UNIVERSIDADE FEDERAL DO CEARÁ

CENTRO DE CIÊNCIAS

DEPARTAMENTO DE CIÊNCIA DA COMPUTAÇÃO

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

DOUTORADO EM CIÊNCIA DA COMPUTAÇÃO

CARLA ILANE MOREIRA BEZERRA

MEDIDAS PARA AVALIAÇÃO DA MANUTENIBILIDADE DO MODELO DE *FEATURES* DE LINHAS DE PRODUTO DE SOFTWARE TRADICIONAIS E DINÂMICAS

FORTALEZA

2016

CARLA ILANE MOREIRA BEZERRA

MEDIDAS PARA AVALIAÇÃO DA MANUTENIBILIDADE DO MODELO DE *FEATURES* DE LINHAS DE PRODUTO DE SOFTWARE TRADICIONAIS E DINÂMICAS

Tese apresentada ao Curso de Doutorado em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Ciência da Computação. Área de Concentração: Sistemas de Informação

Orientadora: Profa. Dra. Rossana Maria de Castro Andrade

Co-Orientador: Prof. Dr. José Maria da Silva Monteiro Filho

FORTALEZA

2016

Dados Internacionais de Catalogação na Publicação
Universidade Federal do Ceará
Biblioteca Universitária
Gerada automaticamente pelo módulo Catalog, mediante os dados fornecidos pelo(a) autor(a)

B469m Bezerra, Carla Ilane Moreira.
Medidas para Avaliação da Manutenibilidade do Modelo de Features de Linhas de Produto de Software Tradicionais e Dinâmicas / Carla Ilane Moreira Bezerra. – 2016.
207 f. : il. color.

Tese (doutorado) – Universidade Federal do Ceará, Centro de Ciências, Programa de Pós-Graduação em Ciência da Computação , Fortaleza, 2016.
Orientação: Profa. Dra. Rossana Maria de Castro Andrade.
Coorientação: Prof. Dr. José Maria da Silva Monteiro Filho.

1. Medidas de qualidade. 2. Modelo de features. 3. Linha de produtos de software. 4. Linha de produtos de software dinâmicas. I. Título.

CDD 005

CARLA ILANE MOREIRA BEZERRA

MEDIDAS PARA AVALIAÇÃO DA MANUTENIBILIDADE DO MODELO DE *FEATURES* DE LINHAS DE PRODUTO DE SOFTWARE TRADICIONAIS E DINÂMICAS

Tese apresentada ao Curso de Doutorado em Ciência da Computação do Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências da Universidade Federal do Ceará, como requisito parcial à obtenção do título de doutor em Ciência da Computação. Área de concentração: Sistemas de Informação.

Aprovada em: 26 / 08 / 2016.

BANCA EXAMINADORA

_____

Profª. Dra. Rossana Maria de Castro Andrade (Orientadora)
Universidade Federal do Ceará (UFC)

_____

Prof. Dr. José Maria da Silva Monteiro Filho (Coorientador)
Universidade Federal do Ceará (UFC)

_____

Profª. Dra. Ana Regina Cavalcanti da Rocha
Universidade Federal do Rio de Janeiro (UFRJ)

_____

Profª. Dra. Rosana Teresinha Vaccare Braga
Universidade de São Paulo (USP São Carlos)

_____

Prof. Dr. Adriano Bessa Albuquerque
Universidade de Fortaleza (UNIFOR)

_____

Prof. Dr. João Paulo Pordeus Gomes
Universidade Federal do Ceará (UFC)

Aos meus pais Luzia e Carlos, meu esposo Emanuel, e meu enteado Lucas!

AGRADECIMENTOS

Agradeço em primeiro lugar à Deus, sem ele nada seria possível!

Expresso a minha gratidão aos meus pais Luzia e Carlos e aos meus irmãos Rildo e Luciana pelo apoio incondicional em todos os momentos da minha vida. Obrigada aos meus pais, por sempre priorizar nossa família e pela educação dos filhos. A distância acaba nos separando em alguns momentos da vida, mas estamos perto de coração.

O meu muito obrigada ao meu esposo Emanuel por todo apoio, paciência, amor e carinho. Você sempre esteve presente nos grandes momentos da minha vida, na alegria e na tristeza, sempre me incentivando e me aconselhando. Muito obrigada por me tornar uma pessoa melhor para vida. Agradeço também ao meu enteado Lucas, por ser o orgulho da nossa família. Como seu pai sempre diz, você é o nosso melhor filho. Vocês hoje são minha base de sustentação, a melhor família que Deus podia ter me dado.

Agradeço aos meus grandes orientadores, a professora Rossana e o professor Zé Maria por terem me guiado nessa jornada tão difícil. Muito obrigada por todas as oportunidades e pela experiência transmitida ao longo desses mais de cinco anos. Vocês são profissionais e pessoas maravilhosas.

Agradeço ao GREat e a todas as pessoas que fazem parte dessa grande família. É muito bom ter o suporte de um grupo de tanta excelência.

Agradeço ao Campus UFC Quixadá e a todos os alunos, ex-alunos, professores e servidores que fazem parte desse projeto maravilhoso. Vocês fazem parte dessa conquista.

Por fim, agradeço a todas as pessoas que contribuíram de alguma forma para realização deste trabalho.

RESUMO

Linhas de Produtos de Software (LPSs) têm por objetivo a construção sistemática de software, a partir de artefatos reutilizáveis, que compartilham um conjunto de *features* comuns e variáveis e satisfazem as necessidades de um domínio particular. As Linhas de Produtos de Software Dinâmicas (LPSDs) estendem o conceito de LPSs incluindo formas de se obter variabilidade em tempo de execução. Um dos principais artefatos das LPSs e LPSDs é o modelo de *features*, o qual é responsável por representar a variabilidade de uma linha de produto. Neste cenário, avaliar a qualidade do modelo de *features* é fundamental para assegurar que erros nas fases iniciais não se propaguem para as demais fases da engenharia da linha de produto. Uma das possíveis estratégias para nortear a avaliação do modelo de *features* consiste na utilização de medidas de qualidade, que estão, em geral, relacionadas às características e subcaracterísticas de qualidade. Como a evolução de uma linha de produto afeta diretamente a complexidade e a manutenção do modelo de *features*, este trabalho tem por objetivo inicial investigar a característica de manutenibilidade. Em seguida, este trabalho visa propor soluções para avaliar o modelo de *features* utilizando medidas de manutenibilidade, uma vez que ainda existem poucos trabalhos na literatura que avaliam a manutenibilidade do modelo de *features*, utilizando medidas estruturais específicas. Para atingir esse objetivo, foi necessário construir um catálogo de medidas de qualidade de manutenibilidade, e para apoiar o uso do catálogo, foi desenvolvida uma ferramenta, que permite a coleta automática de medições pertencentes a este catálogo. Além disso, esta ferramenta auxiliou a construção dos *datasets* a serem utilizados em experimentos que avaliam o uso do catálogo da seguinte forma: um estudo exploratório que investiga o impacto da evolução dos modelos de *features* na manutenibilidade; um estudo de caso exploratório efetuado com o intuito de explorar os relacionamentos entre as medidas de manutenibilidade; e, um estudo com o propósito de agregar medidas, relacionadas à LPSDs e LPSs, por meio da utilização de lógica *fuzzy*. Os resultados desta tese sugerem que as medidas de qualidade podem ser efetivamente utilizadas para apoiar a avaliação da manutenibilidade de modelos de *features*.

Palavras-chave: Medidas de qualidade. Modelo de *features*. Linha de produtos de software. Linha de produtos de software dinâmica.

ABSTRACT

Software Product Lines (SPLs) aim the systematic building of software from reusable artifacts, which share a set of common and variables features, and satisfy the needs of a particular domain. Dynamic Software Product Lines (DSPLs) extend the concept of SPLs including ways to obtain variability at runtime. One of the main artifacts of SPLs and DSPLs is the feature model, which is responsible for representing the variability of a product line. In this scenario, assessing the quality of the feature model is essential to ensure that errors in the early stages do not spread to the other phases of the product line. One of the possible strategies to guide the evaluation of the feature model is the use of quality measures, which are, in general, related to quality characteristics and subcharacteristics. As the evolution of a product line directly affects the complexity and maintenance of the feature model, this work has the initial goal to investigate the maintainability characteristic. The aim of this work is to propose solutions to evaluate the feature model using maintainability measures, since there are still few studies in the literature that evaluate the feature model maintainability using specific structural measures. To do that, it is necessary to built a quality measures catalog and, to support the catalog usage, to develop a tool, which allows the automatic collection of measurements belonging to this catalog. Moreover, this tool helps the construction of quality measures datasets to be used in experiments that evaluate the use of the catalog, as follows: an exploratory study that investigates the impact of the feature models evolution in the maintainability of these models; an exploratory case study that explores the relationships among the maintainability measures; and, a study for aggregating measures, especially related to DSPLs and SPLs, using fuzzy logic. The results of this thesis suggest the quality measures can be effectively used to support the feature models maintainability.

Keywords: Quality measures. Feature model. Software product line. Dynamic software product line.

LISTA DE FIGURAS

Figura 1 – Metodologia de pesquisa. 22
Figura 2 – Histórico do doutorado. 23
Figura 3 – Ciclo de vida de atividades de uma LPSD (CAPILLA *et al.*, 2014a). 34
Figura 4 – Mecanismos de variabilidade dinâmica necessários para a transição de LPSs para LPSDs - adaptado de (CAPILLA; BOSCH, 2011) 38
Figura 5 – Exemplo de um modelo de features básico (adaptado de (BENAVIDES *et al.*, 2010)) 41
Figura 6 – Formas de representar a variabilidade dinâmica baseada em contexto em modelos de *features* (adaptado de (CAPILLA *et al.*, 2014b)) 42
Figura 7 – Exemplo de modelo de *features* com contexto *Smart Home* - adaptado de (CAPILLA *et al.*, 2014a) 44
Figura 8 – Evolução do modelo de *features* da LPS Mobile Media da versão 4 (Figura 8a) para versão 5 (Figura 8b). 46
Figura 9 – Notações dos modelos de *features* (adaptado de (CAPILLA *et al.*, 2013)) . . 47
Figura 10 – Estrutura do arquivo XML do modelo de *features* do repositório S.P.L.O.T. . 51
Figura 11 – Distribuição dos estudos primários por ano. 80
Figura 12 – Distribuição dos artigos selecionados por tipo de pesquisa e questão de pesquisa. 81
Figura 13 – Quantidade de problemas encontrados na revisão por pares por revisor e por tipo de problema nas medidas propostas para LPSs. 93
Figura 14 – Quantidade de problemas encontrados na revisão por pares por revisor e por tipo de problema nas medidas propostas para LPSDs. 94
Figura 15 – Visão geral da arquitetura DyMMer e S.P.L.A.R. 106
Figura 16 – Importando um modelo de *features*. 107
Figura 17 – Visualizando um modelo de *features*. 107
Figura 18 – Editando um modelo de *features*. 108
Figura 19 – Exportando os resultados das medidas extraídos a partir de um ou mais modelos de *features* para um arquivo XLS. 109
Figura 20 – Processo de geração do *dataset* MAcchiATO. 110
Figura 21 – Tipos de evolução e mudanças na análise de diferentes versões dos 16 modelos de *features*. 118

Figura 22 – Histograma e a função de distribuição de probabilidade que melhor descreve as medidas NF, NTop, NGF e ROr. 129
Figura 23 – Um exemplo de PCA (Figure 23a) para NF vs NLeaf. A Figura 23b mostra uma transformação ortogonal de NF e NLeaf. 135
Figura 24 – Seleção dos componentes principais. 136
Figura 25 – Estrutura do sistema especialista *fuzzy* (HETTIARACHCHI *et al.*, 2016). . . 150
Figura 26 – Função de pertinência para o Índice de Dinamicidade do modelo de *features* (DIMF). 155
Figura 27 – Índice de dinamicidade do modelo de *features* de *bikesharing*. 159

LISTA DE TABELAS

Tabela 1 – Recomendação de aplicação para SAT e *Binary Decision Diagrams* (BDD) (adaptado de (MENDONCA *et al.*, 2009a) 50
Tabela 2 – Características e atributos de qualidade da ISO/IEC 25010 (ISO/IEC, 2011) 55
Tabela 3 – Definição operacional da medida (adaptado de (BARCELLOS, 2009)) . . . 63
Tabela 4 – Comparação dos trabalhos relacionados com a proposta da tese de doutorado. 75
Tabela 5 – Formulário de extração de dados. 79
Tabela 6 – Resultados da busca e seleção dos estudos. 80
Tabela 7 – Resultados da busca e seleção de estudos voltados para LPSDs. 85
Tabela 8 – Medidas mensuráveis específicas para o modelo de *features* de LPSDs. . . . 86
Tabela 9 – Adaptações das medidas específicas para modelos de *features* de LPSDs. . . 87
Tabela 10 – Características de qualidade, subcaracterísticas de qualidade e medidas extraídas a partir de artigos da literatura - adaptado de (BEZERRA *et al.*, 2015). 88
Tabela 10 – Características de qualidade, subcaracterísticas de qualidade e medidas extraídas a partir de artigos da literatura - adaptado de (BEZERRA *et al.*, 2015). 89
Tabela 10 – Características de qualidade, subcaracterísticas de qualidade e medidas extraídas a partir de artigos da literatura - adaptado de (BEZERRA *et al.*, 2015). 90
Tabela 10 – Características de qualidade, subcaracterísticas de qualidade e medidas extraídas a partir de artigos da literatura - adaptado de (BEZERRA *et al.*, 2015). 91
Tabela 11 – Catálogo COfFEE - Um catálogo de medidas de qualidade para suportar a avaliação da manutenibilidade do modelo de *features*. 95
Tabela 11 – Catálogo COfFEE - Um catálogo de medidas de qualidade para suportar a avaliação da manutenibilidade do modelo de *features*. 96
Tabela 12 – Catálogo COfFEE: Fórmulas de cálculo das medidas. 98
Tabela 13 – Resultados das medidas para cada modelo de *features*. 99
Tabela 14 – Questões de pesquisa do estudo exploratório. 117
Tabela 15 – Resultados das medidas para diferentes versões dos modelos de *features*. . . 119
Tabela 16 – Correlação de *Spearman* para medidas de qualidade (parte 1). 131
Tabela 17 – Correlação de *Spearman* para medidas de qualidade (parte 2). 134
Tabela 18 – Proporção cumulativa dos PCs. 136
Tabela 19 – Componentes principais (PCs). 137
Tabela 20 – Valores das medidas para o modelo de *features* “Mobile Phone Software”. . 138

Tabela 21 – *Thresholds* definidos baseados na "regra de três-sigma". 140
Tabela 22 – *Thresholds* definidos baseados na estratégia em que 95% dos dados estão dentro dos intervalos. 140
Tabela 23 – Percentual de modelos de *features* fora dos limites definidos para os *thresholds*. 142
Tabela 24 – Medidas de tamanho do modelo de *features*. 151
Tabela 25 – Medidas de estabilidade do modelo de *features*. 152
Tabela 26 – Medidas de flexibilidade do modelo de *features*. 152
Tabela 27 – Medidas de dinamicidade do modelo de *features*. 153
Tabela 28 – Parâmetros das medidas para as funções de pertinência. 154
Tabela 29 – Regras *fuzzy* de tamanho. 155
Tabela 30 – Regras *fuzzy* de estabilidade. 156
Tabela 31 – Regras *fuzzy* de flexibilidade. 156
Tabela 32 – Regras *fuzzy* de dinamicidade. 157
Tabela 33 – Resultado da aplicação das medidas agregadas 158
Tabela 34 – Percentual dos modelos de *features* por intervalo para medidas de tamanho, estabilidade, flexibilidade e dinamicidade. 159
Tabela 35 – Definição operacional da medida número de *features* (NF). 184
Tabela 36 – Definição operacional da medida número de *features* opcionais (NO). 185
Tabela 37 – Definição operacional da medida número de *features* mandatórias (NM). . . 185
Tabela 38 – Definição operacional da medida número de *features top* (NTop). 186
Tabela 39 – Definição operacional da medida número de *features* folhas (NLeaf). 186
Tabela 40 – Definição operacional da medida profundidade máxima (DT Max). 187
Tabela 41 – Definição operacional da medida profundidade média (DT Mean). 187
Tabela 42 – Definição operacional da medida profundidade mediana (DT Median). . . . 188
Tabela 43 – Definição operacional da medida complexidade cognitiva (CogC). 188
Tabela 44 – Definição operacional da medida extensibilidade da *feature* (FEX). 189
Tabela 45 – Definição operacional da medida flexibilidade da configuração (FoC). . . . 189
Tabela 46 – Definição operacional da medida *features* dependentes cíclicas únicas (SCDF). 190
Tabela 47 – Definição operacional da medida *features* dependentes cíclicas múltiplas (MCDF). 190
Tabela 48 – Definição operacional da medida complexidade ciclomática (CyC). 191
Tabela 49 – Definição operacional da medida complexidade composta (ComC). 191

Tabela 50 – Definição operacional da medida número de agrupamento de *features* (NGF). 192
Tabela 51 – Definição operacional da medida restrições variáveis de *cross-tree* (CTCV). 192
Tabela 52 – Definição operacional da medida taxa de restrições *cross-tree* (CTCR). . . . 193
Tabela 53 – Definição operacional da medida taxa de conectividade do grafo (RCon). . . 193
Tabela 54 – Definição operacional da medida densidade do grafo (RDen). 194
Tabela 55 – Definição operacional da medida coeficiente de densidade de conectividade (CoC). 194
Tabela 56 – Definição operacional da medida *features* variáveis (NVF). 195
Tabela 57 – Definição operacional da medida *single hotspot feature* (SHoF). 195
Tabela 58 – Definição operacional da medida *multiple hotspot feature* (MHoF). 196
Tabela 59 – Definição operacional da medida *rigid nohotspot feature* (RNoF). 196
Tabela 60 – Definição operacional da medida taxa de variabilidade (RoV). 197
Tabela 61 – Definição operacional da medida número de configurações válidas (NVC). . 197
Tabela 62 – Definição operacional da medida fator de ramificação máximo (BF Max). . 198
Tabela 63 – Definição operacional da medida número de grupos Or (NGOr). 198
Tabela 64 – Definição operacional da medida número de grupos XOr (NGXOr). 199
Tabela 65 – Definição operacional da medida taxa de *features* Or (ROr). 199
Tabela 66 – Definição operacional da medida taxa de *features* XOr (RXOr). 200
Tabela 67 – Definição operacional da medida número de contextos (NC). 200
Tabela 68 – Definição operacional da medida número de *features* ativadas (NAF). 201
Tabela 69 – Definição operacional da medida número de *features* desativadas (NDF). . . 201
Tabela 70 – Definição operacional da medida número de restrições de contexto (NCC). . 202
Tabela 71 – Definição operacional da medida número de *features* de Contexto (CF). . . 202
Tabela 72 – Definição operacional da medida *features* de contextos em restrições (CFC). 203
Tabela 73 – Definição operacional da medida número de *features* ativadas por contexto (AFCA). 203
Tabela 74 – Definição operacional da medida número de *features* desativadas por contexto (DFCA). 204
Tabela 75 – Matriz de análise de correlação. 205

LISTA DE ABREVIATURAS E SIGLAS

AFFOgaTO	*dAtaset For the Feature mOdel evoluTiOn*
BDD	*Binary Decision Diagrams*
COfFEE	*CatalOg of measures for Feature modEl quality Evaluation*
CSP	*Constraint Satisfaction Problem*
DyMMer	*Dynamic feature Model tool based on Measures*
ESPREssO	*mEasures dataSet for dsPl featuRE mOdel*
FODA	*Feature-Oriented Domain Analysis*
LPS	Linha de Produto de Software
LPSD	Linha de Produto de Software Dinâmica
MAcchiATO	*MeAsures dATaset for feaTure mOdel*
PCA	Análise dos Componentes Principais
PCs	Componentes Principais
S.P.L.O.T.	*Software Product Lines Online Tools*
SAT	*Boolean Satisfiability Problem*
SMO	*Software Measurement Ontology*
SQuaRE	*Software Product Quality Requirements and Evaluation*

SUMÁRIO

1 INTRODUÇÃO 18
1.1 Contextualização 18
1.2 Hipótese e Questões de Partida 21
1.3 Objetivos e Metas 21
1.4 Metodologia de Pesquisa 22
1.5 Histórico do Doutorado 23
1.6 Organização da Tese 26
2 LINHA DE PRODUTOS DE SOFTWARE 28
2.1 Introdução 28
2.2 Engenharia de LPS 32
2.3 Variabilidade 35
2.4 Estrutura do Modelo de *Features* 39
2.5 Evolução e Manutenibilidade do Modelo de *Features* 45
2.6 Exemplo de LPSs e LPSDs 46
2.7 Notação do Modelo de *Features* 47
2.8 Propriedades de Verificação do Modelo de *Features* 48
2.9 O Repositório S.P.L.O.T. 50
2.10 Conclusões 51
3 AVALIAÇÃO DA QUALIDADE DE PRODUTO 53
3.1 Introdução 53
3.2 Modelos de Qualidade 54
3.2.1 SQuaRE 54
3.3 Medição de Software 59
3.4 Definição Operacional das Medidas 62
3.5 Avaliação de Qualidade em LPSs e LPSDs 64
3.6 Conclusões 65
4 TRABALHOS RELACIONADOS 67
4.1 Introdução 67
4.2 Abordagens para Avaliação da Qualidade do Modelo de *Features* 68
4.3 Medidas para o Modelo de *Features* em LPSs 70

4.4 Medidas para o Modelo de *Features* em LPSDs 72
4.5 Ferramentas para Avaliação da Qualidade do Modelo de *Features* 73
4.6 Comparação dos Trabalhos 74
4.7 Conclusões 75
5 IDENTIFICANDO MEDIDAS PARA AVALIAÇÃO DA QUALIDADE DO MODELO DE *FEATURES* 76
5.1 Identificando Medidas para LPSs 76
5.1.1 Método de Pesquisa 77
5.1.2 Resultados do Mapeamento 79
5.2 Identificando Medidas para LPSDs 84
5.3 Revisão por Pares 92
5.4 O Catálogo COfFEE 94
5.5 Aplicando o Catálogo COfFEE 96
5.6 Conclusões 102
6 CONSTRUINDO *DATASETS* PARA SUPORTAR A AVALIAÇÃO DE QUALIDADE DE MODELOS DE *FEATURES* 103
6.1 Introdução 103
6.2 A Ferramenta DyMMer 104
6.2.1 Arquitetura da Ferramenta DyMMer 105
6.2.2 Principais Funcionalidades da DyMMer 106
6.2.2.1 Importando um Modelo de Features *106*
6.2.2.2 Visualizando um Modelo de Features *106*
6.2.2.3 Editando um Modelo de Features *107*
6.2.2.4 Exportando Medidas *108*
6.3 MAcchiATO: Um *Dataset* de Medidas para Modelos de *Features* de LPSs 108
6.4 AFFOgaTO: Um *Dataset* de Medidas para Evoluções de Modelos de *Features* de LPSs 111
6.5 ESPREssO: Um *Dataset* de Medidas para Modelos de *Features* de LPSDs 113
6.6 Conclusões 114
7 ESTUDO EXPLORATÓRIO: EVOLUÇÃO DOS MODELOS DE *FEATURES* 115
7.1 Evolução dos modelos de *features* 115

7.2 **Planejamento do Estudo Exploratório** 116
7.3 **Execução do Estudo Exploratório** 117
7.3.1 ***QP1. O número de features tende a aumentar com a evolução do modelo de features?*** 118
7.3.2 ***QP2. O número de features filhas envolvidas em relacionamentos alternativos (XOR) ou OR tende a aumentar com a evolução do modelo de features?*** 120
7.3.3 ***QP3. A profundidade do modelo de features tende a aumentar com a evolução do modelo?*** 120
7.3.4 ***QP4. A largura do modelo de features tende a aumentar com a evolução do modelo?*** 121
7.3.5 ***QP5. A variabilidade tende a aumentar com a evolução do modelo de features?*** 121
7.3.6 ***QP6. A complexidade cresce com a evolução do modelo de features?*** . . . 122
7.4 **Discussão dos Resultados** 122
7.5 **Ameaças à Validade** 123
7.6 **Conclusões** 124
8 **ESTUDO DE CASO EXPLORATÓRIO: UTILIZAÇÃO DE MEDIDAS NA AVALIAÇÃO DA QUALIDADE DO MODELO DE *FEATURES*** . . 125
8.1 **Planejamento do Estudo de Caso** 125
8.1.1 ***Questões de Pesquisa*** 125
8.1.2 ***Seleção dos Objetivos*** 126
8.1.3 ***Procedimentos para a Coleta dos Dados*** 126
8.1.4 ***Procedimentos para Análise de Dados*** 127
8.2 **Respondendo as Questões de Pesquisa** 127
8.2.1 ***Análise dos Dados*** 127
8.2.2 ***QP1: Existem correlações entre as medidas utilizadas para avaliar a qualidade dos modelos de features em LPSs?*** 128
8.2.3 ***QP2: Como agrupar as medidas relacionadas?*** 133
8.2.4 ***QP3: Como definir thresholds para uma determinada medida?*** 138
8.2.5 ***Validação Cruzada*** 141
8.3 **Discussões** 142
8.3.1 ***Relação com a Literatura Existente*** 142

8.3.2 ***Implicações para os Pesquisadores e Engenheiros de Domínio*** 144
8.4 **Ameaças à Validade** 145
8.5 **Conclusões** 147
9 **AGREGAÇÃO DE MEDIDAS DE MANUTENIBILIDADE UTILIZANDO LÓGICA *FUZZY*** 148
9.1 **Motivação** 148
9.2 **Lógica *Fuzzy*** 149
9.3 **Agregando Medidas** 150
9.3.1 ***Seleção de Medidas*** 151
9.3.2 ***Definição da Função de Pertinência*** 152
9.3.3 ***Projeto das Regras Fuzzy*** 155
9.3.4 ***Execução da Inferência de Fuzzificação e Defuzzificação*** 157
9.4 **Avaliação do Uso das Medidas Agregadas** 158
9.5 **Ameaças à Validade** 160
9.6 **Conclusões** 161
10 **CONCLUSÕES E TRABALHOS FUTUROS** 162
10.1 **Resultados Alcançados** 162
10.2 **Hipótese e Questões de Pesquisa** 162
10.3 **Publicações** 165
10.4 **Trabalhos Futuros** 167
REFERÊNCIAS 169
APÊNDICES 184
APÊNDICE A – Procedimentos Operacionais das Medidas do Catálogo COfFEE 184
APÊNDICE B – Matriz de Análise de Correlação 205
APÊNDICE C – Formulário de Perfil dos Especialistas 206

1 INTRODUÇÃO

Neste capítulo são apresentadas a motivação e a contextualização desta pesquisa, a qual tem por finalidade investigar a avaliação da qualidade de modelos de *features* por meio da utilização de medidas. Na Seção 1.1 são discutidas a contextualização e a motivação desta tese. Na Seção 1.2 são apresentadas a hipótese e as questões de partida desta pesquisa de doutorado. Na Seção 1.3, os objetivos e as metas desta tese são expostos. Na Seção 1.4 é descrita a metodologia utilizada nesta pesquisa. Na Seção 1.5 ressalta-se o histórico da pesquisa durante a tese de doutorado. Por fim, na Seção 1.6 detalha-se a estrutura da tese, com a organização dos capítulos restantes.

