCId-Tool*) e os *templates* para representação de informações sobre o processo de EROA desenvolvidos pelos autores de outras abordagens. Por exemplo, uma ideia seria implementar um módulo na ferramenta *ObasCId-Tool* que permitisse a ela ler e importar os interesses especificados no *template* proposto nas abordagens *MDSoC* (Moreira *et al*., 2005) ou *EROA/XML* (Soeiro *et al*., 2006); e
- Realização de novos experimentos sobre a abordagem *ObasCId*, comparando-a com outras abordagens para EROA, e sobre a ferramenta *ObasCId-Tool*. Dentre os possíveis tipos de experimentos que podem ser realizados sobre a ferramenta *ObasCId-Tool*, destacam-se: (i) testes de performance e escalabilidade, utilizando-se documentos de requisitos de softwares de larga escala; (ii) testes para verificar os valores mais adequados de *threshold* para buscas por similaridade; e (iii) testes para verificar a efetividade da ferramenta *ObasCId-Tool*, em termos de cobertura e precisão de interesses, quando seus resultados são confrontados com o uso manual da abordagem *ObasCId*, considerando-se documentos de requisitos de diferentes tamanhos; entre outros.

REFERÊNCIAS

AGOSTINHO, S.; MOREIRA, A.; MARQUES, A.; ARAÚJO, J.; BRITO, I.; FERREIRA, R.; RAMINHOS, R.; KOVAČEVIĆ, J.; RIBEIRO, R.; CHEVALLEY, P. A Metadata-driven approach for aspect-oriented requirements analysis. In: 10th International Conference on Enterprise Information Systems. **Proceedings…** Barcelona, Spain, p. 129-136, 2008.

ALENCAR, F.; CASTRO, J.; LUCENA, M.; SANTOS, E.; SILVA, C.; ARAÚJO, J.; MOREIRA, A. Towards modular i* models, In: **ACM Symposium on Applied Computing**, p. 292-297, 2010.

ARAÚJO, J.; WHITTLE, J.; KIM, D. K. Modeling and composing scenario-based requirements with aspects. Requirements In: Engineering Conference. **Proceedings…** Washington: IEEE Computer Society, 2004.

ASSAWAMEKIN, N. Resolving Semantic Heterogeneity in Multiperspective Requirements Traceability Using Ontology Matching, In: **Journal of Convergence Information Technology**, v. 6(6), p. 340-350, 2011.

BAKKER J.; TEKINERDOĞAN, B.; AKIST, M. Characterization of Early-Aspects Approaches. In Early-Aspects: In: Aspect-Oriented Requirements Engineering and Architecture Design. **Proceedings…** Chicago, Illinois, USA, 2005.

BANIASSAD E.; CLEMENTS P.; ARAÚJO J.; MOREIRA A.; RASHID A.; TEKINERDOGAN B. Discovering early aspects. In: **IEEE Software**, v. 23(1), p. 61-70, 2006.

BANIASSAD, E.; CLARKE, S. Theme: An approach for aspect-oriented analysis and design. 26th International Conference on Software Engineering. **Proceedings…** Washington: IEEE Computer Society, 2004.

BASILI, V.; ROMBACH, H. Goal question metric paradigm. In: **Encyclopedia of Software Engineering**, v. 2, 1994.

BARGUI, F.; BEN-ABDALLAH, H.; FEKI, J. A Decision Making Ontology Building Process for Analytical Requirements Elicitation. In: International Conference on Trust, Security and Privacy in Computing and Communications. **Proceedings…** Washington, DC, USA, p. 1529-1536, 2011.

BENEVIDES, A. B. A Model-based Graphical Editor for Suppor ting the Creation, Verification and Validation of OntoUML Conceptual Models. Dissertação de Mestrado. Departamento de Informática. Universidade Federal do Espírito Santo. Vitória/ES. 2010.

BOEHM, B; IN, H. Identifying Quality-Requirement Conflicts. In: **IEEE Software** v. 13(2), p. 25–35, 1996.

BOMBONATTI, D. L. G.; MELNIKOFF, S. S. S. Survey on early aspects approaches: non-functional crosscutting concerns integration in software systems. 4th World Scientific and Engineering Academy and Society. **Proceedings...** Wisconsin, USA, p. 137-142, 2010.

BRAGA, R. T. V. Um Processo para Construção e Instanciação de Frameworks Baseados em uma Linguagem de Padrões para um Domínio Específico. 2002. Tese de Doutorado em

Ciência da Computação, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, São Paulo.

BRITO, I.; MOREIRA A. Towards a Composition Process for Aspect-Oriented Requirements. Early Aspects Workshop at AOSD. **Proceedings…** Boston, Massachusetts, USA, 2003.

CALHAU, R. F.; FALBO, R. A. A Configuration Management task ontology for semantic integration. In: Annual ACM Symposium on Applied Computing (SAC '12). **Proceedings…** New York, USA, 2012.

CHERNAK, Y. Requirements Composition Table Explained. In: 20th IEEE International Requirements Engineering Conference. **Proceedings…** Chicago, Illinois, USA, pp. 273-278, 2012.

CHITCHYAN, R.; SAMPAIO, A.; RASHID, A.; RAYSON, P. A tool suite for aspect-oriented requirements engineering. International Workshop on Early Aspects at ICSE. **Proceedings…** New York, USA, p. 19-26, 2006.

CHITCHYAN, R.; RASHID, A; SAWYER, P.; GARCIA, A.; ALARCON, M. P.; BAKKER, J.; TEKINERDOGAN, B.; CLARKE, S.; JACKSON, A. Report synthesizing state-of-the-art in aspect-oriented requirements engineering, architectures and design. Technical Report. Lancaster University: Lancaster, 259 p., 2005.

CHUNG, L.; LEITE, J. S. P. Non-Functional Requirements in Software Engineering: Springer, 441 p., 2000.

CLARKE, S.; BANIASSAD, E. Aspect-Oriented Analysis and Design: The Theme Approach. Addison-Wesley, 2005.

CYSNEIRO, L. M. Catalogues on Non-Functional Requirements. Disponível em: http://www.math.yorku.ca/~cysneiro/nfrs/nfrs.htm. Último acesso: Outubro de 2015.

DAVIS, F. D. User acceptance of information technology: system characteristics, user preceptions and behavioral impacts. In: **International Journal Man-Machine Studies**, 1993, v.38, p. 475-487.

DERMEVAL, D. *et al*. Applications of Ontologies in Requirements Engineering: A Systematic Review of the Literature. In: **Requirements Engineering**, Springer London, 2015, 1-33.

DIJKSTRA, E. W. A Discipline of Programming. Pearson Prentice Hall, 217 p., 1976.

DZUNG, D. V.; OHNISHI, A. A Verification Method of Elicited Software Requirements Using Requirements Ontology. In: Asia-Pacific Software Engineering Conference. **Proceedings…** Washington, DC, USA, p. 553-558, 2012.

FALBO, R. A. *SABiO*: Systematic Approach for Building Ontologies. 2011. Disponível em: http://www.inf.ufes.br/~falbo/files/SABiO.pdf. Último acesso em: Outubro de 2015.

FENSEL, D. Ontologies: silver bullet for knowledge management and electronic commerce. Springer-Verlag. 138 p., 2001.

GÓMEZ-PÉREZ, A.; FERNÁNDEZ-LÓPEZ, M.; CORCHO, O. Ontological Engineering. Springer Verlag, 415 p., 2004.

GRUBER, T. R. Towards Principles for the Design of Ontologies used for Knowledge Sharing. In: **International Journal of Human-Computer Studies**, Stanford, v. 43(5-6), p. 907-928, Nov. 1995.

GRUNDY, J. Aspect-Oriented Requirements Engineering for Component-based Software Systems. In: 4th IEEE International Symposium on Requirements Engineering. **Proceedings...** Limerick, Ireland, p. 84-91, 1999.

GRÜNINGER, M.; FOX, M.S. Methodology for the Design and Evaluation of Ontologies. In: Workshop on Basic Ontological Issues in Knowledge Sharing. **Proceedings…** Toronto, Canadá, 1995.

GUIZZARDI, G. Ontological Foundations for Structural Conceptual Models. Tese de doutorado. Universidade de Twente, 2005.

HEALTH WATCHER. Disponível em: http://www.cin.ufpe.br/~scbs/testbed/requirements /aore/. Último acesso em: Outubro de 2015.

HERNANDES, E. C. M. Abordagem Orientada à Informação para Análise Qualitativa com suporte de Visualização e Mineração de Texto. Tese de Doutorado, UFSCar, São Carlos, 2014.

HERNANDES, E. C. M. Um processo automatizado para tratamento de dados e conceituação de ontologias com o apoio de visualização. Dissertação de Mestrado, UFSCar, São Carlos, 2009.

HERRERA, J.; MACIA, I.; SALAS, P.; PINHO, R.; VARGAS, R.; GARCIA, A.; ARAÚJO, J.; BREITMAN, K. Revealing Crosscutting Concerns in Textual Requirements Documents: An Exploratory Study with Industry Systems. 26th Brazilian Symposium on Software Engineering. **Proceedings…** Natal, RN, p. 111-120, 2012.

HORRIGE, M.; KNUBLAUCH, H.; RECTOR, A.; STEVENS, R.; WROE, C.; JUPP, S.; MOULTON, G.; DRUMMOND, N.; BRANDT, S. A Practical Guide to Building OWL Ontologies Using Protégé 4 and CO-ODE Tools. Tutorial. University Of Manchester: Manchester, 2011.

KAIYA, H.; SAEKI, M. Ontology based requirements analysis: lightweight semantic processing approach. In: International Conference on Quality Software. **Proceedings…** p. 223-230, Melbourne, Austrália, 2005.

KELLENS, A.; MENS, K., TONELLA, P. A survey of automated code-level aspect mining techniques. In: **Transactions on Aspect-Oriented Software Development IV**, v. 4640, p. 143-162, 2007.

KITCHENHAM, B.; DYBA, T.; JORGENSEN, M. Evidence-based Software Engineering. International Conference on Software Engineering. **Proceedings…** Scotland, UK, 2004.

LIMA, J. C.; CARVALHO, C. L. Ontologias - OWL (Web Ontology Language). Relatório Técnico. Universidade Federal de Goiás: Goiânia/GO. Junho de 2005.

LÓPEZ, C.; CYSNEIRO, L. M.; ASTUDILLO, H. NDR Ontology: Sharing and Reusing NFR and Design Rationale Knowledge. In: International Workshop on Managing Requirements Knowledge. **Proceedings...** USA, p. 1-10, 2008.