1.1 Contextualização

Desde os primórdios da engenharia de software, pesquisadores e profissionais têm investigado métodos, técnicas e ferramentas que possibilitassem melhorias em custos, prazo e qualidade (ALMEIDA *et al.*, 2007). Para se alcançar estes objetivos, a ideia de reuso tem sido amplamente explorada. Recentemente, dentre as diversas estratégias propostas para promover o reuso de software, as Linhas de Produtos de Software () ou Famílias de Produto têm obtido grande popularidade. Clements P. e Northrop (2002) definem uma Linha de Produtos de Software (Linha de Produto de Software (LPS)) como uma coleção de sistemas intensivos de software que utiliza e compartilha um conjunto de características comuns, que são gerenciadas para atender às necessidades de um determinado segmento de mercado ou missão, e que se desenvolve a partir de um conjunto comum de artefatos núcleo e de uma forma pré-determinada.

O paradigma de LPS tem sido considerado, tanto pela indústria quanto pela academia, uma estratégia eficiente para lidar com as diferentes necessidades dos usuários, uma vez que torna possível personalizar um produto de software por meio da seleção das variabilidades antes mesmo da sua implantação. Todavia, uma LPS em sua forma tradicional, quando as variabilidades são resolvidas antes da implantação do produto, não é conveniente para sistemas que necessitam se adaptar em tempo de execução, tais como as aplicações sensíveis ao contexto e/ou baseadas em computação ubíqua/pervasiva (BOSCH, 2004). Essas aplicações requerem variabilidades dinâmicas, ou seja, resolvidas em tempo de execução. Neste cenário, surgiu o paradigma denominado Linhas de Produtos de Software Dinâmicas (Linha de Produto de Software Dinâmica (LPSD)) (HALLSTEINSEN *et al.*, 2008). Uma LPSD tem por finalidade

produzir software capaz de se adaptar de acordo com as necessidades de usuários e restrições de recursos em tempo de execução. As podem vincular pontos de variação quando o software é iniciado para se adaptar ao ambiente (contexto) atual, bem como durante sua operação para se adaptar às mudanças no ambiente (CAPILLA *et al.*, 2013). Desta forma, LPSDs podem apresentar características de sistemas autônomos, autoadaptáveis ou sensíveis ao contexto.

Um dos importantes ativos de uma linha de produto é o modelo de *features*[1]. Este modelo representa as *features* do domínio e a variabilidade da linha. Já as *features* descrevem as funcionalidades e as características de qualidade de um produto de software. Desta forma, Um modelo de *features* modela todos os produtos possíveis de uma LPS (BENAVIDES *et al.*, 2010). Por outro lado, uma LPSD deve ser ciente do estado de um ambiente (contexto), percebendo suas mudanças. Por este motivo, os modelos de *features* de LPSDs devem representar também a variabilidade dinâmica, ou seja, as diferentes adaptações de contexto que podem ocorrer em uma LPSD. Neste cenário, o modelo de *features* de LPSDs envolvem a definição das *features* de contexto e das restrições de contexto. Vale ressaltar que, embora os modelos de *features* sejam estudados na engenharia de domínio de uma linha, estes modelos de informação podem ser utilizados para outras áreas que vão desde a coleta dos requisitos, bem como a estrutura do modelo de dados. Daí decorre a importância dos modelos de *features* no domínio de sistemas de informação (BENAVIDES *et al.*, 2010).

A avaliação da qualidade é uma atividade essencial em uma linha de produtos, uma vez que um erro ou inconsistência em um artefato pode ser propagado para todos os seus produtos (MONTAGUD; ABRAHãO, 2009). No entanto, a avaliação da qualidade de todos os artefatos e produtos de software de uma determinada linha revela-se impraticável, tanto por razões econômicas quanto pelo esforço necessário (ETXEBERRIA; SAGARDUI, 2008a). Desta forma, uma alternativa eficiente e econômica consiste em avaliar somente os artefatos mais relevantes. Neste sentido, como o modelo de *features* é um dos principais artefatos de uma linha de produto, avaliar a sua qualidade torna-se de fundamental importância. Por outro lado, uma estratégia bastante popular para avaliar a qualidade de um produto (i.e, software) consiste na utilização de medidas. Uma medida é o mapeamento de uma entidade para um número ou um símbolo com o objetivo de caracterizar uma propriedade da entidade (ISO/IEC, 2014). Vale destacar que a avaliação da qualidade do modelo de *features* em LPSDs envolvem uma maior

[1] Neste trabalho a palavra "*feature model*" foi traduzida para "modelo de *features*", e não "modelo de características", pois como é tratado nesta Tese o conceito de características de qualidade, a tradução para modelo de características poderia confundir os conceitos.

complexidade, por ser necessário modelar e representar as diversas adaptações de contexto, as *features* de contexto e as restrições relacionadas a esses contextos.

Recentemente, diversos trabalhos têm discutido e apresentado medidas para a avaliação da qualidade do modelo de *features* em LPSs (THÖRN, 2010; BAGHERI E. E GASEVIC, 2011; MONTAGUD *et al.*, 2012; BERGER T. E GUO, 2014). Contudo, na maioria desses trabalhos, as definições das medidas apresentam diferentes problemas, tais como, por exemplo: ausência da fórmula de cálculo, falta de uma semântica clara, entre outros. Além disso, essas foram avaliadas a partir de sua aplicação em um número pequeno de modelos de *features*. Já no contexto de LPSDs, poucos trabalhos têm sido propostos com o objetivo de indicar medidas para a avaliação da qualidade do modelo de *features* (BAGHERI E. E GASEVIC, 2011; MONTAGUD *et al.*, 2012; BERGER T. E GUO, 2014). Porém, as poucas medidas propostas para LPSDs apresentam os mesmos problemas encontrados na literatura para LPSs. A maior parte dos trabalhos relacionados à avaliação da qualidade de LPSDs concentra-se na verificação e validação da variabilidade dinâmica do modelo de *features* (CAPILLA; BOSCH, 2011; MARINHO *et al.*, 2012; ALFÉREZ *et al.*, 2014a).

Uma das características de qualidade mais críticas do modelo de *features* é a manutenibilidade. Isso ocorre devido às LPSs e LPSDs se modificarem constantemente, o que gera a mudança da estrutura do modelo de *features*, impactando fortemente na sua manutenibilidade (BAGHERI E. E GASEVIC, 2011; BEZERRA *et al.*, 2013a; BERGER T. E GUO, 2014). A evolução da linha de produto afeta diretamente a complexidade e a manutenção do modelo de *features*, uma vez que há inserção, exclusão e alteração de *features* ao longo da evolução da linha (PUSSINEN, 2002; GAMEZ; FUENTES, 2011; QUINTON *et al.*, 2015; PASSOS *et al.*, 2015). De acordo com o estudo de Bagheri e Gašević (BAGHERI E. E GASEVIC, 2011), algumas medidas estruturais podem ser consideradas como uma boa forma de prever a complexidade dos modelos de *features* para avaliação da capacidade de manutenção de uma LPS. No entanto, ainda existem poucos trabalhos na literatura que avaliam a manutenibilidade do modelo de *features* utilizando medidas estruturais específicas (BAGHERI E. E GASEVIC, 2011; MONTAGUD *et al.*, 2012; BERGER T. E GUO, 2014). Desta forma, foi identificada, então, uma lacuna na área de avaliação da manutenibilidade do modelo de *features* em LPSs e LPSDs.

1.2 Hipótese e Questões de Partida

Considerando que a avaliação da qualidade dos modelos de *features* em LPSs e LPSDs é uma atividade fundamental para a identificação de problemas nas fases iniciais da engenharia de LPSs, esta Tese de doutorado procura averiguar a seguinte hipótese:

> *É possível identificar um conjunto de medidas de qualidade que possa ser utilizado para apoiar a avaliação da manutenibilidade do modelo de features.*

A partir dessa hipótese, cinco questões de partida (QP) foram levantadas:

- **QP1:** Que características e subcaracterísticas são relevantes para a avaliação da qualidade do modelo de *features*?
- **QP2:** Quais as medidas de qualidade que podem ser utilizadas para avaliar a manutenibilidade dos modelos de *features*?
- **QP3:** É possível avaliar a manutenibilidade do modelo de *features* utilizando medidas de qualidade?
- **QP4:** Como apoiar a avaliação da manutenibilidade do modelo de *features* por meio de medidas de qualidade?
- **QP5:** As medidas de qualidade existentes são suficientes para apoiar a avaliação da manutenibilidade do modelo de *features*?

1.3 Objetivos e Metas

O principal objetivo desta Tese consiste em fornecer subsídios para a avaliação da manutenibilidade do modelo de *features* por meio da utilização de medidas de qualidade.

Para atingir este objetivo, este trabalho possui as seguintes metas:

- **Meta 1:** Identificar as características e subcaracterísticas de qualidade que são relevantes para a avaliação da manutenibilidade do modelo de *features*.
- **Meta 2:** Construir um catálogo de medidas de qualidade que possa ser utilizado para avaliar a manutenibilidade dos modelos de *features*.
- **Meta 3:** Investigar técnicas de aprendizagem de máquina com a finalidade de reduzir a quantidade de medidas necessárias para avaliar a manutenibilidade de modelos de *features*.
- **Meta 4:** Realizar estudos de caso para investigar a utilização de medidas de qualidade com a finalidade de avaliar a manutenibilidade de modelos de *features*.

- **Meta 5:** Agregar medidas de qualidade com a finalidade de apoiar a avaliação da manutenibilidade do modelo de *features*.

É importante ressaltar que são considerados fora do escopo da Tese, os seguintes tópicos:

- Construir uma abordagem para avaliação da qualidade do modelo de *features*.
- Propor diretrizes para a melhoria da qualidade do modelo de *features*.

1.4 Metodologia de Pesquisa

A pesquisa realizada nesta tese de doutorado adota a metodologia que é ilustrada na Figura 1.

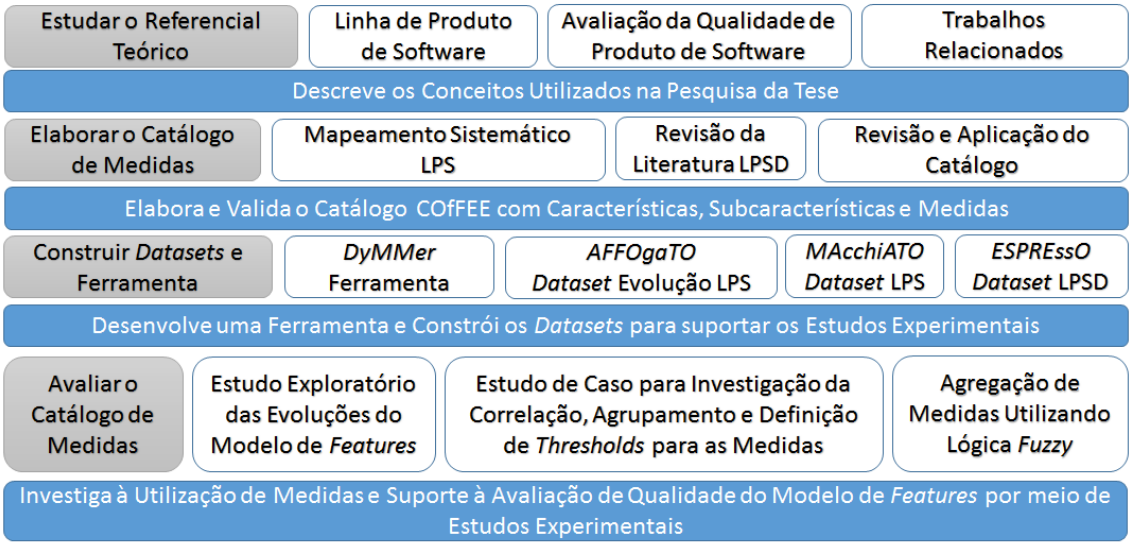


Figura 1 – Metodologia de pesquisa.

As atividades realizadas durante a execução desta pesquisa são detalhadas a seguir:

- **Estudar o Referencial Teórico:** estudo dos principais conceitos relacionados a LPS tradicional e dinâmica,modelo de *features*, avaliação da qualidade de produto e medição de software.
- **Elaborar um Catálogo de Medidas:** realização de uma ampla revisão da literatura com o objetivo de identificar, classificar e catalogar medidas que têm sido utilizadas para avaliar a qualidade de modelos de *features*.
- **Suportar a Coleta e Análise de Medidas:** construir uma ferramenta para a coleta automática das medidas catalogadas e desenvolver *datasets* para apoiar a utilização e avaliação dessas medidas.

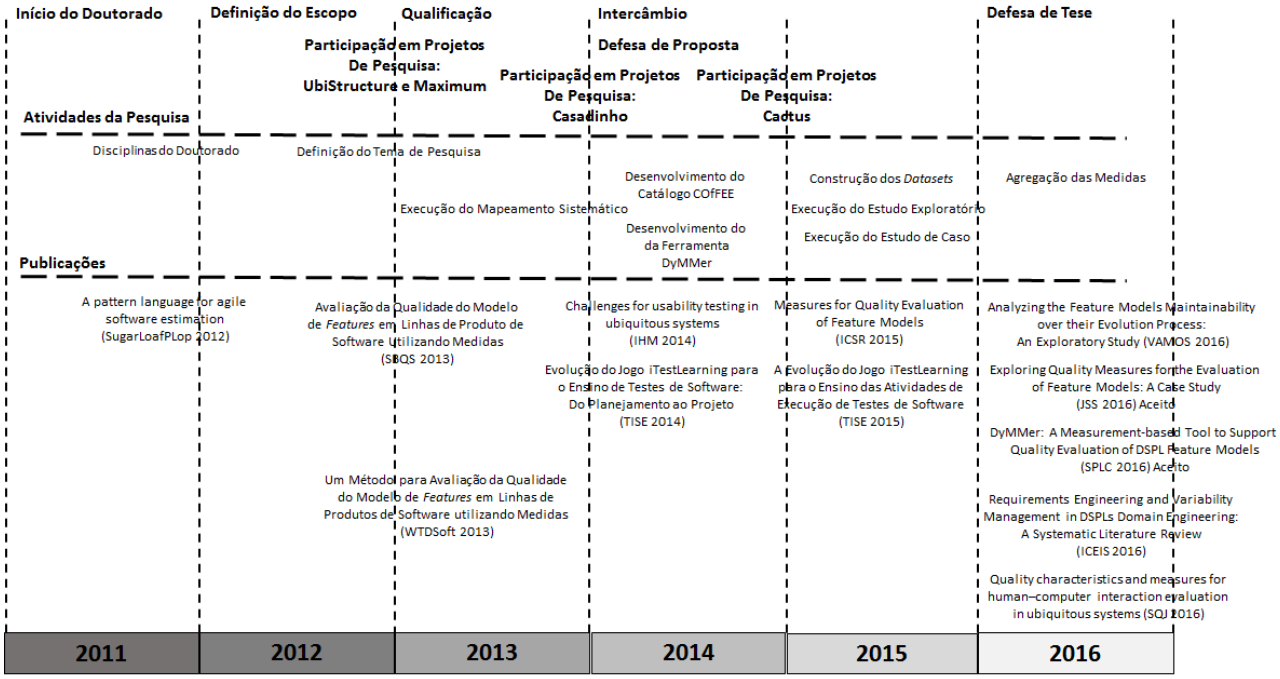


Figura 2 – Histórico do doutorado.

- **Avaliar o Catálogo de Medidas:** realização de estudos experimentais utilizando os *datasets* construídos anteriormente a fim de investigar como o catálogo de medidas concebido pode ser usado para apoiar a avaliação da qualidade de modelos de *features*.

1.5 Histórico do Doutorado

Esta Seção apresenta o histórico de investigação da pesquisa durante o doutorado. A Figura 2 ilustra as atividades executadas e os resultados obtidos em uma sequência temporal, até o final deste trabalho.

Início do Doutorado: O início do doutorado ocorreu em março de 2011. Nos 3 primeiros semestres do doutorado foram cursadas as disciplinas necessárias para obtenção dos créditos exigidos pelo programa de pós-graduação para o curso de doutorado. Como resultado da disciplina Reuso de Software foi publicado um artigo no SugarLoafPlop em 2012 (BRAGA *et al.*, 2012).

Definição do Escopo: O escopo da Tese começou a ser definido no segundo ano do doutorado. O tema de pesquisa foi inspirado no projeto Mobiline, um projeto de pesquisa do grupo GREat juntamente com a COPPE-UFRJ, e em outros trabalhos que o grupo de pesquisa GREat tinha realizado no tema de LPSs (MARINHO *et al.*, 2011; MARINHO *et al.*, 2012; MARINHO *et al.*, 2013) e Sistemas Ubíquos e Sensíveis ao Contexto (MAIA *et al.*, 2009; LIMA

et al., 2011; MAIA *et al.*, 2013). Inicialmente, o tema escolhido foi a avaliação da qualidade nas fases iniciais da LPS, mais especificamente do modelo de *features*, principal artefato da fase de requisitos da Engenharia de Domínio da LPS. Além disso, optou-se por utilizar medidas para avaliar a qualidade do modelo de *features*. Dessa forma, foi executado um estudo inicial envolvendo algumas medidas identificadas de forma não sistemática e um único modelo de *features*. Esta pesquisa inicial foi publicada no Simpósio Brasileiro de Qualidade de Software em 2013 (BEZERRA *et al.*, 2013a).

Participação em Projetos de Pesquisa: Durante o período do doutorado, a aluna participou de quatro projetos de pesquisa: UbiStructure (2012-2014), MAxIMUm (2012-2014), Casadinho (2012-Atual) e CAcTUS (2013-Atual). O projeto de pesquisa que mais se relacionou com a Tese de Doutorado foi o projeto UbiStructure. Como um resultado exclusivo do projeto MAxIMUm foi publicado um artigo no IHM 2014 (BEZERRA *et al.*, 2014).

Qualificação: A qualificação do doutorado foi defendida em outubro de 2013. A qualificação foi constituída por uma monografia intitulada de "Qualidade em Computação Sensível ao Contexto", uma monografia intitulada de "Avaliação da Qualidade em Linhas de Produtos de Software" e um artigo intitulado de "Um Método de Avaliação da Qualidade do Modelo de *Features* em Linhas de Produtos de Software Baseado em Medidas" (BEZERRA *et al.*, 2013a).

Intercâmbio: Durante o doutorado foi possível realizar visitas de curta duração às universidades COPPE-UFRJ, Universidade de *Vallenciennes* e Instituto INRIA na França. As visitas ocorreram no período de maio e junho de 2013. Os projetos Casadinho e UbiStructure financiaram as visitas possibilitando uma oportunidade de troca de conhecimento com grandes pesquisadores. Durante o período de 2013 foi também apresentada a proposta do projeto de pesquisa no Workshop de Teses e Dissertações do CBSoft (BEZERRA *et al.*, 2013b).

Defesa de Proposta: A proposta de doutorado foi defendida em dezembro de 2014. Nessa etapa foi realizado um mapeamento sistemático com a finalidade principal de identificar medidas que pudessem ser utilizadas para avaliar a qualidade do modelo de *features* de LPSs e uma revisão da literatura de forma não sistemática com o objetivo de identificar medidas específicas para LPSDs. A partir desses resultados foi concebido um catálogo de medidas de qualidade voltadas para a avaliação do modelo de *features*, o qual foi publicado em (BEZERRA *et al.*, 2015). Para apoiar a coleta automática das medidas que compõem o catálogo proposto foi desenvolvida uma ferramenta denominada *Dynamic feature Model tool based on Measures*

(DyMMer). Um artigo descrevendo a ferramenta DyMMer foi aceito recentemente no salão de ferramentas da conferência *International Systems and Software Product Line Conference* (SPLC 2016).

Resultados do Doutorado: Após a definição do tema e a realização de um estudo inicial sobre avaliação da qualidade do modelo de *features* em LPSs, foi iniciado um mapeamento sistemático para identificação de características, subcaracterísticas e medidas para avaliação de qualidade do modelo de *features* de LPSs, para elaboração do catálogo denominado *CatalOg of measures for Feature modEl quality Evaluation* (COfFEE). Esse mapeamento foi publicado na conferência *International Conference on Software Reuse* em 2015 (ICSR 2015) (BEZERRA *et al.*, 2015). A partir do catálogo COfFEE foram construídos três *datasets* (denominados *MeAsures dATaset for feaTure mOdel* (MAcchiATO), *dAtaset For the Feature mOdel evoluTiOn* (AFFOgaTO) e *mEasures dataSet for dsPl featuRE mOdel* (ESPREssO)) que deram suporte a três estudos experimentais que investigaram a utilização de medidas de qualidade na avaliação da manutenibilidade do modelo de *features*. O primeiro estudo exploratório investigou o impacto da evolução dos modelos de *features* na manutenibilidade desses modelos. Esse estudo foi publicado no *International Workshop on Variability Modelling of Software-intensive Systems* em 2016 (VAMOS 2016) (BEZERRA *et al.*, 2016). A fim de investigar a utilização do catálogo COfFEE de forma mais aprofundada, foi executado um estudo de caso que explorou a aplicação de três diferentes técnicas de análise de dados para buscar identificar correlações entre as medidas, agrupar essas medidas e definir *thresholds* para essas medidas. Um artigo descrevendo este estudo de caso foi aceito recentemente para publicação no periódico *Journal of Systems and Software* (JSS 2016) (BEZERRA *et al.*, 2016). Por fim, foi realizado um terceiro estudo com a finalidade de explorar a utilização de Lógica *Fuzzy* para agregar medidas voltadas para a avaliação da manutenibilidade de modelos de *features* de LPSDs. Durante a realização do doutorado, outros artigos que não estão diretamente relacionados ao tema desta Tese foram publicados. Dois artigos relacionados ao jogo iTestLearning foram publicados no Congresso Internacional de Informática Educativa em 2014 e 2015 (TISE) (BEZERRA *et al.*, 2014; JORGE *et al.*, 2015). Recentemente, um artigo relacionado a um mapeamento sistemático de características de qualidade e medidas para avaliação da interação humano-computador em sistemas ubíquos (CARVALHO *et al.*, 2016) foi aceito para publicação no periódico no *Software Quality Journal*. Essas publicações foram importantes para o amadurecimento da autora da Tese como pesquisadora.

1.6 Organização da Tese

Este capítulo apresentou a motivação da Tese de doutorado, a hipótese, as questões de partida, os objetivos e as metas que foram definidos, a metodologia que foi utilizada e um resumo das atividades realizadas durante o doutorado. O restante deste documento está organizado da seguinte maneira:

No Capítulo 2 discute-se os conceitos relacionados à Linha de Produtos de Software e Linha de Produtos de Software Dinâmica. São descritas as atividades de desenvolvimento das LPSDs e definições de variabilidade dinâmica. Também é apresentado o conceito de variabilidade nas LPS e LPSD. É descrita a estrutura do modelo de *features*, como ocorre a evolução do modelo de *features* e são apresentados os problemas que podem ser encontrados na verificação da qualidade do modelo de *features*. São exploradas algumas notações utilizadas para representar modelos de *features*, além de alguns exemplos de LPSDs. Por fim, é apresentado o repositório de modelos de *features* S.P.L.O.T.