MANNING, C. D; RAGHAVAN, P; SCHÜTZE, H. Introduction to Information Retrieval, Cambridge University Press. 2008.

MAFRA, S. N.; TRAVASSOS, G. H. Estudos Primários e Secundários apoiando a busca por Evidências em Engenharia de Software. Relatório Técnico. COPPE/UFRJ: Rio de Janeiro/RJ. Março, 2006.

MONTGOMERY, D. C. Design and Analysis of Experiments, 5ª ed., Wiley, 2000.

MOREIRA, A.; RASHID, A.; ARAÚJO, J. Multi-Dimensional Separation of Concerns in Requirements Engineering. In: 13th International Conference on Requirements Engineering. **Proceedings…** Paris, France, p. 285-296, 2005.

MUSSBACHER, G.; AMYOT, D.; ARAÚJO, J.; MOREIRA, A. Requirements modeling with the aspect-oriented user requirements notation (AoURN): A case study. In: **Transactions on aspect-oriented software development.** Springer-Verlag, Berlin, Heidelberg, p. 23-68, 2010.

NARDI, J. C; FALBO, R. A. Uma Ontologia de Requisitos de Software. In: IX Workshop Iberoamericano de Ingeniera de Requisitos y Ambientes de Software. **Proceedings…** La Plata, Argentina, 2006.

PARREIRA JÚNIOR, P. A.; PENTEADO, R. A. D. Domain Ontologies in the Context of Requirements Engineering: A Systematic Mapping. In: XII ACS/IEEE International Conference on Computer Systems and Applications. **Proceedings...** Marrakech, Morocco, 2015a.

PARREIRA JÚNIOR, P. A.; PENTEADO, R. A. D. Crosscutting Concerns Identification Supported by Ontologies: A Preliminary Study. In: **Lecture Notes in Business Information Processing**. Springer International Publishing, 2015b (*to be published*).

PARREIRA JÚNIOR, P. A.; PENTEADO, R. A. D. OnTheme/Doc: An Ontology-Based Approach for Crosscutting Concern Identification from Software Requirements. In: XVII International Conference on Enterprise Information Systems. **Proceedings...** Barcelona, Espanha, 2015c.

PARREIRA JÚNIOR, P. A.; PENTEADO, R. A. D. An Overview on Aspect-Oriented Requirements Engineering Area. In: **Lecture Notes in Business Information Processing**. 16ed.: Springer International Publishing, 2015d, v. 227, p. 244-264.

PARREIRA JÚNIOR, P. A.; PENTEADO, R. A. D. Aspect-Oriented Requirements Engineering: A Systematic Mapping. In: XVI International Conference on Enterprise Information Systems. **Proceedings...** Lisboa, Portugal, 2014.

PARREIRA JÚNIOR, P. A.; PENTEADO, R. A. D. Critérios para Comparação entre Abordagens para Engenharia de Requisitos Orientada a Aspectos. In: XXVII Simpósio Brasileiro de Engenharia de Software. **Proceedings...** Brasília/DF, 2013.

PETERSEN, K. *et al*. Systematic Mapping Studies in Software Engineering. In: XII International Conference on Evaluation and Assessment in Software Engineering, **Proceedings…** Bari, 2008.

RANKSNL. Disponível em: http://www.ranks.nl/stopwords. Último acesso em: Outubro de 2015.

RASHID, A.; CHITCHYAN, R., 2008. Aspect-Oriented Requirements Engineering: A roadmap. In: International Conference on Software Engineering. **Proceedings…** Leipzig, Germany, p. 35-41, 2008.

RASHID, A.; MOREIRA, A.; ARAÚJO, J. Modularisation and composition of aspectual requirements. In: 2nd International Conference on Aspect-Oriented Software Development. **Proceedings…** New York, USA, 2003.

RASHID, A.; SAWYER, P.; MOREIRA, A.; ARAÚJO, J. Early Aspects: a Model for Aspect-Oriented Requirements Engineering. In: International Conference on Requirements Engineering. **Proceedings...** Essen, Germany, 2002.

RESENDE, A. M. P. MIDAI: Um método para Identificação e Definição de Aspectos Iniciais. Tese de Doutorado. Instituto Tecnológico de Aeronáutica (ITA), 2007.

SAMPAIO, A.; GREENWOOD P.; GARCIA, A. F.; RASHID, A. A Comparative Study of Aspect-Oriented Requirements Engineering Approaches. In: First International Symposium on Empirical Software Engineering and Measurement. **Proceedings…** Madrid, Spain, p. 166-175, 2007.

SAMPAIO, A.; CHITCHYAN, R.; RASHID, A.; RAYSON, P. EA-Miner: a Tool for Automating Aspect-Oriented Requirements Identification. In: International Conference Automated Software Engineering. **Proceedings…** California, USA, p. 353-355, 2005.

SHIBAOKA, M. GOORE: Goal-Oriented and Ontology Driven Requirements Elicitation Method. In: **Advances in Conceptual Modeling – Foundations and Applications - LNCS**. v. 4802, p. 225-234, 2007.

SINGH, N.; GILL, N. S. Aspect-Oriented Requirements Engineering for Advanced Separation of Concerns: A Review. In: **International Journal of Computer Science Issues (IJCSI)**. v. 8(5). 2011.

SOEIRO E.; BRITO, I. S; MOREIRA, A. An XML-Based Language for Specification and Composition of Aspectual Concerns. In: 8th International Conference on Enterprise Information Systems. **Proceedings…** Paphos, Cyprus, 2006.

SOLIGEN, R.; and BERGHOUT, E. The goal/question/metric method– A pratical guide for quality improvement of software development. Great Britain: Cambridge, McGraw-Hill, 1999

SOMMERVILLE, Ian. Engenharia de Software. 9ª. edição. Pearson Prentice Hall, 529 p., 2011.

SPEM. Software and Systems Process Engineering Meta-Model Specification, OMG Object Management Group, OMG document number formal: 08-04-01, April 2008.

TARR, P.; OSSHER, H.; Harrison, W.; JR STANLEY, M. S. N-degrees of separation: multi-dimensional separation of concerns. In: International Conference on Software Engineering. **Proceedings…** California, USA, 1999.

USCHOLD, M.; KING, M. Towards a Methodology for Building Ontologies. In: Workshop on Basic Ontological Issues in Knowledge Sharing. **Proceedings…** Montreal, Quebec, Canadá,1995.

VERMA, K.; KASS, A. Requirements Analysis Tool: A Tool for Automatically Analyzing Software Requirements Documents. In: **The Semantic Web. LNCS**. v. 5318, p. 751-763, 2008.

VIANA, M. C. Construção da Camada de Interface Gráfica e de um Wizard para o Framework GRENJ. Dissertação de Mestrado, UFSCar, São Carlos, 2009.

ZAMBORLINI, V. C. Estudo de Alternativas de Mapeamento de Ontologias da Linguagem OntoUML para OWL: Abordagens para Representação de Informação Temporal. Dissertação de Mestrado. Departamento de Informática. Universidade Federal do Espírito Santo. Vitória/ES. 2011.

W3C. Disponível em: http://www.w3schools.com/html/html_responsive.asp. Último acesso em: Outubro de 2015.

WHITTLE J.; ARAÚJO, J. Scenario Modeling with Aspects. In: **IEEE Software**, v. 151(4), p. 157-172, 2004.

WMATRIX. Corpus Analysis and Comparison Tool. Lancaster University. Disponível em: http://ucrel.lancs.ac.uk/wmatrix/. Último acesso em: Outubro de 2015.

WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M.C.; REGNELL, B.; WESSLÉN, A. Experimentation in Software Engineering: an introduction. Springer. 249 p, 2012.

ZHENG, X.; LIU, X.; LIU, S. Use case and non-functional scenario template-based approach to identify aspects. 2nd International Conference on Computer Engineering and Applications. **Proceedings...** Bali Island, Indonesia, p. 89-93, 2010.

Apêndice A

REQUISITOS DA FERRAMENTA *OBASCID-TOOL*

A.1 Requisitos da Ferramenta *ObasCId-Tool*

O Quadro A.8.1 apresenta a lista completa de requisitos da ferramenta *ObasCId-Tool*. Estes requisitos foram elicitados, principalmente, a partir dos conceitos e relacionamentos da ontologia *O4C*, bem como das atividades e artefatos da abordagem *ObasCId*. Tais requisitos são descritos de acordo com o modelo proposto para a abordagem *ObasCId* (Capítulo 4).

Quadro A.8.1. Documento de requisitos da ferramenta *ObasCId-Tool*.

Nome do Requisito (Tipo + ID): Descrição do Requisito (RF – Requisito Funcional; RNF – Requisito Não Funcional)	Requisitos Relacionados
RF-01. O software deve permitir o cadastramento, alteração e exclusão de catálogos de interesses de software. Cada catálogo deve conter nome, descrição (opcional) e tipo de licença, que pode ser pública ou privada.	RNF-1, RNF-2, RNF-3
RF-02. O software deve permitir o cadastramento de pesquisadores. Cada pesquisador deve conter, obrigatoriamente, nome, email e senha. Opcionalmente, pode-se cadastrar: cidade, estado, país e nome da instituição a qual o pesquisador está vinculado.	RNF-1, RNF-3
RF-03. O software deve a alteração do perfil de pesquisadores. Todos os dados de um pesquisador, com exceção do seu *email*, podem ser atualizados.	RNF-1, RNF-2, RNF-3
RF-04. Para cada pesquisador, o software deve cadastrar automaticamente e permitir a atualização da seguinte lista de *stopwords*: of, the, what, and, from, in, at, on, a, an, for, to, de, a, o, que, e, do, da, em, um, para.	RNF-1, RNF-3
RF-05. Para cada pesquisador, o software deve cadastrar automaticamente e permitir a atualização de uma lista de configurações padrão (*default*) para entrada de dados, conforme descrito a seguir: 1. Opção padrão para o tipo de licença de um catálogo de interesses de software: privada; 2. Opção padrão para o tipo de licença de um documento de requisitos: privada; 3. Opção padrão para o tipo de interesses de software: não funcional; 4. Opção padrão para o tipo de requisito de software: funcional;	RNF-1, RNF-3