No Capítulo 3 são descritos os conceitos relacionados à avaliação da qualidade do produto, modelos de qualidade e medições de software. Desta forma, busca-se facilitar o entendimento das normas de avaliação de qualidade e das técnicas para definição de medidas de qualidade. Além disso, são destacados os principais conceitos acerca da avaliação da qualidade em LPSs e LPSDs.

No Capítulo 4 são apresentados os trabalhos relacionados com esta Tese. Os trabalhos relacionados são catalogados, classificados, discutidos e comparados com a presente Tese.

O Capítulo 5 apresenta o catálogo COfFEE, composto por 40 medidas de qualidade voltadas para a avaliação do modelo de *features*. Adicionalmente, a fim de validar a utilização deste catálogo, as medidas que compõem o COfFEE foram aplicadas em 5 modelos de *features*.

O Capítulo 6 apresenta uma ferramenta denominada

O Capítulo 7 apresenta um estudo exploratório sobre o impacto da evolução dos modelos de *features* sobre a manutenibilidade. Este estudo utilizou o *dataset* MAcchiATO.

O Capítulo B apresenta um estudo de caso exploratório com a finalidade de investigar a utilização de três técnicas de análise de dados para identificar correlações entre as medidas, agrupar as medidas e definir limites superiores e inferiores para estas medidas. O *dataset* AFFOgaTO foi utilizado neste estudo.

O Capítulo 9 apresenta um estudo exploratório que buscou utilizar Lógica *Fuzzy* com a finalidade de agregar medidas de qualidades voltadas para a avaliação da qualidade do

modelo de *features* em LPSDs. Como resultado deste estudo quatro medidas agregadas foram concebidas. Essas medidas estão relacionadas aos seguintes aspectos: tamanho, estabilidade, flexibilidade e dinamicidade do modelo de *features*. Ao final, as medidas são validadas em modelos de *features* de LPSDs.

Por fim, o Capítulo 10 é dedicado às considerações finais e trabalhos futuros desta Tese de doutorado.

2 LINHA DE PRODUTOS DE SOFTWARE

Neste capítulo, discute-se um paradigma para construção sistemática de software denominado Linha de Produtos de Software (LPS). Na Seção 2.1 são apresentadas as definições de Linhas de Produtos de Software (LPSs), Linhas de Produtos de Software Dinâmicas (LPSDs) e modelos de *features*. Na Seção 2.2 é apresentada a engenharia de LPSs e LPSDs. Na Seção 2.3 são descritos os principais conceitos relacionados à variabilidade em LPSs. Na Seção 2.4 é apresentada a estrutura do modelo de *features*. Na Seção 2.5 são apresentados os conceitos relacionados à evolução dos modelos de *features*. Na Seção 2.6 são ilustrados exemplos de LPSDs. Na Seção 2.7 são apresentadas notações do modelo de *features*. Na Seção 2.8 são descritas propriedades de verificação da qualidade do modelo de *features*. Na Seção 2.9 é apresentado o repositório de modelos de *features* S.P.L.O.T. Por fim, as conclusões deste capítulo são apresentadas na Seção 2.10.

2.1 Introdução

Desde o início do desenvolvimento de software, pesquisadores e profissionais têm buscado por métodos, técnicas e ferramentas que possibilitassem melhorias em custos, prazo e qualidade (ALMEIDA *et al.*, 2007). Para se alcançar estes objetivos, a ideia de reuso tem sido amplamente explorada.

Existem diferentes definições para reuso de software. Contudo, esta Tese adota a definição de Frakes e Isoda (FRAKES *et al.*, 2005), na qual reuso de software é apresentado como "o uso de engenharia de conhecimento ou artefatos de sistemas existentes para construir novos sistemas". Essa definição é interessante pois destaca que o reuso não está relacionado apenas ao reuso de código, mas também ao reuso de outros artefatos, como documentação e do próprio conhecimento envolvido no desenvolvimento de software.

Como uma das formas de maximizar os benefícios do reuso de software surgiram as Linhas de Produtos de Software (LPSs) ou Famílias de Produto. O paradigma de Linha de Produtos de Software (LPS) tem como proposta a construção sistemática de software baseada em uma família de produtos, guiando as organizações tanto na construção de novas aplicações, a partir de artefatos reutilizáveis (desenvolvimento com reutilização), quanto na construção desses artefatos (desenvolvimento para reutilização) (LINDEN *et al.*, 2007).

A definição mais conhecida de Linha de Produto de Software foi apresentada por

Clements P. e Northrop (2002):

> "Linha de Produto de Software é um conjunto de sistemas que usam software intensivamente, compartilhando um conjunto de *features* comuns e gerenciadas, que satisfazem às necessidades de um segmento em particular de mercado ou missão, e que são desenvolvidos a partir de um conjunto comum de ativos principais e de uma forma preestabelecida."

Os ativos são artefatos produzidos na engenharia da linha de produto e são definidos como partes que são construídas para serem utilizadas por mais de um produto da linha (REINHARTZ-BERGER, 2013). A variabilidade de software é definida como a habilidade de um sistema de software ou um artefato de ser eficientemente estendido, modificado, customizado ou configurado para um contexto particular. Em LPS, uma comunalidade é uma semelhança entre os produtos que podem ser derivados de uma determinada linha, enquanto um ponto de variabilidade (ou de variação) é um ponto de diferenciação entre os produtos da linha.

Um dos ativos importantes das LPSs é o modelo de *features*, que representa as *features* do domínio e a variabilidade de uma LPS. *Features* descrevem as funcionalidades e as características de qualidade de um sistema. As *features* podem ser obrigatórias, opcionais ou variantes (ou *features* alternativas). As variantes são as *features* que compõe os pontos de variação, que capturam as variabilidades do domínio (POHL *et al.*, 2005). O modelo de *features*, proposto primeiramente por Kang *et al.* (1990), é uma representação gráfica de semelhanças e diferenças (comunalidade e variabilidade) em uma LPS na forma de uma estrutura hierárquica de um conjunto de *features* de um sistema.

Em LPSs, a criação de um produto específico consiste em um processo denominado derivação, no qual são selecionadas as variabilidades que devem estar presentes no produto em questão. Uma vez resolvidas as variabilidades, o produto pode ser implantado e executado. O paradigma de LPS proporciona a redução do tempo de desenvolvimento, diminuição de custos e a melhoria da qualidade no processo de criação de um produto de software (POHL *et al.*, 2005).

A resolução de uma variabilidade consiste na decisão sobre quais *features* serão selecionadas para resolver cada ponto de variabilidade. As variabilidades em uma LPS podem ser resolvidas em diferentes momentos, quanto mais tarde no ciclo de desenvolvimento for a resolução das variabilidades maior a dinamicidade da linha (SVAHNBERG *et al.*, 2005). Como exemplos de tempo de resolução de variabilidades, pode-se citar:

- **Derivação da arquitetura do produto**: a resolução dos pontos de variação de uma arquitetura de uma LPS é o que determina a arquitetura de um produto específico;
- **Compilação**: por exemplo, aspectos estáticos, macros e compilação condicional.;
- **Ligação**: por exemplo, DLLs (*Dynamic-link library*); e
- **Tempo de execução**: arquivos de configuração, aspectos dinâmicos, mecanismos de *plug-ins*, entre outros.

O paradigma de LPS tem sido considerado uma forma eficiente de lidar com as diferentes necessidades dos usuários, uma vez que este torna possível personalizar um produto de software por meio da seleção das variabilidades antes mesmo da sua implantação. Contudo, uma LPS em sua forma tradicional, no qual as variabilidades são resolvidas antes da implantação do produto, não é conveniente para sistemas cujas arquiteturas precisem se adaptar em tempo de execução, tais como as aplicações sensíveis ao contexto e/ou baseadas em computação ubíqua/pervasiva (BOSCH, 2004). Essas aplicações requerem variabilidades dinâmicas, ou seja, resolvidas em tempo de execução, exigindo que a arquitetura da linha tenha características de adaptação dinâmica e que mecanismos de monitoramento e controle das adaptações estejam disponíveis. Neste cenário, o processo de derivação, que tradicionalmente ocorre de forma completa em tempo de desenvolvimento, deve passar a ocorrer em tempo de execução, produzindo um produto flexível, que permite adaptações posteriores em função de demandas que são detectadas apenas com o produto em funcionamento (CAPILLA; BOSCH, 2011). Neste contexto, surgiu o paradigma denominado Linha de produtos de software dinâmicas (LPSD) (HALLSTEINSEN *et al.*, 2008).

Recentemente, LPSDs foram propostas como um caminho para promover o uso de variabilidade em tempo de execução, oferecendo uma maneira dinâmica para mudar e selecionar variantes em diferentes tempos de *binding*. Mais precisamente, uma LPSD conecta pontos de variação em tempo de execução quando o software é "inicializado" a fim de adaptar seu comportamento para o contexto atual e durante sua operação normal para se adaptar às mudanças de contexto. Assim, em ambos os casos, *features* de contexto são ativadas ou desativadas em tempo de execução devido a mudanças no contexto (FERNANDES, 2013).

LPSDs produzem software capaz de se adaptar às necessidades do usuário ou restrição de recursos (HALLSTEINSEN *et al.*, 2008). Esta adaptação é possível por meio do adiamento da resolução de variabilidades para o tempo de execução. Neste caso, a resolução se dá no momento em que o sistema é inicializado ou durante sua execução. LPSDs podem

alterar a resolução de um determinado ponto de variabilidade durante sua execução. Assim, LPSDs podem apresentar características de sistemas autônomos, autoadaptáveis ou sensíveis ao contexto. A reconfiguração de uma LPSD em tempo de execução pode ocorrer tanto por ação direta do usuário quanto por ação do próprio software em reação a algum evento percebido em seu contexto de execução (ALFÉREZ *et al.*, 2014a).

Desta forma, uma LPSD é ciente do estado de um ambiente (contexto), percebendo suas mudanças. Além de detectar mudanças, ela reage a essas, ajustando o seu comportamento em tempo de execução, a fim de fornecer serviços relevantes aos usuários, onde a relevância depende da tarefa do usuário (DEY *et al.*, 2001; BARESI *et al.*, 2012).

A definição de contexto utilizada neste trabalho é a de Dey *et al.* (2001), que define o contexto como qualquer informação que pode ser usada para caracterizar a situação de uma entidade (pessoa, lugar ou objeto) a qual é considerada relevante para a interação entre o usuário e a aplicação, incluindo o próprio usuário e a aplicação. Exemplos típicos de contexto envolvem localização, identidade, estado de pessoas e grupos, e objetos computacionais e físicos (HONG *et al.*, 2009).

Schilit *et al.* (1994) identificaram quatro categorias de contexto:

- **Contexto computacional**: pode envolver a rede de comunicação, conectividade, custo da comunicação, banda passante, recursos (impressoras, estações, entre outros.);
- **Contexto do usuário**: perfil do usuário, posição, velocidade, pessoas próximas, situação social, estado de espírito, entre outros;
- **Contexto físico**: luminosidade, nível de ruído, temperatura, umidade, entre outros; e
- **Contexto de tempo**: hora do dia, dia/mês/ano, semana, época do ano.

Chen e Kotz (CHEN *et al.*, 2008) definem contexto em função de seu efeito sobre uma aplicação:

> "Contexto é o conjunto de estados do meio ambiente que: (i) determinam o comportamento de uma aplicação (contexto ativo), ou causam a ocorrência de um evento específico da aplicação que é relevante para o usuário (contexto passivo)".

De acordo com esta definição de contexto, pode-se ter dois tipos de aplicações sensíveis ao contexto:

- Aplicação Sensível ao Contexto Ativa: uma aplicação que adapta o seu comportamento automaticamente ao contexto percebido e

- Aplicação Sensível ao Contexto Passiva: uma aplicação que mostra ao usuário informação de acordo com o contexto, ou registra o contexto em memória persistente para futura consulta.

Assim, uma LPSD tem por objetivo produzir software capaz de se adaptar de acordo com as necessidades de usuários e restrições de recursos em tempo de execução. As LPSDs podem vincular pontos de variação quando o software é iniciado para se adaptar ao ambiente atual, bem como durante a operação para se adaptar as mudanças no ambiente (CAPILLA *et al.*, 2013). Segundo Pascual *et al.* (2015), uma LPSD é um único sistema que é capaz de adaptar seu próprio comportamento em tempo de execução.

2.2 Engenharia de LPS

Segundo Clements P. e Northrop (2002), as atividades para a construção de uma LPS são: desenvolvimento do núcleo de artefatos, desenvolvimento dos produtos e gerenciamento da LPS. A atividade de desenvolvimento do núcleo de artefatos é denominada Engenharia de Domínio e a atividade de configuração dos produtos a partir desse núcleo de artefatos é denominada Engenharia de Aplicações.

A atividade de desenvolvimento do núcleo de artefatos, conhecida como Engenharia de Domínio, tem como objetivo o desenvolvimento para reutilização através da implementação de artefatos reutilizáveis, os quais irão fazer parte da linha de produto. O objetivo principal desta atividade é definir as similaridades e variabilidades da linha, bem como a arquitetura e os componentes reutilizáveis, permitindo a definição do plano de produção dos produtos. O processo de Engenharia de Domínio pode ser dividido nas seguintes atividades (LINDEN *et al.*, 2007): (i) Análise do domínio, no qual é feita a coleta de informações e conhecimento sobre uma coleção de sistemas, visando explicitar seu conjunto de similaridades e diferenças; (ii) Projeto do domínio, com o desenvolvimento da arquitetura da LPS; (iii) Implementação do domínio, onde tem-se a implementação dos artefatos reutilizáveis; (iv) Testes do domínio, que é responsável por validar os componentes implementados na atividade anterior; e (v) Gerenciamento dos produtos, que possui como objetivo definir o escopo da LPS, identificando os produtos que irão constituir a LPS como um todo. É importante salientar que os *core assets* incluem, mas não são limitados a arquitetura, documentação, especificações, componentes de software, modelos de desempenho, planos de testes, casos de testes, planos de trabalho e descrições de processos (CLEMENTS P. E NORTHROP, 2002). Desta forma, o sucesso dos produtos específicos gerados por uma LPS

depende da qualidade dos artefatos criados na engenharia de domínio (SANTOS, 2013).

A atividade de desenvolvimento do produto, conhecida como Engenharia de Aplicações, compreende o desenvolvimento com reutilização e tem por objetivo desenvolver os produtos da linha. A engenharia de aplicações compreende as seguintes atividades (LINDEN *et al.*, 2007): (i) Análise da aplicação, que tem como objetivo identificar os requisitos específicos de um determinado produto, tendo como ponto de partida o modelo de domínio da LPS; (ii) Projeto da aplicação, responsável por instanciar e adaptar a arquitetura da LPS de acordo com os requisitos identificados na atividade anterior; (iii) Implementação da aplicação; e (iv) Testes da aplicação.

A atividade de gerenciamento da LPS inclui tanto o gerenciamento técnico quanto o organizacional. Enquanto o gerenciamento organizacional é responsável pelas estruturas e recursos organizacionais (e.g., pessoal capacitado) e pelas estratégias de produção, o gerenciamento técnico é responsável por garantir que os *core assets* e os produtos estão sendo desenvolvidos conforme as atividades e o processo definido para a linha de produtos, além de coletar dados para acompanhar o progresso. Um dos artefatos mais importantes dessa atividade é o plano de adoção (CLEMENTS P. E NORTHROP, 2002), o qual descreve o estado desejado da organização e uma estratégia para alcançar esse estado. Adicionalmente, o gerenciamento técnico e organizacional também contribui para o núcleo de artefatos deixando disponível para reuso artefatos de gerenciamento como orçamentos e cronogramas usados no desenvolvimento dos produtos da linha.

Capilla et al. (CAPILLA *et al.*, 2014a) adaptaram o ciclo de vida de desenvolvimento de uma LPS, proposto inicialmente por (LINDEN *et al.*, 2007), para LPS Dinâmicas, conforme a Figura 3. Este ciclo de vida contém as propriedades desejadas de uma LPSD, tais como: mudanças em tempo de execução, reconfiguração em tempo de execução, verificação do modelo em tempo de execução e a transição entre múltiplos *binding time*. *Binding time* refere-se ao tempo de vinculação das *features* da linha em tempo de execução (HINCHEY *et al.*, 2012).

As mudanças em uma LSPD podem ocorrer em tempo de execução. Isto significa que *features* ou pontos de variação podem ser adicionados ou removidos durante a execução da linha (HALLSTEINSEN *et al.*, 2008). A reconfiguração em tempo de execução em LPSDs ocorre quando há uma mudança de contexto. Já o *rebind* corresponde a uma nova configuração quando ocorre uma adaptação de contexto (HINCHEY *et al.*, 2012).

Segundo Capilla *et al.* (2014a), o ciclo de vida proposto na Figura 3 estende o ciclo de

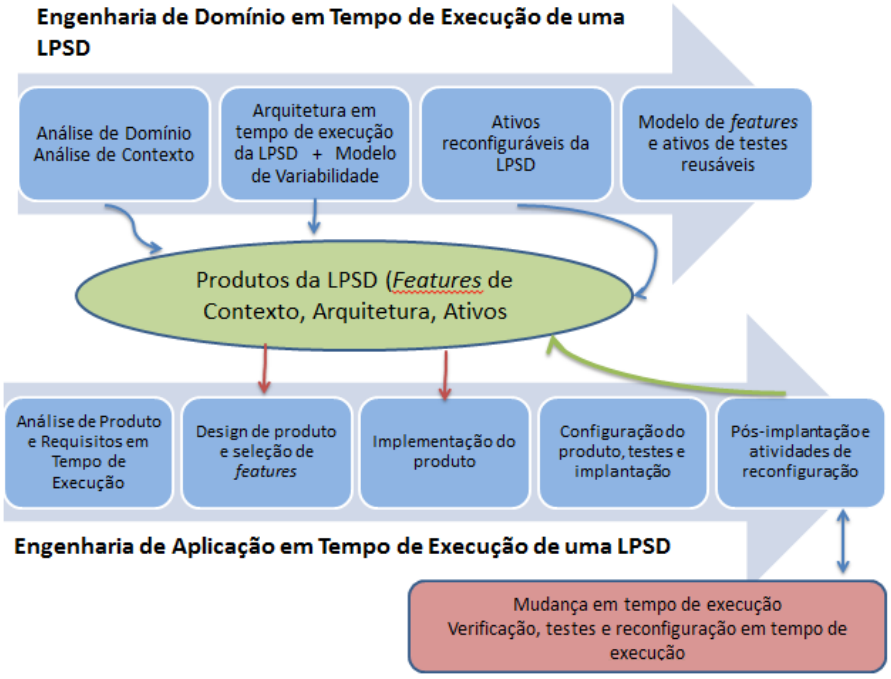


Figura 3 – Ciclo de vida de atividades de uma LPSD (CAPILLA *et al.*, 2014a).

vida tradicionalmente utilizado por LPSs com as seguintes modificações. Na fase de engenharia de domínio, a atividade tradicional de análise de domínio é estendida com uma tarefa de análise do contexto, em que se identifica não apenas os requisitos de ativos e *features*, mas também as propriedades sensíveis ao contexto que são relevantes para a LPSD. Portanto, a variabilidade e as *features* específicas à cada contexto (ou adaptação de contexto), denominadas *features* de contexto, são identificadas durante a fase inicial da modelagem. Além disso, os vários *binding times* e as opções de *rebind* podem ser definidos como propriedades no modelo de *features*. Também podem ser definidos os estados válidos para as *features* de contexto. A arquitetura do software que pode ser alterada em tempo de execução deve incluir todos os mecanismos capazes de lidar com as mudanças dinâmicas de ativos e produtos da LPSD, a ativação e desativação de *features* de forma dinâmica e possíveis modificações da estrutura da variabilidade. Esta arquitetura deve lidar com a implementação desses ativos reconfiguráveis e não configuráveis que serão testados no final da linha. Todos esses artefatos irão popular o repositório da LPSD para a fase seguinte.

Ainda segundo Capilla *et al.* (2014a), durante a fase de engenharia de aplicação na LPSD (ver Figura 3), requisitos que são em tempo de execução e os que não são, conduzem a construção de produtos da LPSD. A arquitetura da LPSD, que é definida em tempo de execução, será customizada juntamente com o modelo de *features* para que a LPSD exiba as características reconfiguráveis requeridas. Uma vez que o produto é testado, configurado e implantado, se

uma nova configuração (por exemplo, devido a uma ativação de *features*, mudanças no nível da qualidade sistema, ou a seleção de novos serviços) é requerida em tempo de execução, o ciclo de vida da LPSD suporta a tarefa de pós-implantação e reconfiguração para lidar com todas essas mudanças. Em alguns casos, a verificação, testes e monitoramento em tempo de execução, e operações de reconfiguração têm de ser realizadas depois que o produto é reimplantado e antes que ele vá para o modo operacional normal. O *feedback* da presente tarefa popula novamente o repositório da LPSD. Como resultado, essas novas atividades estendem-se à fase de engenharia de aplicação tradicional, com novas tarefas de reconfiguração que são executadas de forma dinâmica.

2.3 Variabilidade

Um dos principais conceitos em LPSs é a variabilidade, a qual pode ser definida como a possibilidade de configuração, ou ainda, como a habilidade que um sistema ou artefato de software possui de ser alterado, customizado, ou configurado para um contexto em particular (BOSCH, 2004).

A variabilidade para LPSs tradicionais é definida de forma estática, que corresponde a variabilidade que é implementada em tempo de projeto ou desenvolvimento (GALSTER *et al.*, 2014). No entanto, em sistemas sensíveis ao contexto, a variabilidade estática é muitas vezes insuficiente porque não é possível predefinir como ativar ou desativar as *features* devido à imprevisibilidade do contexto (MURGUZUR *et al.*, 2014). Portanto, para a implementação da variabilidade para esses sistemas em LPSDs, a variabilidade deve ser dinâmica.

A variabilidade de LPSDs é dinâmica, e é definida em tempo de execução. A variabilidade em tempo de execução define as escolhas do projeto de produtos visíveis aos clientes e usuários do sistema que podem ser selecionados entre as opções configuráveis disponíveis. A variabilidade em tempo de *design* é oculta para o usuário e gerenciada pelos desenvolvedores do produto, que podem decidir, por custo ou por outras razões, ativar determinadas opções de *design* em uma variante específica (CAPILLA; BOSCH, 2011).

Segundo Capilla *et al.* (2014b), muitos dos sistemas dinâmicos de software que usam propriedades contextuais são baseados em uma abordagem de linha de produto de software. No entanto, esta abordagem trata a variabilidade como um artefato de *design* estático que implementa as variações na arquitetura de software ou no modelo de *features*. Recentemente, a abordagem de LPSD foi descrita como um modo de promover o uso da variabilidade dinâmica e como um

mecanismo para gerenciar dinamicamente *features* de contexto para atender às necessidades de adaptação dinâmica (CAPILLA *et al.*, 2014a).

A diferença entre a variabilidade estática e a variabilidade dinâmica é que na variabilidade estática as variantes são tipicamente definidas durante o ciclo de desenvolvimento. Já os modelos de variabilidade dinâmica, oferecem uma forma dinâmica para alterar e selecionar variantes em diferentes tempos de execução. A variabilidade em tempo de execução permite aos usuários a seleção de diferentes alternativas e permite reconfiguração em tempo de execução, enquanto que a variabilidade estrutural permanece inalterada (BOSCH; CAPILLA, 2012).

Segundo Bosch e Capilla (2012), a variabilidade dinâmica oferece a capacidade de reconfiguração dinâmica para sistemas que requerem adaptações contínuas ou periódicas. Desta forma, a dinâmica interna do sistema pode desencadear reconfigurações em tempo de execução para ativar e desativar *features* do sistema quando necessário, e essas *features* podem se adaptar ao comportamento do sistema para diferentes cenários.

A variabilidade dinâmica baseada em contexto estende a perspectiva tradicional de modelagem de *features* para sistemas que exploram as propriedades do contexto dinamicamente. Entretanto, a crescente complexidade dos sistemas críticos que exigem maiores capacidades dinâmicas requer algo mais, além de simplesmente ativar e desativar *features*. A combinação de características de contexto e técnicas de variabilidade dinâmicas é uma excelente maneira de fornecer mecanismos automáticos de reconfiguração para sistemas críticos sensíveis ao contexto (CAPILLA *et al.*, 2014b).

A Figura 4 apresenta um conjunto de diferentes tipos de sistemas com propriedades dinâmicas, como: sistemas autoadaptativos, sistemas automáticos de computação, sistemas difusos, sistemas baseados em serviços e ecossistema de software. Além disso, Capilla e Bosch (2011) apresentam na Figura 4 propriedades da variabilidade dinâmica, conforme descritos a seguir:

- **Modelos de variabilidade extensíveis**: refere-se a capacidade de representar a variabilidade, modificando pontos de variação e novas unidades de software durante a execução e reconfiguração automática do sistema. Capilla *et al.* (2014a) consideram este aspecto como uma das formas mais simples de gerenciamento de variações em tempo de execução, em que o usuário ou o sistema, de maneira automatizada, pode ativar/desativar *features* do produto dinamicamente.
- **Verificação em tempo de execução**: refere-se a capacidade que LPSDs devem ter de au-

tomatizar a variabilidade dinâmica e verificar os modelos de *features* que se reconfiguram em tempo de execução, para manter a consistência e estabilidade do produto. Capilla *et al.* (2014a) afirmam que quando uma dada mudança ocorre no produto, adição/desativação de *features*, é necessário garantir que o sistema encontra-se consistente, e que a mudança introduzida não condicionou dependências e/ou restrições no produto, que possam desencadear a ocorrência de erros na nova configuração. Estas dependências devem ser documentadas durante o processo de requisitos através de dependências entre as *features* associadas aos requisitos do produto.

- **Implantação em tempo de execução**: refere-se ao processo de automação de implantação do produto e reconfiguração dos produtos em tempo de execução com o mínimo de interrupções. Capilla *et al.* (2014a) acreditam que a reconfiguração em tempo de execução do produto deve ser entendida como a capacidade do sistema de garantir a reconfiguração quando uma dada mudança ocorre. Pascual *et al.* (2015) indicam que o processo de implantação de um produto pode ser feito de duas formas: (i) em tempo de projeto e em (ii) tempo de execução. Na implantação em tempo de projeto a montagem de todas as partes necessárias para todas as possíveis adaptações do produto ocorrem durante a montagem do produto para a sua entrega. Já na implantação em tempo de execução a conexão das partes necessárias para as possíveis adaptações do produto ocorrem à medida que surgem as necessidades de adicionar variabilidade ao produto.
- **Rastreabilidade alinhada à arquitetura**: refere-se à capacidade de rastreabilidade das variantes do modelo de *features* estar alinhado às variantes representadas na arquitetura da LPSD.
- **Mecanismos de vinculação**: para os *binding times* de LPSs, são necessários mecanismos de vinculação para conectar as diferentes partes do produto. Isso ocorre normalmente antes da execução de fato do produto, ainda na fase de projeto (HINCHEY *et al.*, 2012).

Capilla et al. (CAPILLA *et al.*, 2014a), ainda destacam os seguintes desafios de lidar com a variabilidade dinâmica em uma LPSD:

- **Ativação e desativação de *features* de sistemas**: a ativação e desativação de opções de sistemas (ou seja, *features*) é um das formas mais simples de gerenciar as variações em tempo de execução. O usuário ou o próprio sistema pode mudar de forma autonômica *features on/off* dinamicamente. Em alguns casos, o sistema pode precisar ser reconfigurado se as *features* dos sistemas críticos são mudadas, enquanto em outros casos, o estado do

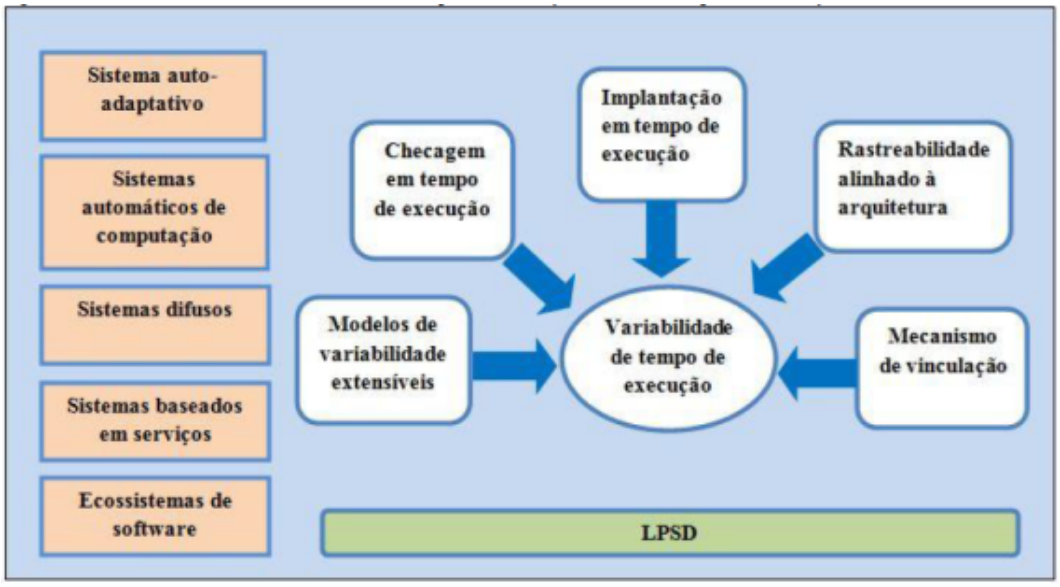


Figura 4 – Mecanismos de variabilidade dinâmica necessários para a transição de LPSs para LPSDs - adaptado de (CAPILLA; BOSCH, 2011)

sistema continua a ser o mesmo.

- **Mudanças automáticas na variabilidade estrutural**: Ao contrário das LPSs tradicionais, onde modelos de variabilidade permanecem estáticos após o tempo de projeto, LPSs dinâmicas podem modificar a estrutura da variabilidade dinâmica. Alterar variantes de forma dinâmica e automaticamente é tecnicamente viável, mas mudar pontos de variação em tempo de execução requer alguma forma de intervenção humana. Adicionar e remover variantes tem implicações diretas para as restrições e dependências entre as variantes.
- **Dependências em tempo de execução e verificação de restrições**: As mudanças no modelo de *features* realizadas em tempo de execução podem exigir, em certos casos, algum tipo de verificação em tempo de execução para detectar novas incompatibilidades de configurações dos produtos, como variantes que mudam também podem introduzir novas alterações nas dependências e regras de restrições que podem ser modificadas em tempo de execução. Além disso, a validade do modelo de *features* deve ser verificada para evitar produtos inviáveis quando, por exemplo, um novo conjunto de *features* é selecionado.
- **Reconfiguração otimizada e dinâmica**: reconfiguração dinâmica é uma propriedade fundamental para LPSDs onde produtos de software precisam ser reconfigurados em tempo de execução. Como a verificação automatizada dos modelos de *features* é feita muitas vezes em modo estático, existe uma necessidade de fornecer mecanismos de reconfiguração otimizados que podem ser executados em tempo de execução quando variantes mudam.

- **Mecanismos de reconfiguração**: Muitos sistemas autônomos requerem uma camada de adaptação para adaptar o comportamento do sistema para as novas condições. A ativação e mudança em uma *feature* do sistema muitas vezes conduz esse comportamento para adaptar o sistema para novas condições de qualidade ou a um novo ambiente.
- **Múltiplos *binding* e *rebinding***: Produtos de uma LPSD podem requerer múltiplos *binding times*. Tal como as linhas de produto tradicionais são concebidas para suportar um único tempo de ligação, na maioria dos casos, a dinamicidade de modelos de LPSD deve oferecer múltiplos *binding times* para alternar entre diferentes modos de operação.
- **Variabilidade do contexto**: Os modelos de *features* tradicionais representam as features do sistema e não as *features* de contexto utilizadas para detectar e gerenciar as condições de contexto. Portanto, a noção de variabilidade de contexto concentra-se na identificação e modelagem das propriedades de contexto que são especificamente concebidas para lidar com a diversidade do contexto que influencia o comportamento dinâmico dos sistemas.

2.4 Estrutura do Modelo de *Features*

Um modelo de *features* modela todos os produtos possíveis de uma LPS (BENAVIDES *et al.*, 2010). Kang *et al.* (1990) definem uma *feature* como uma característica do sistema visível ao usuário final. Já Gurp *et al.* (2001) caracterizam *feature* como sendo uma unidade lógica de comportamento que corresponde a um conjunto de requisitos funcionais e de qualidade.

Segundo Gurp *et al.* (2001), existe uma relação n:m entre *features* e requisitos. Isto significa que um determinado requisito pode ser aplicado a várias *features* e uma *feature* em particular pode abranger mais de um requisito.

Embora os modelos de *features* sejam estudados na engenharia de domínio de uma LPS, estes modelos de informação podem ser utilizados para outras áreas que vão desde a coleta dos requisitos, bem como a estrutura do modelo de dados. Daí a importância dos modelos de *features* no domínio de sistemas de informação (BENAVIDES *et al.*, 2010).

A estrutura de um modelo de *features* é representado como um conjunto hierarquizado de *features* composta por (BENAVIDES *et al.*, 2010):

- Relações entre uma *feature* pai e suas *features* filhas (ou *subfeatures*); e
- Restrições (ou *cross–tree*) que são tipicamente declarações de inclusão ou exclusão de *features*. Como por exemplo, se a *feature* F é incluída, então *features* A e B também devem ser incluídas (ou excluídas).

A Figura 5 apresenta um exemplo de modelo de *features* e uma representação gráfica dos conceitos básicos, baseados na notação *Feature-Oriented Domain Analysis* (FODA) (KANG *et al.*, 1990). Kang *et al.* (1990) definiram pela primeira vez o termo modelo de *features*, e propuseram a primeira notação para o modelo de *features*, do método denominado FODA (*Feature–Oriented Domain Analysis*). Os modelos de *features* básicos são baseados na notação do método FODA e possuem as seguintes relações entre as *features* (BENAVIDES *et al.*, 2010):

- **Obrigatória ou Mandatória**: A *feature* filha tem o relacionamento obrigatório com seu pai, quando a filha é incluída em todos os produtos em que sua *feature* pai aparece. Por exemplo, no modelo de *features* representado na Figura 5, o sistema Mobile Phone deve conter a *feature Calls*;
- **Opcional**: Uma *feature* filha tem um relacionamento opcional com seu pai, quando a filha pode ser opcionalmente incluída em todos os produtos em que a sua *feature* pai aparece. No modelo de *features* da Figura 5 a *feature* GPS é opcional, portanto, os produtos para Mobile Phone podem opcionalmente incluir suporte para a *feature* GPS;
- **Alternativa ou XOr**: Um conjunto de *features* filhas têm uma relação alternativa com o seu pai, quando apenas uma *feature* filha pode ser selecionada, quando a *feature* pai é parte do produto. Como por exemplo, na Figura 5, o modelo Mobile Phones pode incluir suporte para as *features basic*, *colour* ou *high resolution screen*, mas apenas para uma destas *features*; e
- **Or**: Um conjunto de *features* filhas pode ter relacionamento com seu pai, quando uma ou mais dentre elas podem ser incluídas nos produtos em que sua *feature* pai aparece. No modelo de *features* descrito na Figura 5, por exemplo, sempre que a *feature* Media é selecionada, Camera, MP3, ou ambas as *features* podem ser selecionadas.

Segundo Benavides et al. (BENAVIDES *et al.*, 2010), uma *feature* filha só pode aparecer em um produto se a *feature* pai faz parte do produto. A feature raiz é parte de todos os produtos derivados da LPS. Além das relações de parentesco entre as *features*, um modelo de *features* também pode conter restrições entre as *features*. Estes são tipicamente da forma:

- ***Require***: Se uma *feature* A requer uma *feature* B, ou seja, a inclusão de A em um produto implica a inclusão de B. Na Figura 5, Mobile Phones incluem uma *feature* Camera para poder incluir suporte para a *feature High Resolution Screen*; e
- ***Exclude***: Se uma *feature* A exclui uma *feature* B, ambas as *features* não podem ser parte do mesmo produto. Na Figura 5, GPS e *Basic Screen* são *features* incompatíveis.

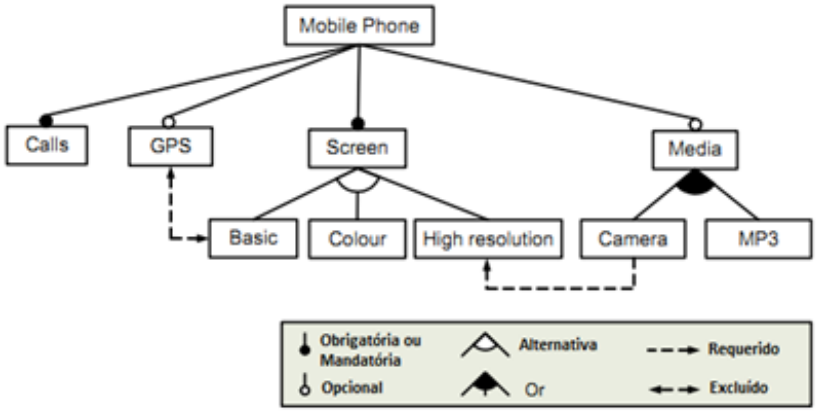


Figura 5 – Exemplo de um modelo de features básico (adaptado de (BENAVIDES *et al.*, 2010))

Alguns autores estendem os modelos de *features* na notação do método FODA (KANG *et al.*, 1990) com recursos da UML, como, por exemplo, a multiplicidade (também chamado cardinalidade). A cardinalidade é utilizada para definir o número mínimo e máximo de *features* variáveis que podem ser escolhidas a partir de um ponto de variação.

Essas relações introduzidas no modelo de *features* são definidas por Benavides et al. (BENAVIDES *et al.*, 2010), como:

- **Cardinalidade de *Feature***: é uma sequência de intervalos indicados [n..m] com n como limite inferior e m como limite superior. Estes intervalos determinam o número de instâncias da *feature* que pode ser parte de um produto. Esse relacionamento pode ser usado como uma generalização das *features* obrigatórias ([1,1]) e opcionais ([0,1]); e
- **Grupos de Cardinalidade**: é um intervalo denotado por <n..m>, onde n representa o limite inferior e m o limite superior limitando o número de *features* filhas que podem ser parte de um produto quando a *feature* pai é selecionada. Assim, uma relação alternativa ou XOr é o equivalente a <1..1> e uma relação Or é equivalente a <1..n>, sendo n o número de *features* no relacionamento.

A modelagem da variabilidade em LPSs não inclui a modelagem de informações do contexto, consideradas informações essenciais para o desenvolvimento de aplicações dinâmicas e adaptativas. Para o desenvolvimento desse tipo de aplicação, surgiram as LPSDs, que são capazes de se adaptar às variações das necessidades dos usuários e às evoluções das restrições dos recursos, conforme mencionado na Seção 2.1.

Em LPSD, além das propriedades apresentadas para modelar a variabilidade no modelo de *features*, deve-se representar também a variabilidade dinâmica baseada em contexto. Para isso, Capilla, Ortiz e Hinchey (CAPILLA *et al.*, 2014b) propõem duas estratégias:

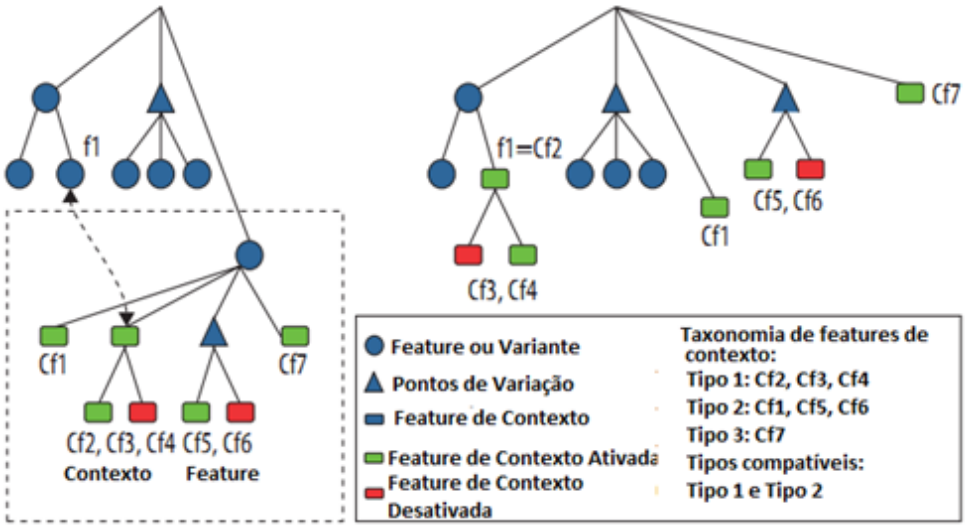


Figura 6 – Formas de representar a variabilidade dinâmica baseada em contexto em modelos de *features* (adaptado de (CAPILLA *et al.*, 2014b))

- **Dois modelos de *features* relacionados**: uma das estratégias é representada no modelo de *features* do lado esquerdo da Figura 6, o qual mostra o modelo de *features* separado das *features* de contexto. As dependências entre as *features* de contexto e as *features* comuns sobrecarregam as relações entre features, e essas dependências podem ser bidirecionais; e
- **Um único modelo de *features***: outra estratégia é representada no modelo de *features* do lado direito da Figura 6, onde *features* de contexto (as que são afetadas por adaptações de contexto) e *features* que não são de contexto são modeladas no mesmo modelo de *features*, o que reduz o número possível de dependências entre as *features*. As *features* de contexto são simplesmente rotuladas no modelo de *features* e classificadas de acordo com um conjunto de tipos pré-definidos. As *features* em verde e em vermelho são, respectivamente, aquelas *features* de contexto que são ativadas e desativadas em um determinado momento.

Além das restrições do modelo de *features*, deve-se tratar com as regras inerentes ao contexto. As regras de contexto devem ter informações de contexto inerentes ao ambiente e devem prever as dependências com o modelo de *features*.

Mens et al. (MENS *et al.*, 2016) definem uma taxonomia para abordagens para representação da variabilidade de LPSDs em sistemas sensíveis ao contexto. A taxonomia classifica as abordagens por:

- **Mecanismos de variabilidade suportados**: aborda diferentes técnicas para representar e gerenciar a variabilidade a fim de apoiar a as mudanças de contexto do sistema. Como os sistemas sensíveis ao contexto normalmente usam informações contextuais na pós-implantação e em tempo de execução, foram classificados apenas os mecanismos de

variabilidade suportados por modelos de variabilidade dinâmicos;

  - Variabilidade dinâmica fechada: suporta a ativação e desativação dinâmica de *features* que foram predefinidas;e
  - Variabilidade dinâmica aberta: suporta adição, remoção e modificação (e.g., substituição de uma *feature* opcional por uma obrigatória) de *features* dinamicamente.

- **Tipos de *features***: diferentes tipos de *features* por parte do tipos de sensibilidade ao contexto que eles suportam, variando de *features* muito estáticas que não são sensíveis ao contexto, por meio de *features* tradicionais que implementam a sensibilidade ao contexto internamente ou a sensibilidade ao contexto implementada por meio de variações de *features*, com abordagens de programação orientada ao contexto que têm uma representação de primeira classe explícita de contextos e suas correspondentes adaptações comportamentais; e
  - *Features* contextuais: variam ligeiramente com os dados do contexto específico, mas essas variações são internalizadas na *feature*;
  - *Features* não contextuais: usadas quando as variações são escolhas estáticas não desencadeadas pelo contexto dinâmico; e
  - *Features* de contexto: representam variantes contextuais específicas de uma *feature* que são apenas relevantes em contextos particulares.
- **Dependências**: dependências entre *features* desempenham um papel fundamental na modelagem e representação da variabilidade. A taxonomia centra-se sobre as alternativas existentes para estabelecer relações entre *features* de contexto e não contextuais.
  - Árvore de contexto com dois galhos separados: usa um único modelo *features* (árvore) com dois ramos separados para representar tanto tanto as *features* de contexto quanto as *features* não-contextuais;
  - Árvore de contexto com um galho: *features* de contexto e não-contextuais são inseridas em uma único modelo de *features*; e
  - Dependências declaradas pelo programador: declarar explicitamente as dependências entre os contextos em um código orientado para o contexto.

Baseado nessa taxonomia, a avaliação de qualidade nesse trabalho será utilizada a representação do modelo de *features* de LPSD com: (i) mecanismo de variabilidade dinâmica fechado; (ii) *features* não contextuais e de contexto; e (iii) dependências em um modelo de *features* único.

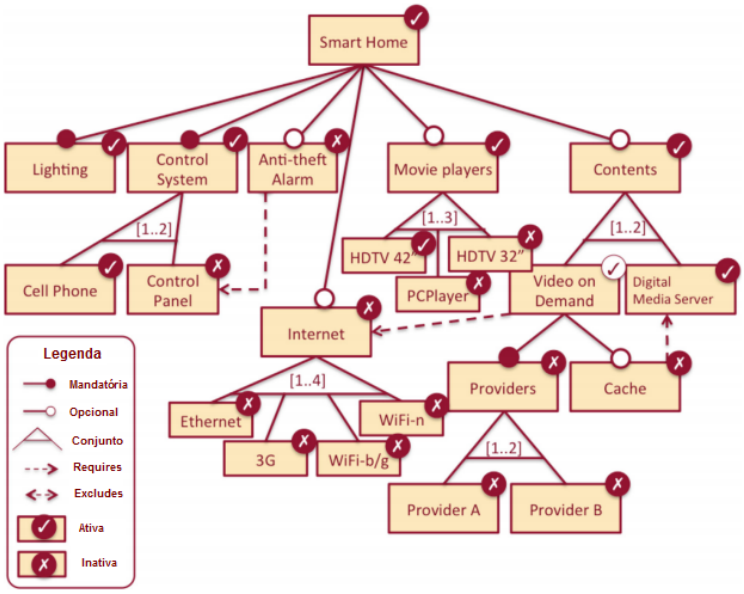


Figura 7 – Exemplo de modelo de *features* com contexto *Smart Home* - adaptado de (CAPILLA *et al.*, 2014a)

Em LPSDs, alguns trabalhos têm se utilizado da modelagem do contexto juntamente ao modelo de *features* (SALLER *et al.*, 2013; CAPILLA *et al.*, 2014a), outros utilizam o modelo de contexto separado do modelo de *features* (FERNANDES *et al.*, 2011; ALFÉREZ *et al.*, 2014a; GAMEZ *et al.*, 2015).

A Figura 7 apresenta um exemplo de modelo de *features* que representa *features* de contexto para o domínio de sistemas *smart homes* (CAPILLA *et al.*, 2014a). Em uma determinada configuração atual do modelo de *features*, são representadas as *features* ativas e inativas de um determinado contexto, assim como as restrições, relacionamentos e *features* obrigatórias e opcionais do modelo de *features*.

No exemplo da Figura 7, as funcionalidades de controle do sistema podem ser ativadas por meio do celular. Caso o usuário deseje utilizar a *feature vídeo on-demand*, a configuração atual não é válida até *vídeo on-demand* ativar a configuração de internet (que no exemplo estar desativado). Consequentemente, uma operação de explicação da configuração utiliza técnicas de diagnóstico para determinar quais *features* devem mudar seu estado para alcançar uma configuração de produto válida em tempo de execução. Nesta situação, é possível obter quatro possíveis reconfigurações, uma para cada tipo de conexão com a Internet: WiFi-b/g, Wi-fi-n, Ethernet e protocolos 3G. O critério de desempate para escolher a melhor configuração pode ser com base em atributos de qualidade exigidos pelas configurações do usuário ou do sistema desejado, como a da taxa de largura de banda disponível. Uma vez que o algoritmo determina a ideal ou correta configuração, as *features* podem mudar seus valores para os estados

válidos uma vez que as restrições e dependências tenham sido satisfeitas.

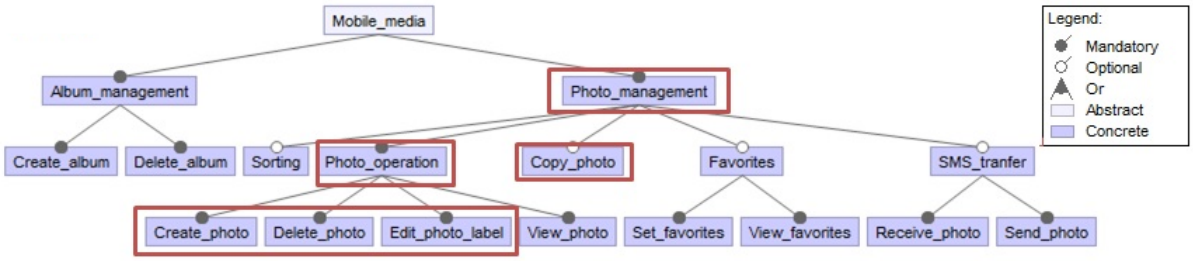
2.5 Evolução e Manutenibilidade do Modelo de *Features*

Para alavancar os benefícios oferecidos por uma LPS ao longo dos anos e manter os produtos atualizados, as LPSs frequentemente têm evoluído. Mesmo assim, é impossível prever todas as mudanças necessárias no futuro e do escopo da LPS em conformidade direta desde o início. Além disso, mudanças podem ser incluídas pela emergência de novas tecnologias no mercado. Em tais situações, é necessário fazer mudanças no modelo de *features*. Assim, modelos de *features* mudam ao longo do tempo. Neste contexto, a evolução do modelo de *features* é um importante aspecto a considerar. Na literatura, alguns trabalhos propuseram estudos sobre a evolução do modelo de *features* (LOTUFO *et al.*, 2010; GAMEZ; FUENTES, 2011; ROMERO *et al.*, 2013; ACHER *et al.*, 2014; DINTZNER *et al.*, 2015; PASSOS *et al.*, 2015).