5. Opção padrão para o tipo de fonte de um interesse: catálogo; 6. Opção padrão para a prioridade de um interesse: *Not specified*; e 7. Opção padrão para o tipo de relacionamento entre interesses: dependência.	
RF-06. O software deve permitir o cadastramento, alteração e exclusão de documentos de requisitos de software. Cada documento de requisito deve conter nome, descrição (opcional) e tipo de licença, que pode ser pública ou privada.	RNF-1, RNF-2, RNF-3
RF-07. O software deve permitir o cadastramento, alteração e exclusão de interesses de software. Cada interesse deve estar vinculado a um catálogo e conter nome, descrição (opcional), tipo (funcional ou não funcional) e prioridade (baixa, média, alta ou "não especificada").	RNF-1, RNF-2, RNF-3
RF-08. O software deve permitir o cadastramento, alteração e exclusão de fontes de um interesse de software. Cada fonte deve estar vinculada a um determinado interesse e conter nome, descrição (opcional) e tipo (*stakeholder*, documento de negócios ou catálogo).	RNF-1, RNF-2, RNF-3
RF-09. O software deve permitir o cadastramento, alteração e exclusão de palavras-chave de um interesse de software. Cada palavra-chave deve estar vinculada a um determinado interesse, conter sua descrição e não estar na lista de *stopwords* cadastradas para o pesquisador.	RNF-1, RNF-2, RNF-3
RF-10. O software deve permitir a importação/exportação das palavras-chave de um interesse por meio de arquivos de texto.	RNF-1, RNF-2, RNF-3
RF-11. O software deve permitir o cadastramento, alteração e exclusão de relacionamentos entre interesses de software. Cada relacionamento deve estar vinculado a um catálogo de interesses e conter tipo (dependência, contribuição negativa ou contribuição positiva), um interesse fonte e um interesse alvo.	RNF-1, RNF-2, RNF-3
RF-12. O software deve permitir o cadastramento, alteração e exclusão de requisitos de software. Cada requisito deve estar vinculado a um documento de requisitos e conter, obrigatoriamente, nome, descrição e tipo (funcional ou não funcional).	RNF-1, RNF-2, RNF-3
RF-13. O software deve permitir o cadastramento, alteração e exclusão de dependências entre requisitos de software. Cada dependência deve estar vinculada a um documento de requisitos e conter tipo um requisito fonte (que depende de outro requisito) e um requisito alvo.	RNF-1, RNF-2, RNF-3
RF-14. O software deve permitir o cadastramento, alteração, exclusão de unidades de identificação. Cada unidade de identificação deve estar vinculada a um documento de requisitos, conter um nome único e o catálogo a ser utilizado na identificação e classificação de interesses de software.	RNF-1, RNF-2, RNF-3
RF-15. O software deve permitir a geração de um relatório geral de um catálogo de interesses de software, o qual deve conter: o nome, a descrição e os interesses do catálogo, juntamente com as palavras-chave, fontes e prioridade dos mesmos.	RNF-1, RNF-2, RNF-3
RF-16. O software deve permitir a geração de um relatório de relacionamentos entre interesses de um catálogo, o qual deve conter: uma matriz de dependência e uma matriz de contribuição entre esses interesses.	
RF-17. O software deve permitir a geração de relatório geral de um documento de requisitos, o qual deve conter: o nome, a descrição e os requisitos do documento, juntamente com seu tipo e suas dependências com relação a outros requisitos.	RNF-1, RNF-2, RNF-3
RF-18. O software deve permitir a identificação de interesses de software, a partir de uma unidade de identificação. Como resultado, deve-se gerar uma lista de requisitos e interesses identificados para os mesmos, destacando-se os interesses principais de cada requisito.	RNF-1, RNF-2, RNF-3
RF-19. O software deve permitir a filtragem dos requisitos da lista de requisitos e interesses identificados pelo nome do "interesse principal" do "interesse transversal".	RNF-1, RNF-2, RNF-3
RF-20. O software deve gerar, caso necessário, uma lista de ocorrências a respeito da identificação de interesses do software, conforme definido na abordagem *ObasCId*.	RNF-1, RNF-2

RF-21. O software deve permitir que seja informado qual é o interesse principal de um determinado requisito. Para isso, deve-se ter executado uma unidade de identificação anteriormente.	RNF-1, RNF-2, RNF-3
RF-22. O software deve permitir a classificação dos interesses identificados em um documento de requisitos. Para isso, deve-se ter executado uma unidade de identificação anteriormente, bem como escolhido o interesse principal de cada requisito. Como resultado, deve-se gerar uma matriz de entrelaçamentos existentes entre os interesses identificados, conforme definido na abordagem *ObasCId*.	RNF-1, RNF-2
RF-23. O software deve permitir consultas ao repositório de catálogos de interesses de software. Para isso, deve-se informar um trecho do nome ou da descrição do catálogo a ser pesquisado. Apenas catálogos públicos podem ser apresentados como resultados de uma busca.	RNF-1, RNF-3
RF-24. O software deve permitir consultas ao repositório de documentos de requisitos. Para isso, deve-se informar um trecho do nome ou da descrição do documento de requisitos a ser pesquisado. Apenas documentos de requisitos públicos podem ser apresentados como resultados de uma busca.	RNF-1, RNF-3
RF-25. O software deve permitir a importação de catálogos de interesses de software desenvolvidos por outros. Para isso, deve-se informar um novo nome para o catálogo, sua descrição (opcional) e tipo de licença. Apenas catálogos públicos podem ser importados.	RNF-1, RNF-2, RNF-3
RF-26. O software deve permitir a importação de documentos de requisitos desenvolvidos por outros. Para isso, deve-se informar um novo nome para o documento de requisitos, sua descrição (opcional) e tipo de licença. Apenas documentos de requisitos públicos podem ser importados.	RNF-1, RNF-2, RNF-3
RF-27. O software deve permitir a união (*merge*) de dois catálogos de interesses de software. Todos os interesses e relacionamentos dos dois catálogos serão intercalados. Para isso, deve-se informar um novo nome para o catálogo de interesse de software, sua descrição (opcional) e tipo de licença. Caso haja nomes de interesses duplicados nos dois catálogos, um novo nome deve ser gerado automaticamente.	RNF-1, RNF-2, RNF-3
RNF-1. A interface de todas as funções do software deve ser responsiva, de forma que os elementos da mesma se adaptem a dispositivos com telas menores (tais como, *smartphones* e *tablets*) com no mínimo 5 polegadas.	-
RNF-2. Para utilizar as funções do software, o usuário deve estar autenticado por meio de seu email e senha.	-
RNF-3. O software deve ser de fácil utilização, permitindo que seus usuários, tendo passado por um treinamento de 40 (quarenta) minutos, consigam executar corretamente a maioria de suas funções.	-

Apêndice B

MATERIAIS UTILIZADOS NOS ESTUDOS EXPERIMENTAIS

B.1 Materiais utilizados no Estudo Experimental I

Formulário para Caracterização do Perfil dos Participantes do Estudo Experimental I

Nome:

1. Já trabalhou com alguma abordagem para Engenharia de Requisitos Orientada a Aspectos? ( ) Não Sim ( ). Qual(is)?

2. Já trabalhou com identificação de interesses de software em nível de código?
( ) Não Sim ( ). Por quanto tempo?

3. Como você classifica seu conhecimento sobre Engenharia de Requisitos?
( ) Baixo: nunca ouvi falar em Engenharia de Requisitos antes desse estudo experimental
( ) Intermediário: meu contato com a Engenharia de Requisitos se deu por meio de disciplinas de graduação e pós-graduação em Engenharia de Software.
( ) Avançado: além de ter estudado na graduação/pós, já trabalhei com engenharia de requisitos de forma ativa, seja na área acadêmica ou no mercado de trabalho.

Documento de Requisitos do Software *Health Watcher*

Este documento especifica os requisitos do sistema da Secretaria de Saúde da Prefeitura da Cidade do Recife, denominado *Health Watcher*. Ele fornece aos desenvolvedores as informações necessárias para o projeto e implementação, assim como para a realização dos testes e homologação do sistema.

RF01

Tem como propósito possibilitar as consultas para o cidadão. O cidadão poderá solicitar: i) quais unidades de saúde atendem determinada especialidade; ii) quais são as especialidades de uma unidade de saúde; iii) informações sobre a queixa feita pelo cidadão, como especificação da queixa, situação (ABERTA, SUSPENSA ou FECHADA), parecer técnico, data e hora do parecer e funcionário que realizou o parecer; e iv) informações sobre as doenças, tais como descrição da doença, quais são os sintomas, qual é a forma de manifestação, qual é o tempo de duração e os locais de referência para tratamento da doença.

RF02

Tem como propósito o registro de queixas. As queixas podem ser:

- Queixa Animal: casos de apreensão de animais, controle de vetores e animais sinantrópicos (roedores, escorpiões, morcegos, etc.), doenças associadas ao pernilongo (dengue, filariose), maus tratos com animais.
- Queixa Alimentar: casos de suspeita por ingestão de alimentos estragados;
- Queixa Diversa: casos relacionados a diversos motivos, motivos estes que não têm ligações com as queixas citadas anteriormente (restaurante suspeito quanto à higiene, fossas a céu aberto, carros pipas de procedimento suspeito, etc).

Os três tipos de queixa têm as seguintes informações em comum:

- Dados da queixa: descrição (obrigatório) e observações (opcional);
- Dados do reclamante: nome, rua, complemento, bairro, cidade, estado, CEP, número do telefone e email. Todas estas informações são opcionais;
- Situação da queixa (obrigatório), que pode ser: ABERTA, SUSPENSA ou FECHADA. No registro da queixa a sua situação deve ser ABERTA;
- O sistema deve registrar a data de registro da queixa.

Além destas informações, cada queixa tem suas informações específicas. São elas:

- Queixa Animal: tipo de animal (obrigatório), quantidade de animais (obrigatório) e data do incômodo (obrigatório); rua, complemento, bairro, cidade, estado, CEP e número do telefone do local de ocorrência (todas estas informações são opcionais).
- Queixa Alimentar: nome da vítima (obrigatório); rua, complemento, bairro, cidade, estado, CEP e número do telefone da vítima (todas opcionais); quantidade de comensais (pessoas que comeram a comida), quantidade de doentes, número de pessoas internadas e número de óbitos (todos obrigatórios); local em que os pacientes foram atendidos e refeição suspeita (todas opcionais).
- Queixa Diversa: idade (obrigatório), escolaridade (opcional) e ocupação (opcional); rua, complemento, bairro, cidade, estado, CEP e número do telefone

do local mais próximo da ocorrência da queixa (todas estas informações são opcionais).

RF03

Tem como propósito permitir o acesso do funcionário a operações restritas no sistema *Health Watcher*.