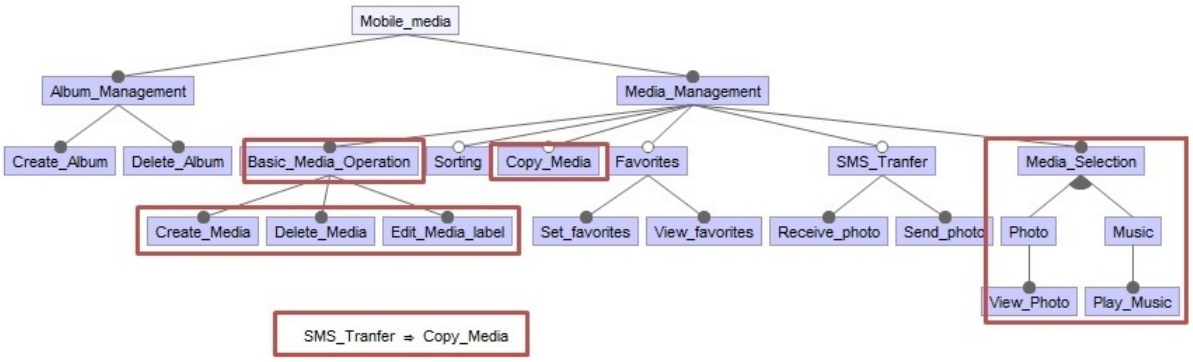
Neste trabalho, a evolução do modelo de *features* é considerada como uma sequência de modelos, tendo um modelo de *features* (versão) para cada passo de evolução. As principais mudanças que podem ocorrer em uma transição de uma versão de um modelo de *features* para outro, são: (i) adicionar ou remover *features* ("obrigatórias" ou "selecionáveis"), (ii) adicionar ou remover relacionamentos de *features* (relacionamentos "OR" ou "XOR"), (iii) adicionar ou remover restrições, e (iv) modificar a variabilidade de uma *feature* (por exemplo, uma obrigatória pode torna-se opcional) ou de um relacionamento. Em geral, uma mudança em uma *feature* específica (ou relacionamento) pode ser tratada como uma exclusão (de *features* ou relacionamentos prévios), seguida por uma inclusão (a partir de uma nova *feature* ou relacionamento). Neste trabalho, estes tipos de mudanças são denotados como "tipos de evolução".

Um exemplo de evolução de modelo de *features* é ilustrado na Figura 8, que também destaca as mudanças do modelo de *features* do *Mobile Media* da versão 4 para a versão 5. A Figura 8a ilustra a versão 4 do modelo de *features* do *Mobile Media*. A Figura 8b apresenta a versão 5 do modelo de *features* do *Mobile Media*. Os retângulos vermelhos representam as mudanças na evolução do modelo de *features* com a adição de novas *features*, alterações de *features*, adição de restrições e adição de relacionamento-OR.

Essas mudanças estruturais, que ocorrem no modelo de *features* ao longo da evolução de uma LPS ou LPSD, podem afetar a manutenibilidade do modelo acarretando uma maior complexidade e decréscimo da qualidade do modelo de *features* (OLIVEIRA *et al.*, 2017). Além disso, a manutenção e a compreensão de um modelo com milhares de *features* possui alta



(a) Versão 4



(b) Versão 5

Figura 8 – Evolução do modelo de *features* da LPS Mobile Media da versão 4 (Figura 8a) para versão 5 (Figura 8b).

complexidade devido ao tamanho e as dependências entre *features* importantes (SCHRÖTER *et al.*, 2016).

De acordo com o estudo de Bagheri e Gašević (BAGHERI E. E GASEVIC, 2011), medidas estruturais podem ser consideradas como bons exemplos para prever a complexidade dos modelos de *features* para avaliação da capacidade de manutenção de uma LPS. No entanto, ainda existem poucos trabalhos na literatura que avaliam a manutenibilidade do modelo de *features* utilizando medidas estruturais específicas (BAGHERI E. E GASEVIC, 2011; MONTAGUD *et al.*, 2012; BERGER T. E GUO, 2014). Dessa forma, são necessários mais estudos sobre o impacto da evolução do modelo de *features* em sua manutenção ao longo do ciclo de vida da linha.

2.6 Exemplo de LPSs e LPSDs

Na literatura foram identificados exemplos de LPSs e LPSDs que foram utilizadas para validar abordagens que apoiam o desenvolvimento dessas linhas de produto. Dentre os domínios que envolvem as linhas identificadas pode-se citar o domínio de *Smart Homes* (CETINA *et al.*, 2009b; BOSCH; CAPILLA, 2012; ARCEGA *et al.*, 2016), dispositivos e aplicações móveis (HALLSTEINSEN *et al.*, 2006; FIGUEIREDO *et al.*, 2008; ALFEREZ; PELECHANO, 2011;

MARINHO *et al.*, 2013; SALLER *et al.*, 2013), sistemas de filmes (PARRA *et al.*, 2009), sistemas de saúde (CAVALCANTE *et al.*, 2012) e e-commerce (ALFÉREZ *et al.*, 2014a).

Todas as LPSDs citadas possuem a representação da variabilidade no modelo de *features*. Neste trabalho não foi identificado nenhum repositório de LPSDs e nenhum projeto que disponibilizasse a estrutura das LPSDs. Foram identificados dois repositórios que contém apenas modelos de *features* de LPSs tradicionais, o S.P.L.O.T. (MENDONCA *et al.*, 2009a) e o SPL2go[1]. O repositório S.P.L.O.T. apresenta alguns exemplos de LPSDs, no entanto, ela não suporta a representação da variabilidade dinâmica.

2.7 Notação do Modelo de *Features*

A modelagem de *features* é a atividade de identificação de aspectos visíveis e externos aos produtos de uma LPS e a sua organização em um modelo de *features* (LEE; MUTHIG, 2006). Existem algumas notações para representação das *features* e variabilidades de LPSs, as quais foram ao longo do tempo (e.g., (KANG *et al.*, 1990; KANG *et al.*, 1998; GRISS *et al.*, 1998; CZARNECKI *et al.*, 2004; BENAVIDES *et al.*, 2005)). Essas notações são ilustradas na Figura 9.

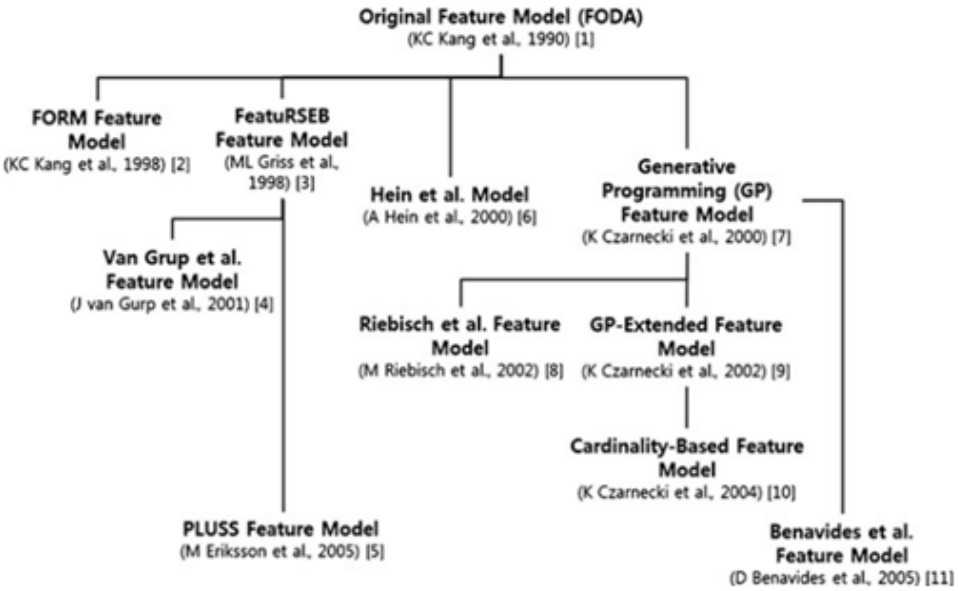


Figura 9 – Notações dos modelos de *features* (adaptado de (CAPILLA *et al.*, 2013))

No entanto, no contexto de LPSDs, poucas notações foram definidas (FERNANDES *et al.*, 2011). Muitas vezes estas notações se utilizam de outras, previamente existentes, como,

[1] http://spl2go.cs.ovgu.de

por exemplo, a notação FODA a fim incluir a modelagem do contexto, junto ou separado do modelo de *features* (LEE; MUTHIG, 2006; HARTMANN; TREW, 2008; CAPILLA *et al.*, 2014a).

A UbiFEX é uma notação específica para o modelo de *features* em LPSD e utiliza o modelo de contexto para modelagem da variabilidade dinâmica (FERNANDES *et al.*, 2011). O UbiFEX é uma notação para modelagem de *features* que tem como objetivo permitir a representação, de forma explícita, das entidades e informações de contexto relevantes para o domínio e, principalmente, a representação do impacto dessas informações na variabilidade dos produtos em tempo de execução.

2.8 Propriedades de Verificação do Modelo de *Features*

Para analisar a qualidade do modelo de *features* é necessário conhecer possíveis problemas que podem ocorrer nesse modelo, tais como: redundância[2], anomalias[3] e inconsistências[4]. Esses problemas podem servir de base para a derivação de medidas para avaliação da qualidade do modelo de *features*.

Na revisão sistemática realizada por Galster *et al.* (2014) sobre variabilidade em sistemas de software, foram identificadas algumas limitações dos modelos de variabilidade existentes, como:

- Corretude e consistência são difíceis de garantir, os autores reportam a necessidade para verificação e checagem da consistência. A corretude se refere a até que ponto o modelo atende às propriedades de boa formação especificadas. Já a consistência se refere ao grau de uniformidade, padronização e ausência de contradições entre tais regras, permitindo a configuração de pelo menos um produto. Essas regras de boa formação consistem de um conjunto de restrições sintáticas que determinam a estrutura de um modelo ou regra (MARINHO *et al.*, 2011);
- Limitações sobre atributos de qualidade, em particular a sobrecarga de desempenho, há limitações de suportar necessidades de segurança e *features* dinâmicas, como a capacidade dinâmica;

[2] Um modelo de *features* contém redundância, se pelo menos uma informação semântica é modelado de forma múltipla (MASSEN; LICHTER, 2004).

[3] Um modelo de *features* contém anomalias, se os potenciais configurações são perdidas, embora estas configurações devem ser possíveis. Isto decorre em circunstância de que alguma informação sem sentido foi modelada (MASSEN; LICHTER, 2004).

[4] Um modelo de *features* contém inconsistências, quando o modelo inclui informações contraditórias (MASSEN; LICHTER, 2004).

- Interface pobre com usuário, como por exemplo, em termos de flexibilidade e suporte ao usuário para manipulação da variabilidade; e
- Limitações sobre a identificação de variabilidade e variantes, como por exemplo, mais técnicas avançadas como mineração de dados poderiam ser aplicadas.

A corretude do modelo de *features* é verificada a partir de sua estrutura. Para verificação da corretude podem ser construídas regras de boa formação do modelo de acordo com a notação utilizada no modelo de *features*. Modelos corretos são aqueles que não possuem *features* mortas.

Um modelo de *features* contém anomalias se as possíveis configurações são inicialmente perdidas, embora estas configurações devam ser possíveis. Desta forma, o modelo de *features* deve ter informações modeladas que façam sentido. Um exemplo de anomalia é se uma *feature* opcional é mutuamente exclusiva com uma *feature* obrigatória, consequentemente a *feature* opcional nunca vai ser escolhida (MASSEN; LICHTER, 2004).

A consistência do modelo de *features* significa que ele é bem formado (consistência sintática) e que ele define pelo menos um produto válido (consistência semântica) (GUO *et al.*, 2012). Um modelo de *features* contém inconsistências se ele tem informações contraditórias. Segundo Von der Maßen e Lichter (MASSEN; LICHTER, 2004), a falta de consistência de um modelo de *features* é considerada um problema grave, maior do que a presença de anomalias.

Muitas abordagens têm sido propostas para automatizar a detecção de inconsistências, anomalias e problemas de corretude em modelos de *features* (BENAVIDES *et al.*, 2010). Eles mostram o uso de *Boolean Satisfiability Problem* (SAT) (*Satisfiability problem*) (BATORY, 2005; THÜM *et al.*, 2009; MENDONCA *et al.*, 2009b); BDD (*Binary Decision Diagrams*) (CZARNECKI; WASOWSKI, 2007; MENDONCA *et al.*, 2008; COSTA *et al.*, 2015), ou *Constraint Satisfaction Problem* (CSP) (*Constraint Satisfaction Problem*) (BENAVIDES *et al.*, 2005; TRINIDAD *et al.*, 2008) para automação do processo de verificação. Essas abordagens foram desenvolvidas para analisar modelos de *features* de LPSs tradicionais. Algumas propostas de automação da verificação do modelo de *features* foram identificadas para LPSDs utilizando SAT (FERNANDES *et al.*, 2011; MARINHO *et al.*, 2011; MARINHO *et al.*, 2012; LOCHAU *et al.*, 2015).

Segundo ZHANG et al. (ZHANG *et al.*, 2001), o SAT é um dos problemas NP-Completo mais estudado devido à sua importância tanto em pesquisa teórica quanto em aplicações práticas. Dada uma fórmula booleana, o SAT determina se há alguma interpretação que a satisfaça.

BDD são estruturas de codificação compactas para fórmulas booleanas que oferecem diversos algoritmos de raciocínio eficientes (BRYANT, 1986). O CSP é um paradigma declarativo para modelar e resolver problemas usando restrições (TSANG, 2014).

Tabela 1 – Recomendação de aplicação para SAT e BDD (adaptado de (MENDONCA *et al.*, 2009a)

Aplicação	**Caso**
SAT	Checar a satisfabilidade do modelo
	Detectar *features* mortas no modelo
	Detectar se uma determinada *feature* está morta no modelo
	Verificar a equivalência entre modelos
	Checar a especialização de modelos
	Validar configuração de modelo
	Enumerar as configurações de modelo
BDD	Detectar *features* mortas no modelo
	Verificar a equivalência entre modelos
	Checar a especialização de modelos
	Calcular domínios válidos
	Enumerar as configurações de modelo
	Calcular o número de configurações válidas
	Calcular o fator de variabilidade
	Calcular a similaridade de uma *feature*

Com a implementação SAT e BDD é possível derivar medidas em modelos de *features* para, checar a satisfabilidade do modelo, se há alguma *feature* morta, computar a quantidade de configurações válidas que podem ser geradas pelo modelo, entre outras. A Tabela 1 apresenta algumas recomendações de aplicação com SAT e BDD.

2.9 O Repositório S.P.L.O.T.

O S.P.L.O.T. é um sistema web desenvolvido em Java que utiliza a *engine* do modelo HTML. Ele é suportado por motores de configuração sofisticados e sistemas de raciocínio automatizados eficientemente baseados em SAT e BDD. Além disso, o S.P.L.O.T. disponibiliza um repositório de modelos de *features* (MENDONCA *et al.*, 2009a) e um conjunto de medidas para verificação do modelo de *features*.

O repositório *Software Product Lines Online Tools* (S.P.L.O.T.) tem mais de 600 modelos introduzidos pela comunidade de LPS para baixar, editar ou configurar produtos. No S.P.L.O.T, um modelo de *features* é representado por um formato XML e armazenado como um arquivo. Esses arquivos contêm um cabeçalho e as relações do modelo de *features* (corpo). O cabeçalho contém informações sobre autoria, tais como contas de autoria e filiação, além da data de criação do modelo de *feature*. O corpo do arquivo compreende a estrutura do modelo

de *features* e suas restrições de integridade. Em relação a estrutura do modelo no arquivo XML, as *features* são classificadas e representadas do seguinte modo: *feature* raiz (: r); *feature* obrigatória (: m), *feature* opcional (: o); *feature* com cardinalidade [1 *] (: g [1 *]); e *feature* com cardinalidade [1, 1] (: g [1, 1]). A Figura 10 apresenta a estrutura do modelo de *features* disponibilizada pelo S.P.L.O.T. no formato XML.

```
<feature_model name="Simple e-shop">
<meta>
<data name="description">family of e-shop stores, adapted from Wikipedia</data>
<data name="creator">Hartmut Lackner</data>
<data name="address"/>
<data name="email">hartmut.lackner@fokus.fraunhofer.de</data>
<data name="phone"/>
<data name="website"/>
<data name="organization">Fraunhofer FOKUS</data>
<data name="department"/>
<data name="date">Dec. 2012</data>
<data name="reference"/>
</meta>
<feature_tree>
:r eShop(_r)
    :m catalogue(_r_1)
    :m payment(_r_2)
        :g (_r_2_5) [1,*]
            : bank trasnsfer(_r_2_5_6)
            : credit card(_r_2_5_7)
            : e-coins(_r_2_5_11)
    :m security(_r_3)
        :g (_r_3_8) [1,1]
            : high(_r_3_8_9)
            : standard(_r_3_8_10)
    :o search(_r_4)
</feature_tree>
<constraints>
constraint_1:~_r_2_5_7 or _r_3_8_9
</constraints>
</feature_model>
```

Figura 10 – Estrutura do arquivo XML do modelo de *features* do repositório S.P.L.O.T.

Nesta Tese de doutorado, foram extraídos diversos modelos de *features* do repositório S.P.L.O.T. Além disso, também foram identificadas algumas medidas para avaliação da qualidade do modelo e o arquivo XML do modelo de *features* é utilizado como entrada para ferramenta desenvolvida neste trabalho.

2.10 Conclusões

Neste capítulo foram apresentados os principais conceitos relacionados a LPSs, LPSD e modelo de *features*. Dentre os conceitos apresentados, foram definidas as etapas de desenvolvimento das LPSDs, em um ciclo de vida que consiste da Engenharia de Domínio, Engenharia de Aplicação e Gerenciamento da linha. Foram apresentadas as principais propriedades das LPSDs, as quais se diferenciam das LPSs pela geração de sistemas adaptativos, como, por exemplo, os sistemas sensíveis ao contexto. Foram descritos os conceitos de variabilidade

estática e dinâmica, a representação do modelo de *features* foi discutida e exemplos de LPSDs foram apresentados. Além disso, também foram discutidas as notações existentes para o modelo de *features* e as propriedades de verificação da qualidade deste modelo. Por fim, foi apresentado o repositório de modelos de *features* S.P.L.O.T., que é utilizado neste trabalho para extração de modelos de *features*.

3 AVALIAÇÃO DA QUALIDADE DE PRODUTO

Neste capítulo são apresentados os principais conceitos relacionados à avaliação da qualidade de produto, modelos de qualidade e medições de software. Na Seção 3.1 são discutidas as definições de qualidade de software e avaliação da qualidade do produto. A Seção 3.2 apresenta os principais modelos de qualidade e um estudo comparativo entre esses modelos. Terminologias e conceitos relativos a medição de software são apresentadas na Seção 3.3. Os procedimentos operacionais das medidas são apresentados na Seção 3.4. Na Seção 3.5 são discutidos conceitos referentes à avaliação de qualidade em LPSs e LPSDs. Por fim, na Seção 3.6, são apresentadas as conclusões deste capítulo.

3.1 Introdução

Para avaliar a qualidade de um produto de software são necessárias avaliações de qualidade. A norma ISO/IEC 12207 (ISO/IEC, 2008) define qualidade como o grau no qual um sistema, componente ou processo atende as necessidades ou expectativas (i.e, requisitos) do cliente ou do usuário. Já a norma ISO/IEC 9000 (ISO/IEC, 2015) define qualidade como o grau no qual um conjunto de características inerentes satisfaz os requisitos. A qualidade é fundamental para o sucesso do produto de software (KITCHENHAM; WALKER, 1989).

A avaliação da qualidade de produtos de software tem como objetivo satisfazer as necessidades de qualidade em cada uma das etapas do ciclo de vida do desenvolvimento de software. A qualidade do produto de software pode ser avaliada medindo-se os atributos internos (tipicamente medidas estáticas de produtos intermediários), os atributos externos (tipicamente pela medição do comportamento do código quando executado) ou os atributos de qualidade em uso. O objetivo é que o produto tenha o efeito requerido em um cenário de uso particular (ISO/IEC, 2011).

É recomendado que, para a avaliação de qualidade de um produto de software, seja definido um modelo de qualidade e que este seja usado na definição das metas de qualidade para os produtos de software final e intermediários (ISO/IEC, 2014). Segundo a norma ISO/IEC 25010 (ISO/IEC, 2011), um modelo de qualidade de produto é definido por um conjunto de características e as relações entre elas, e fornece uma estrutura para avaliação de qualidade.

Modelos de qualidade são uma forma comum de lidar com a qualidade de uma forma estruturada e sistemática (WAGNER *et al.*, 2009). A qualidade do software, no entanto, não é a

mesma em todos os cenários de desenvolvimento de software, sendo influenciada por diferentes fatores, tais como o domínio da aplicação, a tecnologia utilizada e as restrições do projeto.

Os modelos de qualidade podem ser usados para identificar as características relevantes de qualidade, os critérios para a satisfação e as medidas a serem utilizadas para verificar se esses critérios estão sendo atendidos ou não (ISO/IEC, 2011). Dessa forma, de acordo com o domínio, é importante definir as características de qualidade e as medidas específicas para avaliar a qualidade do produto/artefato. Mais especificamente, um modelo de qualidade, que é definido de forma genérica, precisa ser customizado para projeto específico.

3.2 Modelos de Qualidade

Desde 1970 foi proposta uma série de modelos de qualidade de software com a finalidade de avaliar a qualidade de domínios gerais e específicos de produtos de software. Dentre os modelos de qualidade gerais mais citados na literatura tem-se os seguintes: McCall *et al.* (1977), Boehm *et al.* (1978), Dromey (1996), a norma ISO/IEC 9126 (ISO/IEC, 2001b) e a norma ISO/IEC 25000 (ISO/IEC, 2014).

A norma ISO/IEC 25000 (ISO/IEC, 2014), também conhecida como *Software Product Quality Requirements and Evaluation* (SQuaRE) (*Software Product Quality Requirements and Evaluation*), é uma evolução das normas ISO/IEC 9126 (ISO/IEC, 2001a) e ISO/IEC 14598. Da mesma forma que a ISO/IEC 9126, a SQuaRE propõe um modelo de qualidade composto por características, subcaracterísticas e atributos de qualidade. SQuaRE é a norma técnica mais recente para avaliação da qualidade do produto de software. Devido a esse fato, a norma SQuaRE (ISO/IEC, 2014) será utilizada para balizar as atividades realizadas nesta Tese com a finalidade de investigar a avaliação da qualidade de modelos de *features*.

3.2.1 SQuaRE

A norma ISO/IEC 25000 (ISO/IEC, 2014) consiste em um conjunto de normas para avaliação da qualidade de produtos de software. A SQuaRE possui as seguintes divisões (ISO/IEC, 2014): ISO/IEC 2500n (Divisão da Gestão da Qualidade), ISO/IEC 2501n (Divisão do Modelo de Qualidade), ISO/IEC 2502n (Divisão de Métricas de Qualidade), ISO/IEC 2503n (Divisão de Requisitos de Qualidade), ISO/IEC 2504n (Divisão da Avaliação da Qualidade) e ISO/IEC 25050 – ISO/IEC 25099 (Extensão de Padrões SQuaRE).

Tabela 2 – Características e atributos de qualidade da ISO/IEC 25010 (ISO/IEC, 2011)

Características de Qualidade	**Atributos de Qualidade**
Adequação Funcional	Completude Funcional
	Corretude Funcional
	Funcionalidade apropriada
Confiabilidade	Maturidade
	Tolerância a Falhas
	Recuperabilidade
	Disponibilidade
Usabilidade	Conhecimento Adequado
	Apreensibilidade
	Operabilidade
	Acessibilidade
	Proteção de Erro de Usuário
	Estética de Interface com o Usuário
Eficiência de Desempenho	Comportamento em Relação ao Tempo
	Comportamento em Relação aos Recursos
	Capacidade
Manutenibilidade	Analisabilidade
	Modificabilidade
	Modularidade
	Testabilidade
	Reusabilidade
Portabilidade	Adaptabilidade
	Instabilidade
	Substituibilidade
Segurança	Confidencialidade
	Integridade
	Não-Repúdio
	Responsabilização
	Autenticidade
Compatibilidade	Coexistência
	Interoperabilidade

Segundo a norma SQuaRE (ISO/IEC, 2011), um modelo de qualidade é composto de características, subcaracterísticas e atributos de qualidade. Característica de qualidade é um conjunto de propriedades de um produto de software pela qual sua qualidade pode ser definida e avaliada (ISO/IEC, 2001a). Uma característica pode ser dividida em várias subcaracterísticas. Atributos são propriedades mensuráveis, físicas ou abstratas, de uma entidade. As medidas de qualidade são utilizadas para medir, de forma direta ou indireta, os atributos, as subcaracterísticas e as características de qualidade em um produto de software. Uma medida é um mapeamento de uma entidade para um número ou símbolo, a fim de caracterizar uma propriedade da entidade (ISO/IEC, 2011). As medidas são essenciais na avaliação da qualidade de um produto, e estas podem compor um modelo de qualidade.

O modelo de qualidade SQuaRE categoriza a qualidade de software a partir das características de qualidade, as quais se subdividem em subcaracterísticas de qualidade. O modelo de qualidade para software da norma ISO/IEC 25000, foi atualizado de acordo com o

modelo de qualidade definido na ISO/IEC 9126. O modelo SQuaRE consiste em duas partes: (i) o modelo para qualidade de software interna e externa e (ii) o modelo para qualidade em uso (ISO/IEC, 2014).