RF04

Tem como propósito o cadastramento das tabelas do sistema. Para este requisito estão previstas as operações de inclusão, alteração, exclusão, consulta e impressão. O funcionário deve estar logado no sistema. As tabelas são as seguintes:

- Unidade de saúde (código da unidade, descrição da unidade).
- Especialidade (código e descrição).
- Unidade de saúde / Especialidade (unidade de saúde e especialidade).
- Funcionário (login, nome e senha).
- Tipo de doença (código, nome, descrição, manifestação e duração).
- Sintoma (código e descrição).
- Tipo de doença / Sintoma (tipo de doença e sintoma).

RF05

Tem como propósito realizar a atualização do andamento de uma queixa. A queixa deve estar cadastrada e com a situação ABERTA e funcionário logado no sistema.

RF06

O sistema deve ter uma interface de fácil utilização, visto que o sistema pode ser utilizado por qualquer pessoa que tem acesso a Internet. O sistema deve ter um HELP on-line para ser consultado por qualquer pessoa que acesse o sistema.

RF07

O sistema deve estar disponível 24 horas por dia durante os 7 dias da semana. Por não ser um sistema crítico, o sistema poderá ficar fora do ar até que seja corrigida alguma falha que possa ocorrer.

RF08

O sistema deve prover acesso a 20 usuários simultaneamente. O tempo de resposta não deve ultrapassar 5 segundos por acesso.

RF09

O sistema deve utilizar algum protocolo de segurança para envio de dados pela Internet. Para ter acesso aos recursos de registro das queixas, o usuário deve estar habilitado pelo controle de acesso ao sistema.

RF10

O sistema deve ser desenvolvido dentro dos padrões estabelecidos pela Emprel, responsável pelas normas de padronização de sistemas da Prefeitura da Cidade do Recife.

Documento de Requisitos do Software LocaDVD

RF01

Tem como propósito o gerenciamento de transações. Uma transação é uma ação que transfere a posse de um ou mais recursos para um destino. Transação possui os atributos número, data e desconto. O valor final é calculado com base nos itens da transação e no desconto. Há três tipos de transação:

- Aluguel, em que ocorre uma transferência temporária do recurso para um destino obrigatoriamente definido. Possui como atributos adicionais a data de devolução esperada e a data de devolução efetiva. Uma transação de aluguel pode ser antecedida por uma reserva;
- Reserva é uma solicitação prévia de uma transação aluguel. Uma reserva possui os seguintes atributos adicionais: número identificador, data da reserva, data de retirada, data de devolução e valor; e
- Comercialização, em que ocorre uma transferência definitiva do recurso para um destino opcionalmente definido.

RF02

Tem como propósito o gerenciamento de Itens de Transação. Um Item de Transação indica um recurso envolvido na transação e, se for o caso, uma instância desse. Possui os atributos valor e quantidade.

RF03

Tem como propósito o gerenciamento de Destinos. Destino é o solicitante de uma transação para o qual o(s) recurso(s) envolvido(s) é (são) entregue(s). Seu registro é opcional, exceto para aluguel e reserva. Destino possui os atributos número identificador e nome.

RF04

Tem como propósito o gerenciamento de Executores da Transação. Um Executor da Transação é o responsável pela execução da transação. Possui os atributos número identificador e nome.

RF05

Tem como propósito o gerenciamento de Recursos. Um Recurso é a entidade envolvida na transação e possui os atributos número identificador e nome. Além disso, um recurso pode ser de dois tipos:

- Único, recurso singular que não pode participar de duas ou mais transações simultâneas. Define um status de disponibilidade.
- Instanciável, que representa uma entidade com várias instâncias, sendo que cada Instância do Recurso possui seu próprio status e número identificador, podendo participar de uma transação independentemente das demais.

RF06

Tem como propósito o gerenciamento de Tipos de Recursos. Um Tipo de Recurso classifica um recurso com base em alguma referência. Os atributos de tipo de recurso são número identificador e nome. Um tipo de recurso também pode ter subtipos.

RF07

Podem ser registrados os Pagamentos das transações, cujos atributos são número identificador, data, valor e documento (código de boleto, por exemplo). Um pagamento pode ser de quatro tipos: Dinheiro; e Cartão, com número, bandeira, nome do titular e data de vencimento.

RF08

Devem ser criados registros de log de todas as transações realizadas no sistema.

B.2 Materiais utilizados no Estudo Experimental II

Treinamento sobre a ferramenta *ObasCId-Tool* - Exercícios

1. Cadastre um catálogo de interesses, de acordo com os dados abaixo:

Nome: Cat1 **Descrição**: -

Licença: Privada

2. Cadastre os seguintes interesses transversais para esse catálogo:

Nome: IT1 **Descrição**: -

Tipo: Não funcional **Fonte**: Catálogo

Prioridade: Média **Palavras-chave**: palavra1, palavra2

Nome: IT2 **Descrição**: -

Tipo: Funcional **Fonte**: Stakeholder

Prioridade: Alta **Palavras-chave**: palavra3, sinônimo1, sinônimo2

3. Cadastre o seguinte relacionamento entre interesses:

Fonte: IT1 **Alvo**: IT2 **Tipo**: Dependência

4. Cadastre um documento de requisitos, de acordo com os dados abaixo:

Nome: Doc1 **Descrição**: -

Licença: Privada

5. Cadastre os seguintes requisitos para esse documento:

Nome: RNF1 **Tipo**: Não funcional

Descrição: o requisito não funcional 1 possui a palavra1.

Nome: RF1 **Tipo**: Funcional

Descrição: o requisito funcional 1 não possui palavras-chave

Nome: RF2 **Tipo**: Funcional

Descrição: o requisito funcional 2 possui o sinônimo1

6. Cadastre uma unidade de identificação, de acordo com os dados abaixo:

Nome: UI1 **Catálogo**: Cat1 **Documento de requisitos**: Doc1

7. Execute essa unidade de identificação e complete a lista de requisites e interesses identificados abaixo:

RNF1: o requisito não funcional 1 possui a palavra1.

Interesses identificados: _____

RF1: o requisito funcional 1 não possui palavras-chave

Interesses identificados: _____

RF2: o requisito funcional 2 possui o sinônimo1

Interesses identificados: _____

Avaliação da ferramenta *ObasCId-Tool* – Atividade 1

1. Inclua o seguinte interesse no catálogo Cat1.

Nome: IT3 **Descrição**: -

Tipo: Não funcional **Fonte**: Catálogo

Prioridade: Média **Palavras-chave**: palavra5

2. Cadastre o seguinte relacionamento entre interesses:

Fonte: IT1 **Alvo**: IT3 **Tipo**: Dependência

3. Execute essa unidade de identificação e complete a lista de requisites e interesses identificados abaixo:

RNF1: o requisito não funcional 1 possui a palavra1.

Interesses identificados: _____

RF1: o requisito funcional 1 não possui palavras-chave

Interesses identificados: _____

RF2: o requisito funcional 2 possui o sinônimo1

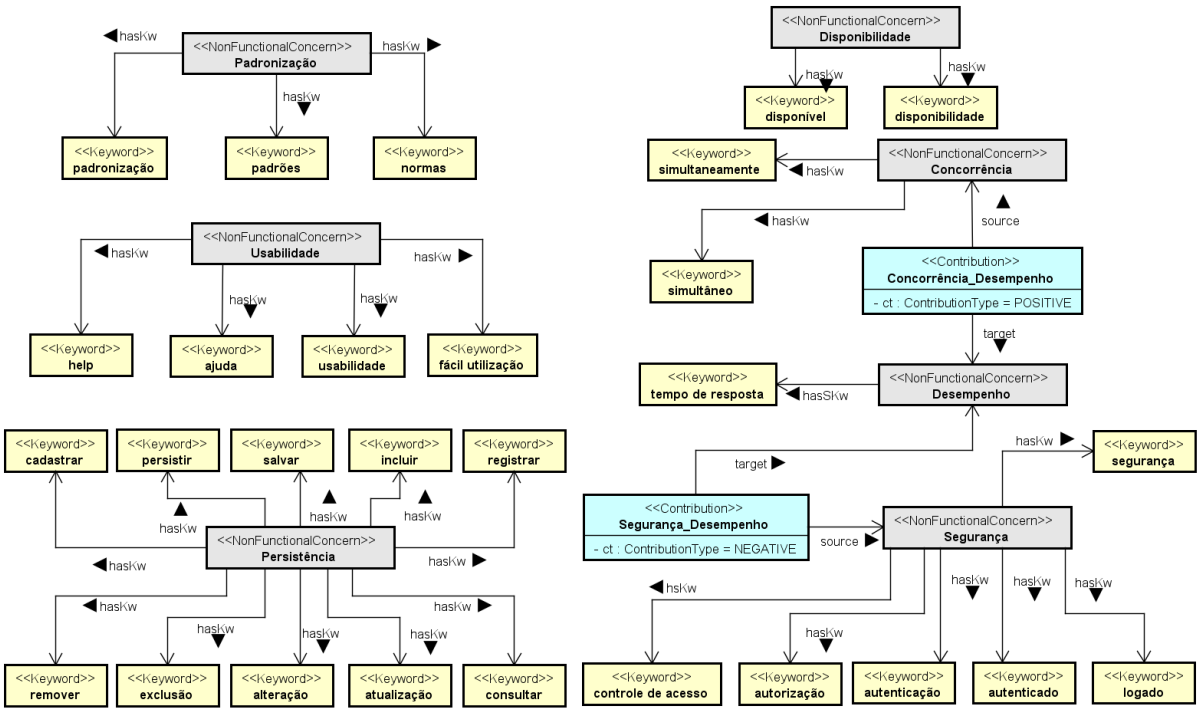
Interesses identificados: _____

4. Liste as ocorrências geradas pela ferramenta *ObasCId-Tool* no espaço abaixo:

5. Faça as modificações necessárias no catálogo de interesses/documento de requisitos criados anteriormente para que nenhuma ocorrência seja gerada pela ferramenta *ObasCId-Tool* e para que todos os interesses (IT1, IT2 e IT3) sejam corretamente identificados no documento de requisitos.

Avaliação da ferramenta *ObasCId-Tool* – Atividade 2

1. A figura abaixo representa um modelo gráfico de um catálogo de interesses. Cadastre um catálogo na ferramenta *ObasCId-Tool* correspondente a essa figura.



Avaliação da ferramenta *ObasCId-Tool* - Questionário

O questionário para avaliação da ferramenta *ObasCId-Tool*, de acordo com o modelo TAM (*Technology Acceptance Model*) (Davis, 1993), pode ser acessado em: http://goo.gl/OEi18U.