A qualidade em uso busca avaliar a qualidade global do sistema no seu ambiente operacional, para usuários específicos, na realização de tarefas específicas. A qualidade de software externa fornece uma visão fechada do software e das propriedades relacionadas com a execução do software em um determinado hardware de computador e sistema operacional (ISO/IEC, 2011).

A qualidade de software interna fornece uma visão aberta do software e aborda as propriedades do produto de software que normalmente estão disponíveis durante o desenvolvimento. Qualidade de software interna está relacionada principalmente às propriedades estáticas do software (ISO/IEC, 2011).

O modelo SQuaRE possui oito características e trinta e uma subcaracterísticas relacionadas à qualidade interna e externa, conforme ilustrado na Tabela 2. As características e subcaracterísticas de qualidade apresentadas na Tabela 2, são descritas a seguir (ISO/IEC, 2011):

- **Adequação funcional**: grau em que um produto ou sistema fornece funções que correspondam às necessidades explícitas e implícitas quando utilizado sob as condições especificadas.
  - Completude funcional: grau no qual um conjunto de funções abrange todas as tarefas especificadas e os objetivos do usuário.
  - Corretude funcional: grau no qual um produto ou sistema fornece resultados corretos, com um determinado grau de precisão.
  - Funcionalidade apropriada: grau em que as funções facilitam a realização das tarefas e dos objetivos para os quais o sistema foi especificado.
- **Confiabilidade**: grau em que um sistema, produto ou componente mantém, ao longo do tempo, um comportamento consistente com o esperado, sob as condições especificadas.
  - Maturidade: grau no qual um sistema, produto ou componente satisfaz às necessidades de confiabilidade em sua operação normal.
  - Tolerância a falhas: grau em que um sistema, produto ou componente opera como pretendido, apesar da presença de falhas de hardware ou software.
  - Recuperabilidade: grau em que, em caso de interrupção ou falha, um produto ou sistema pode recuperar os dados diretamente afetados e re-estabelecer o estado

desejado do sistema.
  - Disponibilidade: grau em que um sistema, produto ou componente está operacional e acessível quando requisitado para uso.
- **Usabilidade**: grau em que um produto ou sistema pode ser utilizado por usuários específicos para atingir metas especificadas com eficácia, eficiência e satisfação, em um determinado contexto.
  - Conhecimento adequado: grau em que os usuários podem reconhecer se um produto ou sistema é apropriado para suas necessidades.
  - Apreensibilidade: grau em que um produto ou sistema pode ser usado por usuários para alcançar objetivos específicos de aprendizagem para utilização do produto ou sistema com eficácia, eficiência, inexistência de risco e satisfação, em um contexto de uso especificado.
  - Operabilidade: grau de facilidade com a qual um produto ou sistema é operado ou controlado.
  - Acessibilidade: grau em que um produto ou sistema pode ser usado por pessoas com a mais ampla gama de características e capacidades, a fim de alcançar um objetivo especificado em um contexto de uso determinado.
  - Proteção de erro do usuário: grau em que um sistema protege os usuários de cometer erros.
  - Estética da interface de usuário: grau em que uma interface de usuário permite interação agradável e satisfatória.
- Eficiência de desempenho: desempenho do produto em relação à quantidade dos recursos utilizados sob condições estabelecidas.
  - Comportamento no tempo: grau em que os tempos de resposta e de processamento e taxas de transferência de um produto ou sistema, no desempenho das suas funções, atende aos requisitos.
  - Utilização de recursos: diz respeito à quantidade de recursos necessários para que um produto ou sistema atenda aos requisitos.
  - Capacidade: grau em que os limites máximos do produto ou sistema satisfazem os requisitos.
- **Manutenibilidade**: grau de eficácia e eficiência com que um produto ou sistema pode ser modificado pela equipe de manutenção.

  - Analisabilidade: grau de eficácia e eficiência com a qual é possível avaliar o impacto sobre um produto ou sistema de uma mudança em uma ou mais de suas partes.
  - Modificabilidade: grau em que um produto ou sistema pode ser modificado de forma eficiente e eficaz sem a introdução de defeitos ou sem degradação de sua qualidade.
  - Modularidade: grau em que um sistema ou programa de computador é composto por componentes discretos, de forma que uma mudança em um componente tem um impacto mínimo sobre os demais.
  - Reusabilidade: grau em que um produto pode ser utilizado em mais do que um sistema, ou na construção de outros produtos.
  - Testabilidade: grau de eficácia e eficiência com que critérios de teste podem ser estabelecidos e executados.
- **Portabilidade**: grau de eficácia e eficiência com a qual um sistema, produto ou componente pode ser transferido de um hardware, software ou ambientes de uso.
  - Adaptabilidade: grau em que um produto ou sistema pode eficazmente e eficientemente ser adaptado para um hardware, software ou ambientes de uso diferentes ou em evolução.
  - Instalabilidade: grau de eficácia e eficiência com que um produto ou sistema pode ser instalado e/ou desinstalado com sucesso em num ambiente especificado.
  - Substituibilidade: grau em que um produto pode substituir outro produto de software especificado para o mesmo fim no mesmo ambiente.
- **Segurança**: grau em que as funções e os dados, de um produto ou sistema, são protegidos do acesso não autorizado e o grau em que são disponibilizados para acesso autorizado.
  - Confidencialidade: grau em que um produto ou sistema garante que os dados são acessíveis somente por pessoas autorizadas.
  - Integridade: grau em que um sistema, produto ou componente evita o acesso não autorizado ou a modificação de programas de computador ou dados.
  - Não-repúdio: grau em que um produto ou sistema permite constatar que ações ou acessos foram efetivamente realizados, de forma que não possam ser negados posteriormente.
  - Responsabilização: grau em que as ações de uma entidade podem ser atribuídas exclusivamente a esta entidade.
  - Autenticidade: grau em que a identidade de uma entidade (pessoa ou recurso) pode

ser comprovada a quem requisitar.

- **Compatibilidade**: grau em que um produto, sistema ou componente pode trocar informações com outros produtos, sistemas ou componentes, e/ou executar suas funções, enquanto compartilham o mesmo ambiente de hardware ou software.
  - Coexistência: grau em que um produto pode desempenhar as suas funções de forma eficiente ao compartilhar um ambiente e recursos comuns com outros produtos, sem impacto negativo em qualquer outro produto.
  - Interoperabilidade: grau em que dois ou mais sistemas, produtos ou componentes podem trocar informações e utilizar as informações que foram trocadas.

3.3 Medição de Software

Para avaliar a qualidade de um artefato/produto, uma das estratégias mais comuns consiste na utilização de medidas. Diferentes definições de medição de software podem ser encontradas na literatura. A seguir, algumas dessas definições são apresentadas:

> "Medição de software é uma avaliação quantitativa de qualquer aspecto dos processos e produtos da Engenharia de Software, que permite seu melhor entendimento e, com isso, auxilia o planejamento, controle e melhoria do que se produz e de como é produzido (BASS; CLEMENTS, 1999)."

> "Medição de software é o processo contínuo de definição, coleta e análise de dados sobre o desenvolvimento de software e de seus produtos, a fim de compreendelos e controlá-los, afim de fornecer informações significativas de forma a melhorar tanto o processo de desenvolvimento de software quanto os seus produtos (SOLINGEN *et al.*, 2002)."

> "A medição é definida como o processo pelo qual os números ou símbolos são atribuídos aos atributos de entidades em mundo real, de tal forma que pode-se descrevê-los de acordo com regras claramente definidas (FENTON; BIEMAN, 2014)."

Terminologias, conceitos, procedimentos e métodos de medição de software vêm sendo definidos na última década. Porém, não há consenso quanto a conceitos e terminologias,

havendo duplicidades e inconsistências nas propostas encontradas na literatura, inclusive nos termos mais comuns da área, tais como medida, métrica e medição (GARCÍA *et al.*, 2006).

Kitchenham *et al.* (1995) apresentam um modelo estrutural para medição de software que permite descrever os objetos envolvidos na medição e seus relacionamentos como:

- **Entidades**: são os objetos observados no mundo real, que são capturados em suas características e manipulados formalmente, por exemplo, produtos e processos;
- **Atributos**: são propriedades que as entidades possuem. Para um dado atributo, há um relacionamento de interesse no mundo empírico, que, formalmente, se quer apreender no mundo matemático; e
- **Relacionamento entre entidades e atributos**: uma entidade pode ter vários atributos, enquanto que um atributo pode qualificar diferentes entidades.

García *et al.* (2006) desenvolveram uma ontologia para medição de software denominada *Software Measurement Ontology* (SMO) (*Software Measurement Ontology*). Nesta ontologia, os autores apresentam alguns conceitos que são comuns à medição, tais como:

- **Atributo**: propriedade física ou abstrata mensurável de uma entidade, que é compartilhado por todas as entidades de uma classe de entidade. Onde entidade é um objeto que está sendo caracterizado pela medição de seus atributos e classe de entidades é uma coleção de todas as entidades semelhantes;
- **Modelo de qualidade**: conjunto de conceitos mensuráveis e as relações entre eles que fornecem a base para a especificação de requisitos de qualidade e avaliação da qualidade das entidades de determinada classe de entidade;
- **Medida**: quantificação de atributos de entidades. A abordagem de medição é definida e a escala de medição. A abordagem de medição é o método da medida, uma função de medição ou um modelo de análise;
- **Escala**: indica os valores que podem ser atribuídos a uma medida;
- **Medida Base**: uma medida de um atributo que não depende de qualquer outra medida;
- **Medida Derivada**: uma medida que é derivada de outra medida base ou derivada, utilizando uma função de medição;
- **Indicador**: medida utilizada para analisar o alcance a objetivos;
- **Medição**: ação de medir, ou seja, de atribuir um valor a uma medida, executando seu procedimento de medição; e
- **Resultado da Medição**: O número ou categoria atribuído a um atributo de uma entidade

por meio de uma medição.

Barcellos (BARCELLOS, 2009; ROCHA *et al.*, 2012) desenvolveu outra ontologia de medição que complementa os conceitos apresentados por García *et al.* (2006). Os conceitos de medição complementares são:

- **Elemento Mensurável**: propriedade de uma entidade que pode ser quantificada;
- **Entidade Mensurável**: entidade que pode ser caracterizada pela quantificação dos seus atributos;
- **Unidade de Medida**: unidade por meio do qual uma medida pode ser expressa;
- **Escala**: indica os valores que podem ser atribuídos a uma medida;
- **Procedimento de Medição**: procedimento que descreve como a coleta da medida deve ser realizada;
- **Procedimento de Análise de Medição**: procedimento que descreve como os dados coletados para a medida devem ser apresentados e analisados;
- **Definição Operacional da Medida**: detalhamento associado a uma medida que fornece informações sobre sua coleta e análise. Os detalhes da definição operacional de uma medida estão dispostos na Seção 3.4; e
- **Análise de Medição**: ação de analisar os dados para uma medida, executando seu procedimento de análise.

Tendo como base essa terminologia, Bush e Fenton (BUSH; FENTON, 1990) classificam as entidades da engenharia de software, que são passíveis de medição, em três grupos distintos:

- **Processos**: que são todas as atividades (com um fator tempo);
- **Produtos**: que são quaisquer artefatos que surgem dos processos; e
- **Recursos**: que são os itens usados por processos, excluindo produtos de outros processos.

Segundo Fenton e Melton (1990), os atributos das medidas podem ser internos ou externos:

- **Atributos internos**: podem ser medidos em termos meramente de produto, processo ou recurso em si; e
- **Atributos externos**: são aqueles que só podem ser medidos em relação à forma como o produto, processo ou recurso se relaciona com outras entidades no seu ambiente.

Fenton e Melton (1990) ainda indicam que existem dois tipos de medidas:

- **Medição direta de um atributo**: é a medição que não depende da medida de qualquer

outro atributo; e item **Medida indireta de um atributo**: envolve a medição de um ou mais outros atributos.

Kan (2002) define níveis de medição que podem ser:

- **Escala nominal**: os dados são identificados apenas pela atribuição de um nome que representa uma categoria. Qualquer dado pertence a uma categoria, estas devem ser exaustivas, mutuamente exclusivas (cada dado pertence a uma só categoria) e não ordenáveis (não é estabelecida preferência de uma classe em relação a outra). Por exemplo, se o atributo de interesse é religião, então valores que este atributo pode assumir seriam: Católico, evangélico, Judeu, Budista, dentre outros;
- **Escala ordinal**: ordena as entidades segundo um critério definido. Nesta escala, as afirmações do tipo "maior do que..." podem ser realizadas;
- **Escala intervalo**: ordena os valores da mesma forma que a escala ordinal, mas acrescenta a noção de distância relativa entre as entidades. As operações matemáticas de adição e subtração podem ser aplicadas aos valores na escala de intervalo.

3.4 Definição Operacional das Medidas

É importante que as medidas definidas de acordo com as características, subcaracterísticas e/ou atributos de qualidade, possuam definições operacionais para guiar a coleta, análise e armazenamento da medida de forma independente. Uma Definição Operacional de Medida é um detalhamento de aspectos relacionados à coleta e análise de uma medida, estabelecido por uma organização, de acordo com objetivos de medição específicos (BARCELLOS, 2009).

Para Florac e Carleton (1999), as medidas precisam estar bem definidas, satisfazendo os seguintes critérios:

- **Comunicação**: os métodos de medição usados para definir medidas ou descrever valores medidos devem permitir que se atenda precisamente o que foi medido e como os dados foram coletados, de forma que seja possível interpretar os resultados corretamente;
- **Repetitividade**: a definição operacional da medida deve ser repetível, de modo que pessoas diferentes possam realizar a mesma medição e obter os mesmos resultados; e
- **Rastreabilidade**: a origem dos dados deve estar identificada em termos de tempo, sequência, atividade, produto, status, ambiente e ferramentas de medição utilizadas, e responsável pela coleta.

Barcellos (2009) propõe uma ontologia para definição operacional das medidas.

Tabela 3 – Definição operacional da medida (adaptado de (BARCELLOS, 2009))

Definição Operacional da Medida	**Descrição**
Nome	Nome da Medida
Definição da Medida	Descrição sucinta da medida
Mneumônico	Sigla utilizada para identificar a medida
Tipo da Medida	Classificação da medida quanto à sua dependência funcional, podendo uma medida ser uma medida base ou uma medida derivada.
Entidade da Medida	Entidade que a medida mede. Exemplos: Organização, projeto, processo, atividade, e artefato, dentre outros.
Propriedade da Medida	Propriedade da entidade medida quantificada pela medida. Exemplos: tamanho, custos, defeitos, esforço, dentre outros.
Unidade da Medida	Unidade de medida em relação à qual a medida é medida. Exemplos: pessoa/mês, pontos de função, reais, dentre outros.
Tipo de Escala	Natureza dos valores que podem ser atribuídos à medida. Exemplos: escala nominal, escala intervalar, escala ordinal, escala absoluta e escala taxa.
Valores da Escala	Valores que podem ser atribuídos à medida. Exemplos: números reais positivos, {alto, médio, baixo}, dentre outros.
Intervalo esperado dos dados (se possível)	Limites de valores da escala definida de acordo com dados históricos ou com metas estabelecidas. Exemplo: [0, 10].
Procedimento de Medição	Descrição do procedimento que deve ser realizado para coletar uma medida. A descrição do procedimento de medição deve ser clara, objetiva e não ambígua.
Fórmula de Cálculo da Medida	Fórmula utilizada no procedimento de medição de medidas derivadas, para calcular o valor atribuído à medida considerando-se sua relação com outras medidas ou com outros valores.
Responsável pela Medição	Papel desempenhado pelo recurso humano responsável pela coleta da medida. É importante que o responsável pela medição seja fonte direta das informações a serem fornecidas na medição. Exemplos: analista de sistemas, engenheiro de domínio, dentre outros.
Momento da Medição	Momento em que deve ser realizada a coleta e registro de dados para a medida. O momento da coleta deve ser uma atividade do processo definido para o projeto ou de um processo organizacional. Exemplos: na atividade Elaborar modelo de *features*.
Periodicidade da Medição	Frequência de coleta da medida. Exemplos: diária, mensal, uma vez por fase, entre outros.
Procedimento de Análise (se indispensável)	Descrição do procedimento que deve ser realizado para representar e analisar os dados coletados para uma medida, incluindo, além do procedimento propriamente dito, as ferramentas analíticas que devem ser utilizadas (por exemplo: histograma, gráfico de controle XmR, entre outros). O procedimento de análise pode ser omitido em medidas base que não são analisadas isoladamente, ou seja, quando não estão associadas a outras medidas onde o procedimento e análise é claramente descrito.
Momento da Análise de Medição (se aplicável)	Momento em que deve ser realizada a análise de dados coletados para a medida. O momento da análise deve ser uma atividade do processo definido para o projeto ou de um processo organizacional como, por exemplo, em atividades de monitoramento de projeto.
Periodicidade da Análise (se aplicável)	Frequência de análise de dados da medida.Exemplos: diária, mensal, uma vez por fase, uma vez por projeto, uma vez em cada ocorrência da atividade designada como momento da análise.
Responsável pela Análise (se aplicável)	Papel desempenhado pelo recurso humano responsável pela análise da medida. Exemplos: gerente do projeto, engenheiro de domínio, entre outros.

Segundo Barcellos (2009) uma definição operacional da medida é composta pelos elementos ilustrados na Tabela 3. Nesta Tese, as medidas foram definidas conforme a definição operacional proposta na Tabela 3.

3.5 Avaliação de Qualidade em LPSs e LPSDs

A avaliação da qualidade no contexto de linhas de produto de software é essencial, uma vez que um erro ou uma incompatibilidade em um ativo reutilizável pode ser propagado para um lote de produtos. Neste sentido, a avaliação da qualidade em LPSs possui alguns desafios que não estão presentes no desenvolvimento de software tradicional (ETXEBERRIA; SAGARDUI, 2008b).

Segundo Clements P. e Northrop (2002), a avaliação de uma LPS deve se concentrar nas variabilidades, garantindo assim, que sejam apropriadas, ofereçam flexibilidade suficiente para o escopo da LPS, possam ser exercitadas para derivar os produtos específicos e não imponham custos de desempenho inaceitáveis em tempo de execução.

Em uma linha de produtos de software, os requisitos dos atributos de qualidade podem ser classificados em dois tipos diferentes (ETXEBERRIA; SAGARDUI, 2005):

- **Atributos de qualidade da linha de produto**: são atributos considerados de desenvolvimento ou não observáveis através de execução. São aqueles que são intrínsecos ou específicos para linhas de produtos. Estes atributos são aqueles relacionados à variabilidade ou flexibilidade. A avaliação da variabilidade de uma linha de produto garante que é possível obter todas as funcionalidades dos produtos em um escopo imaginado. e
- **Atributos de qualidade relevantes ao domínio**: geralmente são observáveis através da execução do produto de software. Os atributos de qualidade relevantes ao domínio (como a segurança em um domínio de segurança crítica, o desempenho em sistemas de tempo real, entre outros) também devem ser abordados na linha de produtos. Caso contrário, as implicações ou consequências podem ser muito graves e de difícil correção. Como produtos distintos podem exigir valores de atributos diferentes (nem todos os produtos requerem o mesmo nível de segurança, por exemplo), a variabilidade, na forma como o atributo é traduzido para o produto, é relevante para a avaliação.

Muitos dos estudos em avaliação da qualidade em LPSs têm investigado os atributos de qualidade da linha de produto, mas do ponto de vista da arquitetura na engenharia de domínio (MATINLASSI *et al.*, 2002; ETXEBERRIA; SAGARDUI, 2005; OLUMOFIN; MIŠIĆ, 2007; KIM *et al.*, 2008; JUNIOR *et al.*, 2013). Poucas pesquisas têm se concentrado na avaliação de qualidade dos atributos de qualidade do modelo de *features* (THÖRN, 2007; BAGHERI E. E GASEVIC, 2011; ZHANG *et al.*, 2014).

Em relação às técnicas de avaliação de qualidade para Linhas de Produto de Software,

Etxeberria and Sagardui (ETXEBERRIA; SAGARDUI, 2005) classificam essas técnicas em dois grupos:

- **Técnicas de Questionários para Avaliação Qualitativa**: incluem cenários, questionários e *checklists*. Podem ser usadas para avaliar a qualidade de desenvolvimento ou operacional; e
- **Técnicas de Medição para Avaliação Quantitativa**: incluem simulação, prototipagem, modelos matemáticos e experimentais. Podem ser usadas para medir a aplicação de técnicas que englobam qualidades específicas, normalmente operacionais.

Segundo Etxeberria e Sagardui (ETXEBERRIA; SAGARDUI, 2008b), a avaliação de qualidade pode ser realizada em diferentes fases: nas fases iniciais, relacionadas à engenharia de domínio, como, por exemplo, na fase de *design* (métodos de avaliação de arquitetura de software), ou em fases relacionadas à engenharia de aplicação (métodos de medição de qualidade). Para reduzir custos e aumentar a qualidade dos artefatos e dos produtos da linha, a avaliação de qualidade deve ser aplicada em fases iniciais da linha, mais especificamente na Engenharia de Domínio. Muitos trabalhos têm investigado a avaliação de qualidade da arquitetura da LPS, mas pouco se tem pesquisado acerca da avaliação da qualidade do modelo de *features*. Devido a essa limitação, novas pesquisas nesse tema são necessárias.

Durante a realização desta Tese, não foram encontrados na literatura estudos ou abordagens que tivessem por objetivo principal a avaliação da qualidade dos modelos de *features* de LPSD a partir da utilização de medições.

Porém, foram encontrados alguns trabalhos que utilizam medidas para avaliar LPSDs (SALLER *et al.*, 2013; ALFÉREZ *et al.*, 2014a). A maior parte dos trabalhos de avaliação de LPSDs concentra-se na verificação e validação da variabilidade dinâmica do modelo de *features* (CAPILLA; BOSCH, 2011; MARINHO *et al.*, 2012; ALFÉREZ *et al.*, 2014a). Dessa forma, existe uma lacuna na área de avaliação de qualidade do modelo de *features* em LPSs e LPSDs.

3.6 Conclusões

Neste capítulo foram apresentados os principais conceitos relacionados à avaliação da qualidade do produto, modelos de qualidade e medições de software. Foram apresentados os principais modelos de qualidade encontrados na literatura. Dentre os modelos de qualidade apresentados, foi detalhado o modelo de qualidade SQuaRE, utilizado como base nesta Tese de doutorado. Além disso, foram apresentados os principais conceitos sobre medição de software.

Por fim, foram destacados os principais conceitos acerca da avaliação da qualidade em LPSs e LPSDs.

4 TRABALHOS RELACIONADOS

Este capítulo apresenta os principais trabalhos relacionados à tese de doutorado. Na Seção 4.1 é descrita uma introdução sobre estratégias e abordagens para avaliação da qualidade em LPSs e LPSDs. A Seção 4.2 apresenta as abordagens de avaliação de qualidade para o modelo de *features* de LPSs. Na Seção 4.3 e na Seção 4.4 são apresentados estudos que aplicam medidas para avaliar o modelo de *features* de LPSs e LPSDs, respectivamente. Na Seção 4.5 são apresentadas ferramentas para avaliação da qualidade do modelo de *features*. Em seguida, na Seção 4.6 é apresentada a comparação entre os trabalhos relacionados. Finalmente, na Seção 4.7 são apresentadas as conclusões do capítulo.

4.1 Introdução

Muitas abordagens de avaliação da qualidade em LPSs têm sido propostas ao longo dos últimos anos (OLUMOFIN; MIŠIĆ, 2007; CUEVAS *et al.*, 2007; TRINIDAD *et al.*, 2008; KIM *et al.*, 2008; MENDONCA *et al.*, 2009a; BAGHERI E. E GASEVIC, 2011; JUNIOR *et al.*, 2013).

Montagud e Abrahão (MONTAGUD; ABRAHãO, 2009) executaram uma revisão sistemática compilando o conhecimento sobre avaliação da qualidade em LPSs. Dentre os dados extraídos dos estudos primários identificados nesta revisão tem-se as abordagens de avaliação da qualidade, as fases do ciclo de vida, os artefatos avaliados, os mecanismos usados para capturar os atributos de qualidade, o tipo de atributo de qualidade avaliado, a conformidade com padrões de qualidade, o suporte à avaliação e o procedimento de avaliação. Nesse estudo foi verificado que a maior parte dos modelos de avaliação da qualidade se concentram na fase de arquitetura da engenharia de domínio da LPS (OLUMOFIN; MIŠIĆ, 2007; KIM *et al.*, 2008; JUNIOR *et al.*, 2013), poucos estudos tem avaliado o modelo de *features* (TRINIDAD *et al.*, 2008; BAGHERI E. E GASEVIC, 2011). Poucas ferramentas de suporte à avaliação da qualidade foram identificadas na revisão (e.g., (CUEVAS *et al.*, 2007; TRINIDAD *et al.*, 2008; MENDONCA *et al.*, 2009a)), e algumas ferramentas identificadas apenas suportam a análise semi-automática da avaliação da arquitetura. A maior parte dos estudos também não utiliza padrões e normas de conformidade, como o SQuaRE (ISO/IEC, 2014), para avaliação da qualidade (OLUMOFIN; MIŠIĆ, 2007; TRINIDAD *et al.*, 2008; MENDONCA *et al.*, 2009a; JUNIOR *et al.*, 2013).

Thüm *et al.* (2014) elaboraram um *survey* em que abordam estratégias de análise de

linhas de produto de software. Nesse estudo, algumas abordagens identificadas estão relacionadas à abordagens e estudos sobre testes em LPSs (KIM *et al.*, 2012; KÄSTNER *et al.*, 2012; NGUYEN *et al.*, 2014; SANTOS *et al.*, 2015). Outras abordagens identificadas no *survey* e na literatura estão relacionadas à verificação e análise automática do modelo de *features* (BENAVIDES *et al.*, 2010; MARINHO *et al.*, 2012; ASADI *et al.*, 2016). Além disso, outros estudos identificados apresentam medidas para avaliar o modelo de *features* de LPSs (BAGHERI E. E GASEVIC, 2011; MONTAGUD *et al.*, 2012; BERGER T. E GUO, 2014; OLIINYK *et al.*, 2015; BEZERRA *et al.*, 2015).

A avaliação da qualidade em LPSDs ainda é um tema recente e possui maiores desafios que a avaliação da qualidade em LPSs, devido à variabilidade dinâmica das LPSDs. Foram identificados na literatura alguns estudos que tratam da verificação das LPSDs em tempo de execução (PASCUAL *et al.*, 2015; LOCHAU *et al.*, 2015). Também foram identificados outros estudos que propuseram abordagens para o modelo de *features* em LPSDs e avaliaram essas abordagens com medidas específicas (CETINA *et al.*, 2009b; SALLER *et al.*, 2013; ALFÉREZ *et al.*, 2014a). No entanto, não foram identificadas na literatura abordagens de avaliação da qualidade para o modelo de *features* de LPSDs baseada em medidas.

Nesta Tese de doutorado são usados apenas os estudos relacionados à avaliação de qualidade do modelo de *features* em LPSs e LPSDs utilizando medidas. Dessa forma, são discutidos de forma mais detalhada, os trabalhos de LPSs e LPSDs que estão relacionados à avaliação da qualidade do modelo de *features* baseados em medidas. Esses trabalhos foram agrupados para discussão em quatro tópicos e discutidos nas próximas seções: abordagens para avaliação da qualidade do modelos de *features*, estudos relacionados à medidas para o modelo de *features* de LPSs, estudos que apresentam medidas para o modelo de *features* de LPSDs e ferramentas para avaliação da qualidade do modelo de *features*.

4.2 Abordagens para Avaliação da Qualidade do Modelo de *Features*

Na literatura foram identificadas poucas abordagens de avaliação da qualidade da variabilidade em LPSs utilizando o modelo de *features* (ETXEBERRIA; SAGARDUI, 2008b; DEELSTRA *et al.*, 2009; THÖRN, 2007).

Dentre as abordagens mais relevantes identificadas têm-se o modelo de qualidade em LPSs dirigido à variabilidade (ETXEBERRIA; SAGARDUI, 2008b), o método COSVAM (DEELSTRA *et al.*, 2009) e o modelo de qualidade para modelos de *features* baseado em

medições (THÖRN, 2007; THÖRN, 2010). Não foram encontradas abordagens para avaliação da qualidade do modelo de *features* de LPSDs.

Etxeberria e Sagardui (ETXEBERRIA; SAGARDUI, 2008b) apresentam um modelo para avaliação da qualidade em LPSs dirigido à variabilidade do modelo de *features*. A notação utilizada para avaliação do modelo de *features* é o FeatuRSEB (GRISS *et al.*, 1998). O modelo de avaliação apresentado é dividido em: (i) construção de um método de avaliação genérico, (ii) seleção de produtos, (iii) avaliação, e (iv) adição de *features*. Os autores utilizam outros métodos de avaliação da qualidade de arquitetura de LPSs para o modelo de *features*, como o modelo PASA (WILLIAMS; SMITH, 2002), reduzindo o escopo de avaliação para os produtos derivados do modelo de *features*. O fato do modelo utilizar outros métodos de avaliação da qualidade pode dificultar a avaliação do modelo de *features* pelo engenheiro de domínio. Além disso, os métodos utilizados são específicos para avaliação da arquitetura. Como o escopo de avaliação é reduzido para alguns produtos, o modelo de *features* pode não ser avaliado como um todo.

Outro método de avaliação identificado foi o COSVAM (*COVAMOF software variability assessment method*) (DEELSTRA *et al.*, 2009). O COSVAM é um método de avaliação da variabilidade em LPS. Para modelagem da variabilidade o método utiliza o *framework* COVAMOF (SINNEMA *et al.*, 2004). O método trata uma variedade de situações em que a principal questão é saber como e quando evoluir a variabilidade. Este método provê um processo para avaliar a variabilidade de forma qualitativa, a partir do desmembramento da variabilidade e do impacto da variabilidade. No entanto, a aplicação do método é genérica e apenas aponta falhas de variabilidade, não envolvendo regras de como realizar essa avaliação. Uma dificuldade identificada para utilização do método é a representação do modelo de variabilidade (COVAMOF), pois é um modelo de variabilidade pouco utilizado na indústria e na academia no campo de LPSs.

Thörn (THÖRN, 2007; THÖRN, 2010) propõe um modelo de qualidade para modelos de *features* em LPSs, procedimentos para a priorização e avaliação da qualidade em modelos de *features* e um conjunto inicial de princípios de desenvolvimento empiricamente fundamentados para atingir determinada qualidade no modelo de *features*. O modelo desenvolvido utiliza fatores de qualidade para obter, a partir dos *stakeholders*, resultados qualitativos e quantitativos a respeito da qualidade do modelo de *features*. O modelo de qualidade ainda define seis fatores de qualidade que podem ser selecionados, que são: modificabilidade, reusabilidade, formalismo, mobilidade, corretude e usabilidade (THÖRN, 2007; THÖRN, 2010). A abordagem de (THÖRN, 2007), embora específica para o modelo de *features*, não indica quais as possíveis medidas ou

indicadores poderiam representar esses fatores de qualidade, o que pode dificultar a execução do modelo de qualidade pelo engenheiro de domínio.

As abordagens identificadas para avaliação da qualidade do modelo de *features* em LPSs são genéricas e dependem do esforço do engenheiro de domínio para identificar métodos de avaliação de qualidade, fatores de qualidade que afetam o modelo de *features* e medidas para avaliação da qualidade. Esses modelos de avaliação podem não ser atrativos para o engenheiro de domínio devido ao alto custo de execução dos modelos e por não conterem informações mais específicas para avaliação da qualidade do modelo de *features*. Além disso, nenhum dos modelos trata da avaliação da qualidade do modelo de *features* em LPSDs.

4.3 Medidas para o Modelo de *Features* em LPSs

Os trabalhos descritos nesta Seção foram os que apresentaram maior relação com a pesquisa apresentada nesta Tese. Foram identificados estudos que propuseram medidas para avaliação da qualidade do modelo de *features* em LPSs (BAGHERI E. E GASEVIC, 2011; MONTAGUD *et al.*, 2012; BERGER T. E GUO, 2014).

Bagheri E. e Gasevic (2011) propuseram 12 medidas estruturais para avaliar a qualidade de 14 modelos de *features* em LPSs extraídos do repositório S.P.L.O.T. (MENDONCA *et al.*, 2009a). Os autores validaram as medidas propostas usando princípios de medição teórica (BRIAND *et al.*, 1996; POELS; DEDENE, 2000). Foi realizado um experimento controlado com alunos de graduação em Ciência da Computação a fim de analisar se estas medidas estruturais poderiam ser bons preditores de três subcaracterísticas da característica de qualidade manutenibilidade: analisabilidade, modificabilidade e entendibilidade. Uma ameaça à validade é que o estudo utiliza apenas 14 modelos de *features* para realização do experimento. Além disso, a maior parte dos modelos de *features* não pertencem a LPSs reais e não representam o domínio de software (e.g., representam o domínio de bicicleta, *laptops*, entre outros.). A quantidade de medidas utilizadas pode não representar todos os fatores de manutenibilidade do modelo de *features*, podem existir outras subcaracterísticas de manutenibilidade que não foram cobertas. Por último, o experimento realizado com estudantes de graduação é uma ameaça à validade do estudo, para maior confiabilidade o experimento deveria ser realizado com especialistas em LPSs.

Em Montagud et al. (MONTAGUD *et al.*, 2012) foi executada uma revisão sistemática com o objetivo de identificar na literatura os estudos que apresentam atributos ou

medidas para LPSs. Os atributos e medidas identificados no estudo foram classificados usando um conjunto de critérios que inclui a fase do ciclo de vida em que as medidas são aplicadas. Como resultado da revisão sistemática, foi elaborado um catálogo com medidas e atributos de qualidade para todas as fases da LPS. No entanto, apesar de extenso, no catálogo foram identificadas poucas medidas relacionadas à avaliação da qualidade do modelo de *features*. A maior parte das medidas do catálogo estão relacionadas à avaliação da qualidade da arquitetura da engenharia de domínio na LPS. Além disso, o catálogo elaborado não fornece as fórmulas de cálculo das medidas e não foi validado o uso destas medidas, não comprovando se as medidas são eficazes para avaliação da qualidade da LPS.

Berger e Gui (BERGER T. E GUO, 2014) executaram a análise de correlação com medidas de código e medidas do modelo de *features* para LPSs. Foram identificadas nesse estudo algumas medidas para avaliação do modelo de *features* extraídas da literatura e outras medidas que foram propostas pelos autores. Para realização do estudo, os autores utilizaram um conjunto de 8 LPSs reais. O estudo apresentou resultados iniciais da análise das medidas de LPSs, concluindo que existe correlação entre essas medidas de código e do modelo de *features*, principalmente as medidas relacionadas ao tamanho. Uma grande ameaça identificada nesse estudo é o pequeno número de LPSs utilizadas para análise de correlação.

Os estudos que abordam medidas para avaliação do modelo de *features* de LPSs possuem algumas deficiências e limitações. Grande parte dos estudos apresentam poucas medidas e avaliam uma quantidade pequena de modelos de *features* (BAGHERI E. E GASEVIC, 2011). Apenas algumas características e subcaracterísticas para avaliação do modelo de *features* são cobertas.

Nesta Tese de doutorado, resultados iniciais apresentando medidas para avaliar a qualidade do modelo de features de LPSs já foram publicados, diferenciando dos trabalhos existentes como explicado a seguir. Bezerra et al. (BEZERRA *et al.*, 2015), propuseram um catálogo de características, subcaracterísticas e medidas para suportar a avaliação da qualidade do modelo de *features* em LPSs, extraídas da literatura a partir de um mapeamento sistemático. As medidas identificadas no catálogo são relacionadas à característica de qualidade de manutenibilidade. O catálogo foi aplicado para avaliação do uso em poucos modelos de *features*. Esse estudo foi estendido em Bezerra et al. (BEZERRA *et al.*, 2016), para realização de um estudo de caso exploratório para análise e investigação das medidas de manutenibilidade do catálogo proposto. O catálogo apresenta um conjunto extenso de medidas relacionados às características

e subcaracterísticas de qualidade, e cada medida possui informações sobre a fórmula de cálculo definida. O estudo de caso exploratório conclui que a utilização de medidas pode suportar a avaliação da qualidade de modelos de *features*. Também conclui que algumas medidas estão correlacionadas e que pode utilizar apenas um conjunto reduzido de medidas para avaliação da qualidade. Além disso, é possível agrupar medidas para cada subcaracterística e é possível definir *thresholds* para as medidas a partir de uma base de dados.

4.4 Medidas para o Modelo de *Features* em LPSDs

Na literatura foram identificados poucos estudos que propusessem ou apresentassem medidas para avaliação da qualidade do modelo de *features* em LPSDs (CETINA *et al.*, 2009b; SALLER *et al.*, 2013; ALFÉREZ *et al.*, 2014a).

Os estudos identificados propuseram medidas para avaliação de abordagens para variabilidade da LPSDs que foram propostas para os estudos específicos. Alguns dos trabalhos relacionados foram identificados a partir da revisão da literatura descrita no Capítulo 5, na Seção 5.2. Aqui, nesta Seção, são discutidos os trabalhos mais relevantes dessa revisão, que propuseram mais medidas para o modelo de *features* de LPSDs.

Em Cetina et al. (CETINA *et al.*, 2009b) é introduzida uma abordagem baseada no modelo de *features* para reconfiguração dinâmica de sistemas *smart homes*. A abordagem foi automatizada para modelagem e verificação do modelo de *features* utilizando a ferramenta FAMA (TRINIDAD *et al.*, 2008). Para validação, a abordagem foi aplicada em sistemas *smart homes* e foram coletadas medidas relacionadas a reconfiguração dinâmica.

Da mesma forma que Cetina *et al.* (2009a), Saller *et al.* (2013) propuseram uma nova abordagem para a modelagem de cenários de adaptação em LPSDs sensíveis ao contexto para sistemas com restrições de recurso (e.g., o uso de memória). A abordagem é baseada no modelo de *features* enriquecido com informações de contexto. Para avaliar a abordagem, o trabalho propôs medidas relacionadas ao tempo de projeto e a reconfiguração em tempo de execução da LPSD.

As medidas propostas em Cetina *et al.* (2009a) e Saller *et al.* (2013) são específicas para as abordagens propostas, mas podem ser adaptadas para outras abordagens de modelos de *features* de LPSDs.

Alférez *et al.* (2014a) propuseram um *framework* que utiliza modelos de *features* em tempo de execução para apoiar a adaptação dinâmica para composição de serviços na web. A

composição de serviços adiciona e remove fragmentos do código WS-BPEL (*Business Process Execution Language*). Para avaliação do *framework*, os autores utilizaram o método GQM *(Goal/Question/Metric*) (SOLINGEN *et al.*, 2002). As medidas elaboradas foram relacionadas a seis objetivos: eficiência do tempo de geração da configuração do modelo de *features*, redução da complexidade do espaço da adaptação, redução das anomalias no modelo de *features* e na sua configuração, eficiência de verificação do modelo de *features*, eficiência da adaptação dinâmica e evitar erros em circunstâncias de *stress*. Embora o objetivo do trabalho fosse propor o *framework*, ele disponibiliza uma grande quantidade de medidas para avaliação do modelo de *features* de LPSDs. No entanto, a maior parte das medidas se aplica apenas ao cenário específico do *framework*.

Pela análise dos trabalhos na revisão apresentada nesta Seção e considerando também o restante apresentado no próximo Capítulo 5, foi identificada uma lacuna de medidas para avaliação do modelo de *features* de LPSDs relacionadas à variabilidade dinâmica do modelo nesses trabalhos. Dessa forma, as medidas para avaliação do modelo de *features* em LPSDs ainda é um tema de pesquisa pouco explorado. Isso decorre devido do fato do conceito de LPSD ter sido criado recentemente (HALLSTEINSEN *et al.*, 2008), e muitas pesquisas nessa área ainda estarem em aberto.

4.5 Ferramentas para Avaliação da Qualidade do Modelo de *Features*

Em relação às ferramentas de suporte para avaliação da qualidade do modelo de *features* utilizando medidas, foram identificadas apenas ferramentas para avaliar o modelo de *features* de LPSs. O S.P.L.O.T. oferece, além do repositório de modelos de *features*, a funcionalidade de análise automática dos modelos de *features* a partir da coleta de algumas medidas (MENDONCA *et al.*, 2009a). Além disso, outras ferramentas que dão suporte a análise automática do modelo como FAMILIAR (ACHER *et al.*, 2013), FAMA (TRINIDAD *et al.*, 2008) e FeatureIDE (THÜM *et al.*, 2014), também suportam a coleta de algumas medidas estruturais do modelo. No entanto, o propósito dessas ferramentas é na modelagem da variabilidade do modelo de *features*, e não na avaliação da qualidade. Além disso, as ferramentas compilam apenas poucas medidas para auxiliar na verificação de tamanho, corretude e consistência do modelo de *features*.

Neste trabalho foram identificadas na literatura apenas duas ferramentas para modelagem do modelo de *features* de LPSDs. A ferramenta MOSKitt4SPL (ALFÉREZ *et al.*, 2014a)

e a ferramenta VariaMos (MAZO *et al.*, 2015) permitem a modelagem do modelo de *features* e a configuração de produtos da LPSDs. Essas ferramentas não suportam a avaliação dos modelos de *features* de LPSDs utilizando medidas.

Neste contexto, a automação da avaliação da qualidade do modelo de *features* em LPSs e LPSDs ainda é, portanto, uma lacuna a ser tratada. Não há suporte ferramental para coleta automática de um conjunto grande de medidas para avaliação do modelo de *features*.

4.6 Comparação dos Trabalhos

De acordo com os trabalhos relacionados, foi elaborada uma comparação da proposta da Tese de doutorado com os trabalhos identificados na literatura, ilustrada na Tabela 4. Para realizar a comparação foram utilizados os seguintes critérios: (i) Avaliação do modelo de *features* em LPSs: critério relacionado aos trabalhos que propõem uma abordagem ou um estudo para avaliação da qualidade do modelo de *features* em LPSs; (ii) Avaliação do modelo de *features* em LPSDs: critério relacionado aos trabalhos que propõem uma abordagem ou um estudo para avaliação da qualidade do modelo de *features* em LPSDs; (iii) Medidas para LPSs: critério que analisa quais estudos identificaram ou propuseram medidas para avaliar a qualidade do modelo de *features* em LPSs; (iv) Medidas para LPSDs: critério que analisa quais estudos identificaram ou propuseram medidas para avaliar a qualidade do modelo de *features* em LPSDs ; (v) Características de qualidade: critério relacionado aos estudos que avaliam a qualidade do modelo de *features* de LPSs e LPSDs baseado em características de qualidade; e (vi) Suporte ferramental: critério relacionado aos estudos que apresentam suporte ferramental para avaliação da qualidade do modelo de *features* em LPSs e LPSDs baseado em medidas.

Pela comparação realizada na Tabela 4 é possível verificar que nenhum dos trabalhos relacionados suporta a avaliação da qualidade do modelo de *features* de LPSDs e que poucos trabalhos identificam medidas para o modelo de *features* de LPSDs. Esse fato ocorre porque os estudos identificados de LPSDs apenas propõem medidas para avaliar uma abordagem específica. Também é possível visualizar que nenhum dos trabalhos relacionados comparados possui uma ferramenta para suportar a avaliação da qualidade do modelo de *features* utilizando medidas. Outros trabalhos discutidos na Seção 4.5, automatizam a coleta de algumas medidas para o modelo de *features* de LPSs, mas não suportam medidas para o modelo de *features* de LPSDs. Além disso, poucos trabalhos realizam estudos mais aprofundados utilizando medidas para o modelo de *features* de LPSs, e a maior parte das abordagens e estudos não são baseados em

Tabela 4 – Comparação dos trabalhos relacionados com a proposta da tese de doutorado.

Trabalhos Relacionados	Avaliação do Modelo de *Features* de LPSs	Avaliação do Modelo de *Features* de LPSDs	Medidas para LPSs	Medidas para LPSDs	Características de Qualidade	Suporte Ferramental
(ETXEBERRIA; SAGARDUI, 2008b)	√	×	×	×	×	×
(DEELSTRA *et al.*, 2009)	√	×	×	×	×	×
(THÖRN, 2010)	√	×	√	×	√	×
(BAGHERI E. E GASEVIC, 2011)	√	×	√	×	√	×
(MONTAGUD *et al.*, 2012)	√	×	√	×	√	×
(BERGER T. E GUO, 2014)	√	×	√	×	√	×
(CETINA *et al.*, 2009b)	×	×	×	√	×	×
(SALLER *et al.*, 2013)	×	×	×	√	×	×
(ALFÉREZ *et al.*, 2014a)	×	×	×	√	×	×

características de qualidade.

Dessa forma, esta tese de doutorado tem por objetivo atender a todos os critérios avaliados na Tabela 4 para suportar a avaliação da qualidade do modelo de *features* em LPSs tradicionais e dinâmicas da característica manutenibilidade.

4.7 Conclusões

Neste capítulo foram apresentados os trabalhos relacionados a esta Tese de doutorado. Foram identificadas lacunas em aberto e deficiências dos estudos no tema de avaliação da qualidade dos modelos de *features* em LPSs tradicionais e dinâmicas.

Especificamente, na análise dos trabalhos relacionados foi verificado que existem na literatura poucas abordagens para avaliação da qualidade do modelo de *features* em LPSs e nenhuma abordagem para avaliação do modelo de *features* em LPSDs. Também foi identificado que ainda existem poucos estudos utilizando medidas para avaliar o modelo de *features* em LPSs, e que há deficiências de validação das medidas devido ao fato dos estudos utilizarem poucos modelos de *features* e alguns estudos não utilizarem especialistas em LPSs. Existem ainda poucas medidas na literatura para avaliação do modelo de *features* em LPSDs. Além disso, não há compilação dessas medidas nos estudos identificados para avaliar o modelo de *features* de LPSs tradicionais e dinâmicas. Os estudos também não fornecem informações para coleta e análise dessas medidas e um suporte ferramental para coleta automática das medidas compiladas.

Nos Capítulos 5, 6, 7 e 8 são apresentadas as principais contribuições e os estudos experimentais desta Tese de doutorado para avaliação do uso das medidas.

5 IDENTIFICANDO MEDIDAS PARA AVALIAÇÃO DA QUALIDADE DO MODELO DE *FEATURES*

Neste Capítulo são apresentadas as atividades realizadas com a finalidade de elaborar um catálogo de medidas de qualidade que possa ser utilizado para suportar a avaliação da manutenibilidade do modelo de *features*. Este catálogo, denominado COfFEE (*CatalOg of measures for Feature modEl quality Evaluation*), também é descrito neste capítulo.

Na Seção 5.1 é apresentado um mapeamento sistemático realizado a fim de identificar medidas de qualidade para avaliação da manutenibilidade do modelo de *features* em LPSs. Na Seção 5.2 é descrita uma revisão da literatura executada com o objetivo de identificar medidas de qualidade para avaliação da manutenibilidade do modelo de *features* em LPSDs. Na Seção 5.3 é discutida uma revisão por pares realizada com a finalidade de validar as medidas descritas nas seções 5.1 e 5.2. Na Seção 5.4 é apresentado o catálogo COfFEE, o qual é composto por 40 medidas de qualidade para avaliação da manutenibilidade do modelo de *features*, as quais resultam da revisão por pares discutida na Seção 5.3. Posteriormente, uma avaliação de uso das medidas do catálogo COfFEE é apresentada na Seção 5.5. Por fim, na Seção 5.6 são apresentadas as conclusões deste capítulo.

5.1 Identificando Medidas para LPSs

Esta seção apresenta um mapeamento sistemático realizado com o objetivo de identificar um conjunto de medidas de qualidade que pode ser utilizado para suportar a avaliação da manutenibilidade de modelos de *features* em LPSs, o qual foi publicado em (BEZERRA *et al.*, 2015). Um mapeamento sistemático é um tipo de revisão sistemática da literatura (PETERSEN *et al.*, 2008) usado quando existem muitas questões de pesquisas e há a necessidade de se verificar um elevado número de documentos relacionados com o tema em investigação. Muitas vezes, o mapeamento sistemático proporciona um resumo visual, que é um mapa de seus resultados (PETERSEN *et al.*, 2008). Os estudos de mapeamento sistemático em engenharia de software têm sido recomendados principalmente para áreas de pesquisa nas quais faltam estudos primários relevantes e de alta qualidade (KITCHENHAM; CHARTERS, 2007).

Por meio do mapeamento sistemático realizado, inicialmente, procurou-se identificar as características e subcaracterísticas de qualidade que são relevantes para a avaliação do modelo de *features*. Em seguida, buscou-se catalogar as medidas de qualidade encontradas na literatura com a finalidade de suportar a avaliação do modelo de *features*, além de averiguar se

essas medidas possuem ou não procedimentos operacionais de coleta e análise. A partir deste mapeamento, foi possível compilar um catálogo de medidas de qualidade que pode ser utilizado na avaliação da manutenibilidade de modelos de *features*.

5.1.1 Método de Pesquisa

O protocolo utilizado para orientar a execução do mapeamento sistemático apresentado neste capítulo baseou-se nas orientações definidas por Petersen et al. (PETERSEN *et al.*, 2008). O protocolo é composto pelas seguintes etapas: definição de questões de pesquisa, busca de documentos relevantes, triagem de documentos, classificação dos estudos e extração de dados.

Para o mapeamento sistemático foram definidas as seguintes questões de pesquisas (QPs):

- QP1: Que medidas têm sido utilizadas para avaliar a qualidade do modelo de *features* em LPSs?

As seguintes questões de pesquisa secundárias também foram definidas como relevantes para se atingir o objetivo do mapeamento:

- QP2: Quais são as características de qualidade que têm sido utilizadas na avaliação de modelos de *features*?
- QP3: Quais são as subcaracterísticas de qualidade que têm sido utilizados para avaliar a qualidade dos modelos de *features*?
- QP4: Quais são os modelos de qualidade que utilizam medidas para avaliação da qualidade do modelo de *features*?
- QP5: Quais as notações utilizadas para representar o modelo de *features*?
- QP6: Quais as ferramentas utilizadas para suportar a avaliação da qualidade do modelo de *features*?

As seguintes bibliotecas de pesquisa foram utilizadas: *IEEE Computer Society Digital Library*; *ACM Digital Library*; *Science Direct*; e *Springer Link*. Essas bibliotecas foram escolhidas porque são fontes confiáveis e são amplamente utilizadas por outros estudos na literatura. Para realizar uma busca automática nas bibliotecas digitais selecionadas foi utilizada a seguinte *string* de busca:

((*Quality* OR *Attribute* OR *Metric* OR *Measure* OR *Characteristic*) AND ("*Feature Model*" OR "*Feature Diagram*") AND ("*Product Line*" OR "*Product Family*") AND ("*Quality Evaluation*" OR "*Quality Assessment*"))

Foram realizadas buscas nessas bibliotecas até o dia 31 de janeiro de 2014. Logo, os artigos encontrados foram publicados e indexados antes desta data. Vale ressaltar que alguns trabalhos selecionados por este mapeamento já eram bastante populares (MONTAGUD; ABRAHãO, 2009; MONTAGUD *et al.*, 2012) na comunidade de LPS.

Para a seleção dos artigos foram definidos critérios de inclusão e exclusão. Os estudos que atenderam a pelo menos um dos seguintes critérios de inclusão foram considerados:

- Artigos que apresentam medidas de qualidade para avaliar modelos de *features*;
- Artigos que apresentam modelos de qualidade para avaliar modelos de *features*; e
- Artigos que apresentam características, subcaracterísticas ou atributos de qualidade relacionados ao modelo de *features*.

Os estudos que atenderam a pelo menos um dos seguintes critérios de exclusão foram removidos:

- Artigos que não estão relacionados a LPS;
- Artigos que não apresentam medidas relacionadas ao modelo de *features*; e
- Artigos introdutórios de edições especiais e livros.

Durante a leitura dos resumos de todos os artigos encontrados inicialmente, foram aplicados os critérios de inclusão e exclusão. Em seguida, os artigos resultantes foram lidos por completo e os filtros foram aplicados novamente. Os artigos que resultaram deste processo compuseram os estudos primários. O passo seguinte consistiu na extração e análise dos dados relevantes dos estudos primários. Para guiar a extração de dados foram examinadas as questões de pesquisa e definidos os itens para guiar o mapeamento sistemático, conforme ilustrado na Tabela 5.

Também foram utilizados os tipos de classificação de facetas propostos por Petersen *et al.* (2008) para classificar os estudos primários. A classificação dos estudos incluiu duas facetas. A primeira faceta é estruturada em termos de questões de pesquisa. A segunda considera o tipo de pesquisa. Para isso, foi utilizada a classificação das abordagens de pesquisa proposta por Wieringa et al. (WIERINGA *et al.*, 2006):

- **Pesquisa de Validação**: envolve técnicas novas, que ainda não foram implementadas na prática, como, por exemplo, as experiências ou trabalhos realizados em laboratório;
- **Pesquisa de Avaliação**: envolve técnicas já implementadas na prática e uma avaliação da técnica é apresentada em termos de benefícios e desvantagens;
- **Solução Proposta**: uma solução para um determinado problema é proposta. A solução

Tabela 5 – Formulário de extração de dados.

QP	Item	Respostas Possíveis
	Faceta por tipo de pesquisa	Pesquisa de Validação; Pesquisa de Avaliação; Solução Proposta; Artigos Filosóficos; Artigos de Opinião; Artigos de Experiência
QP1	Medidas	Extrair medidas a partir de artigos que são relacionados com a avaliação do modelo de *features* em LPSs (e.g. Complexidade Ciclomática, Número de pontos de variação)
QP1	Medidas Especificadas	( ) Sim ( ) Não
QP2	Características de Qualidade	Extrair características de qualidade que estão relacionadas ao modelo de *features* (e.g., Manutenibilidade, Confiabilidade, Eficiência de Desempenho)
QP3	Subcaracterísticas/ Atributos de Qualidade	Extrair subcaracterísticas/atributos de qualidade que estão relacionados ao modelo de *features* (e.g., Acurácia, Consistência)
QP4	Modelo de Qualidade	( ) Sim ( ) Não
QP5	Notação do modelo de *features*	Extrair notações do modelo de *features* (e.g., FODA, FORM, Modelo de *Features* Estendido)
QP6	Ferramentas	( ) Sim ( ) Não

pode ser qualquer técnica nova ou uma extensão significativa de uma técnica existente. Um pequeno exemplo ou uma boa linha de argumentação mostra os potenciais benefícios e a aplicabilidade da solução;

- **Artigos Filosóficos**: estes artigos esboçam uma nova maneira de olhar as coisas existentes, estruturando um campo de conhecimento sob a forma de uma taxonomia ou estrutura conceitual;
- **Artigos de Opinião**: estes artigos expressam a opinião pessoal de alguém, se uma determinada técnica é boa ou má, ou como algo deve ser realizado. Eles não dependem de trabalhos relacionados e de metodologias de pesquisa; e
- **Artigos de Experiência**: artigos de experiência explicam o que e como algo foi executado na prática. Estes artigos refletem as experiências pessoais dos autores.

5.1.2 Resultados do Mapeamento

Inicialmente, o processo de busca de artigos foi realizado utilizando a *string* de pesquisa em bibliotecas definidas previamente. O passo seguinte para a seleção de estudos primários foi aplicar filtros de seleção sobre os resultados da busca inicial utilizando os critérios de inclusão e exclusão definidos. A seleção dos estudos primários utilizou três filtros de seleção:

- **Filtro 1**: Refinar a busca excluindo artigos duplicados;

- **Filtro 2**: Aplicar os critérios de seleção de inclusão e exclusão, definidos anteriormente, na leitura do resumo e título; e
- **Filtro 3**: Aplicar novamente os critérios de seleção de inclusão e exclusão na leitura de todo o artigo.

A Tabela 6 apresenta os resultados obtidos após a execução de cada filtro. Durante a aplicação do primeiro filtro foram encontrados três trabalhos duplicados. A aplicação do segundo filtro resultou em 40 artigos. Por fim, após a aplicação do terceiro filtro foram selecionados 17 trabalhos.

Tabela 6 – Resultados da busca e seleção dos estudos.

Fonte	**Resultados da Busca**	**Filtro 1**	**Filtro 2**	**Filtro 3**
IEEE	22	20	11	3
ACM	16	16	8	4
Springer	19	18	11	7
Science Direct	22	2	10	3
Total	**79**	**56**	**40**	**17**

A distribuição dos estudos primários por ano é apresentada na Figura 11. Cada linha de tendência representa um determinado filtro de seleção e a distribuição dos artigos ao longo dos anos. Recentemente, o número de estudos publicados aumentou rapidamente, como apresentado na Figura 11, o que justifica a necessidade de novas pesquisas empíricas nesta área. Outra evidência para este fato, é o pequeno número de artigos selecionados pelo mapeamento sistemático após a aplicação dos três filtros. No entanto, a linha de tendência ascendente indica que o número de trabalhos nesta área tem aumentado nos últimos anos.

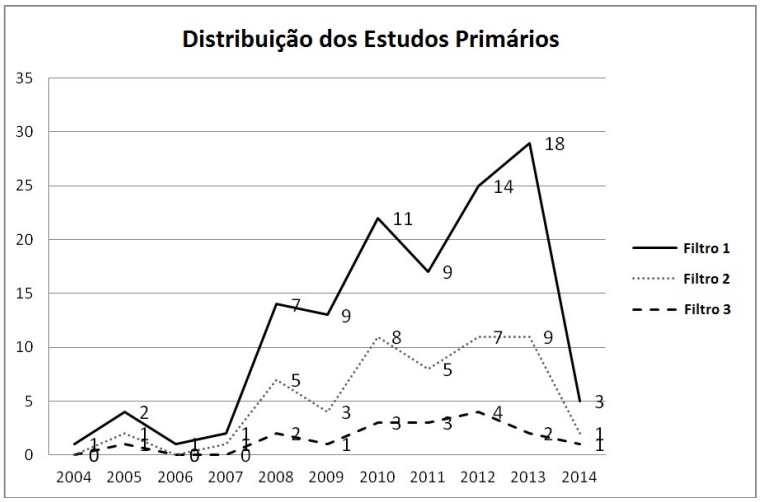


Figura 11 – Distribuição dos estudos primários por ano.

Em relação à distribuição dos estudos primários por tipo de faceta, foram obtidos os seguintes resultados: Pesquisa de Avaliação (35%), Proposta de Solução (41%) e Artigos Filosóficos (18%). A maioria dos estudos selecionados apresentaram medidas, características de qualidade ou modelos de qualidade para avaliar o modelo de *features* (JANAKIRAM; RAJASREE, 2005; ZHANG *et al.*, 2010; LEE; KANG, 2010; BELATEGI *et al.*, 2011; GONZALEZ-HUERTA *et al.*, 2012; DUAN *et al.*, 2013; ETXEBERRIA; SAGARDUI, 2008a). Alguns trabalhos classificados como Pesquisa de Avaliação propõem novos experimentos para validar uma determinada solução, muitas vezes usando casos na indústria (BAGHERI E. E GASEVIC, 2011; PATZKE *et al.*, 2012; BAGHERI *et al.*, 2012; ZHANG *et al.*, 2014; WHITE *et al.*, 2010; WHITE *et al.*, 2014). Os artigos encontrados que foram classificados como sendo do tipo Artigos Filosóficos, foram aqueles que apresentaram revisões sistemáticas. Foram encontradas três revisões sistemáticas (MONTAGUD; ABRAHãO, 2009; CHEN; BABAR, 2011; MONTAGUD *et al.*, 2012). A primeira revisão investiga a avaliação da qualidade em LPSs, a segunda revisão estuda a utilização de medidas para a avaliação de LPSs, enquanto a terceira revisão envolve a análise do gerenciamento de variabilidades em LPSs. Apenas um artigo foi classificado como Artigo de Opinião (ETXEBERRIA *et al.*, 2008). Este artigo apresenta uma comparação de métodos para avaliar a qualidade em LPSs.

Após a seleção dos artigos, foi elaborado um mapa sistemático dos estudos por facetas. Foram consideradas duas facetas: o tipo pesquisa e as questões de pesquisa. Este mapa é apresentado na Figura 12.

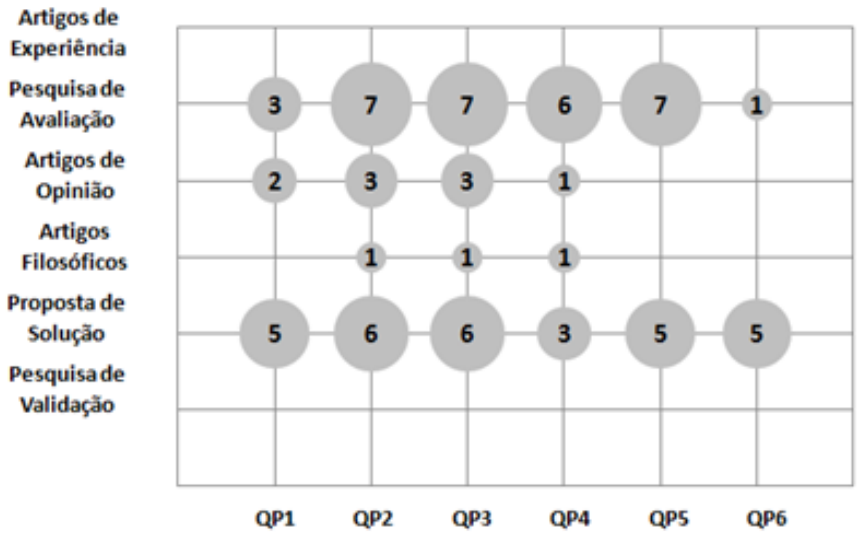


Figura 12 – Distribuição dos artigos selecionados por tipo de pesquisa e questão de pesquisa.

Pode-se observar, por meio da Figura 12, que não foram selecionados artigos dos tipo

Artigos de Experiência e Pesquisa de Validação. Portanto, a linha no gráfico da Figura 12 relativas a esses dois tipos não são exibidas. O mapa sistemático da Figura 12 indica que os estudos selecionados se concentram em dois tipos de pesquisa: Pesquisas de Avaliação e Propostas de Solução. Estes resultados respondem parcialmente a questão de pesquisa QP1. É importante destacar que alguns dos trabalhos selecionados apresentam medidas para avaliar a qualidade do modelo de *features*. No entanto, nenhuma destas pesquisas descreve os procedimentos operacionais dessas medidas.

Além dos artigos selecionados a partir do mapeamento sistemático, foi incluído o estudo realizado por Berger e Guo (BERGER T. E GUO, 2014), que identifica e propõe medidas relacionadas ao modelo de *features* em LPSs. Este artigo foi apresentado em uma conferência em Janeiro de 2014, período de execução do mapeamento. No entanto, ele foi publicado após a execução do mapeamento sistemático. Devido a relevância do artigo para o nosso estudo, ele foi adicionado.

Após analisar a faceta tipos de pesquisa, o passo seguinte foi procurar explorar a faceta questões de pesquisa. A Tabela 10 consolida os resultados que podem ser utilizados para responder as questões de pesquisa QP1, QP2 e QP3. Esta tabela apresenta as características, subcaracterísticas e medidas de qualidade extraídas dos trabalhos selecionados. As características de qualidade extraídas foram classificadas de acordo com a norma SQuaRE (ISO/IEC, 2011), a qual define oito características de qualidade para um produto de software: Adequação Funcional, Confiabilidade, Usabilidade, Eficiência de Performance, Manutenibilidade, Portabilidade, Segurança e Compatibilidade. As subcaracterísticas de qualidade encontradas nos artigos selecionados (ver Tabela 10) estão relacionadas a fatores que podem afetar a qualidade do modelo de *features* de LPSs. Estas subcaracterísticas podem ser avaliadas a partir de medidas relacionadas a elas. As medidas encontradas podem ser combinadas para avaliar a qualidade do modelo de *features* ou para derivar novas medidas de qualidade. Pode-se observar que a maioria das subcaracterísticas de qualidade extraídas estão relacionadas com a característica manutenibilidade (ver Tabela 11), indicando que essa característica possui vários aspectos que afetam a qualidade do modelo de *features*. As outras características de qualidade que também foram encontradas no mapeamento sistemático realizado foram: Adequação Funcional, Usabilidade, Portabilidade, Eficiência de desempenho, Confiabilidade e Segurança.

A questão de pesquisa QP4 foi respondida por meio dos estudos que propõem modelos de qualidade e que se concentram na avaliação do modelo de *features*. Tais modelos de

qualidade poderiam ser representados por métodos, abordagens, ou processos para avaliar o modelo de *features*. Mais precisamente, 61% dos artigos selecionados apresentaram modelos para avaliação da qualidade do modelo de *features* em LPSs (BENAVIDES *et al.*, 2007; JANAKIRAM; RAJASREE, 2005; ZHANG *et al.*, 2010; LEE; KANG, 2010; BELATEGI *et al.*, 2011; GONZALEZ-HUERTA *et al.*, 2012; DUAN *et al.*, 2013; ETXEBERRIA; SAGARDUI, 2008a; BAGHERI E. E GASEVIC, 2011; WHITE *et al.*, 2010; ETXEBERRIA *et al.*, 2008). Por outro lado, 39% dos trabalhos propuseram medidas, ou discutiram características e/ou subcaracterísticas de qualidade que seriam relevantes para avaliar o modelo de *features*, porém, estas não estavam associadas a um modelo de qualidade. Contudo, nenhum dos trabalhos que apresentaram medidas de qualidade descreveu os procedimentos operacionais para estas medidas. Alguns artigos apresentaram apenas as fórmulas de cálculo das medidas, outros somente as descrições das medidas, e outros ainda somente mencionaram os nomes das medidas. A ausência dessas informações dificulta a coleta e análise dessas medidas de forma padronizada, possibilitando que o engenheiro de domínio interprete as medidas de forma equivocada, proporcionando erros de coleta e análise.

Pode-se observar, por meio da Tabela 10, que os artigos selecionados neste mapeamento sistemático mencionam algumas subcaracterísticas de qualidade que não estão presentes na norma SQuaRE, são elas: Complexidade Cognitiva, Extensibilidade, Flexibilidade, Complexidade Estrutural e Variabilidade. Estas subcaracterísticas estão destacadas em negrito na Tabela 10.

Um fator que pode afetar a avaliação da qualidade do modelo de *features* é o tipo de notação utilizada. Diversas notações para representar as *features* e a variabilidade de uma LPS têm sido propostas, como, por exemplo, pode-se citar: FODA (KANG *et al.*, 1990), FORM (KANG *et al.*, 1998), FeatuRSEB (GRISS *et al.*, 1998), Modelo de *Features* baseado em Cardinalidade (CZARNECKI *et al.*, 2004), Modelo de *Features* Estendido (BENAVIDES *et al.*, 2007) e Linguagem de Variabilidade Comum (FLEUREY *et al.*, 2011).

A questão de pesquisa QP5 foi incluída para identificar as principais notações utilizadas para representar os modelos de *features*. Os resultados mostram que 29% dos trabalhos selecionados não indicam a notação em que o modelo de *features* é representado. Por outro lado, 24% dos trabalhos representam o modelo de *features* na notação FODA (JANAKIRAM; RAJASREE, 2005; BELATEGI *et al.*, 2011; BAGHERI *et al.*, 2012; ZHANG *et al.*, 2014; WHITE *et al.*, 2014). A segunda notação mais frequentemente foi o Modelo de *Features* Estendido

(ETXEBERRIA; SAGARDUI, 2008a; ZHANG *et al.*, 2014; WHITE *et al.*, 2010; WHITE *et al.*, 2014), utilizado em 19% dos artigos. Esta notação incorpora requisitos não-funcionais no modelo. As demais notações mencionadas foram: FORM (JANAKIRAM; RAJASREE, 2005; ZHANG *et al.*, 2010), modelo de *features* baseado em Cardinalidade (GONZALEZ-HUERTA *et al.*, 2012; BAGHERI E. E GASEVIC, 2011), FeatuRSEB (ETXEBERRIA; SAGARDUI, 2008a), e Linguagem de Variabilidade Comum (DUAN *et al.*, 2013). No entanto, nenhum dos trabalhos discutiu sobre a influência da notação do modelo de *features* em sua avaliação de qualidade.

A questão de pesquisa QP6 diz respeito à identificação de ferramentas cujo objetivo seja auxiliar a avaliação da qualidade do modelo de *features*. Essas ferramentas são importantes para a coleta e avaliação das medidas, contribuindo para tornar mais precisa a análise das medidas. Identificou-se que apenas 39% dos artigos mencionaram uma ferramenta de suporte (ZHANG *et al.*, 2010; BAGHERI E. E GASEVIC, 2011; PATZKE *et al.*, 2012; BAGHERI *et al.*, 2012; WHITE *et al.*, 2010; WHITE *et al.*, 2014; BERGER T. E GUO, 2014). Assim, pode-se observar que ainda são necessários esforços para construir ferramentas de apoio à avaliação da qualidade do modelo de *features* com base em medições.

5.2 Identificando Medidas para LPSDs

Os modelos de *features* utilizados em LPSDs possuem algumas especificidades, como, por exemplo, a necessidade de representar as adaptações de contexto, uma vez que a variabilidade é dinâmica, podendo ocorrer em tempo de execução. Por esse motivo, é importante analisar as características e subcaracterísticas de qualidade, específicas para este cenário, que podem influenciar a avaliação dos modelos de *features* em LPSDs. Além disso, é fundamental procurar identificar medidas de qualidade que sejam específicas para modelos de *features* de LPSDs. Dessa forma, foi realizada uma revisão da literatura complementar ao mapeamento sistemático apresentado na Seção 5.1 com a finalidade de identificar características, subcaracterísticas e medidas de qualidade voltadas especificamente para modelos de *features* de LPSDs. Essa revisão da literatura não foi realizada de forma sistemática, como o estudo anterior.

Para realizar a busca dos artigos nas bibliotecas foi utilizada a seguinte *string* de busca:

((*Quality* OR *Attribute* OR *Metric* OR *Measure* OR *Characteristic*) AND ("*Feature Model*"OR "*Feature Diagram*") AND ("*Dynamic Software Product Line*"))

As bibliotecas de busca selecionadas foram as mesmas utilizadas no mapeamento

sistemático descrito na Seção 5.1: *IEEE*, *ACM*, *Springer* e *Science Direct*. A busca foi realizada até 31 de outubro de 2015 considerando todos os anos anteriores. Os resultados obtidos nessa busca são apresentados na Tabela 7.

Tabela 7 – Resultados da busca e seleção de estudos voltados para LPSDs.

Fonte	**Resultados da Busca**	**Filtro 1**	**Filtro 2**	**Filtro 3**
IEEE	23	23	16	3
ACM	28	27	10	7
Springer	18	17	8	4
Science Direct	21	19	11	5
Total	**90**	**86**	**45**	**19**

Para selecionar os artigos foram utilizados três filtros, conforme definido na Seção 5.1. Os critérios de inclusão e exclusão também foram os mesmos definidos no mapeamento sistemático apresentado na Seção 5.1. Os artigos selecionados de acordo com cada filtro são ilustrados na Tabela 7. Após a aplicação dos três filtros, foram selecionados 19 artigos para extração de características, subcaracterísticas e medidas de qualidade específicas para o modelo de *features* de LPSDs. A Tabela 10 apresenta as características, subcaracterísticas e medidas de qualidades extraídas dos artigos selecionados e suas respectivas referências. Analisando a Tabela 10, pode-se observar que todas as características de qualidade mencionadas nos artigos selecionados estão presentes na norma SQuaRE (ISO/IEC, 2014). Pode-se notar ainda que as mesmas características de qualidade identificadas para o modelo de *features* de LPSs foram identificadas para o modelo de *features* de LPSDs (Adequação Funcional, Manutenibilidade, Portabilidade, Confiabilidade, Eficiência de Desempenho, Segurança), com exceção de Usabilidade. No entanto, algumas subcaracterísticas encontradas não fazem parte da norma SQuaRE (ISO/IEC, 2014), são elas: Flexibilidade, Variabilidade Dinâmica, Consistência, Estabilidade, Tolerância a Falhas e Não Repúdio. Essas subcaracterísticas foram destacadas em negrito na Tabela 10.

Durante esta revisão da literatura, foram identificadas também medidas de qualidade específicas para LPSDs. Algumas dessas medidas levam em conta as mudanças de contexto (e.g., Número de eventos de contexto problemáticos, Número de condições de contexto que poderão ser afetadas pelo aparecimento de eventos de contexto, Número de adaptações de contexto, Número de *features* ativadas no modelo de *features*, Número de *features* desativadas no modelo de *features*). Outras medidas levam em conta as atividades relacionadas ao tempo de

execução (e.g., Tempo de reconfiguração dinâmica, Latência, Diferença de tempo entre eventos problemáticos de contexto). Já outras estão relacionadas à confiabilidade (e.g., Número de *features* mortas, Número de configurações mortas, Núcleo de contextos que estão sempre ativos). Algumas medidas identificadas se repetiram nas duas revisões da literatura (e.g., Número de *features*, Número de *features* opcionais, Número de *features* mandatórias, Número de Grupos XOr (NGXOr); Número de Grupos Or (NGOr), Número de pontos de Variação, Número de configurações válidas). De forma semelhante ao que foi observado no mapeamento sistemático, a característica de qualidade mais utilizada na avaliação dos modelos de *features* em LPSDs foi a Manutenibilidade. No entanto, verifica-se também uma preocupação dos estudos com as características de Adequação Funcional, Confiabilidade e Eficiência de Desempenho.

Tabela 8 – Medidas mensuráveis específicas para o modelo de *features* de LPSDs.

Medidas	**Descrição**	**Referência**
Features de Contexto	Representam variantes contextuais específicas de uma *feature* que só são relevantes em contextos particulares	(CAPILLA *et al.*, 2014a; ALFÉREZ *et al.*, 2014a; MENS *et al.*, 2016)
Features Ativadas	Representam *features* de contexto que são ativadas no modelo de *features* quando uma determinada adaptação de contexto é ativada	(CAPILLA *et al.*, 2014a; ALFÉREZ *et al.*, 2014a)
Features Desativadas	Representam *features* de contexto que são desativadas no modelo de *features* quando uma determinada adaptação de contexto é ativada	(CAPILLA *et al.*, 2014a; ALFÉREZ *et al.*, 2014a)
Adaptações de Contexto	A adaptação de contexto é a capacidade do sistema para reunir informações sobre o domínio com o qual compartilha uma interface, avalia esta informação, e altera o seu comportamento de acordo com a situação atual, e é modelado no modelo de *features*	(CETINA *et al.*, 2009a; CAPILLA *et al.*, 2014a; ALFÉREZ *et al.*, 2014a; MENS *et al.*, 2016)
Restrições de Contexto	Em relação às restrições de contexto, os modelos de *features* e pontos de variação descrevem um caminho possível para representar a variabilidade estrutural com dependências, como as as retrições “requires” e “excludes”. As *features* de contexto podem ter dependências estabelecidas com outras *features*. Abordagens podem usar os *features* de contexto, onde *features* de contexto e que não possui contexto são enredadas no mesmo modelo de *features*	(MENS *et al.*, 2016)

Nessa revisão não foram identificados artigos que propõem abordagens e métodos para avaliação da qualidade do modelo de *features* em LPSDs. Apenas foram apresentadas algumas características, subcaracterísticas e medidas relevantes neste cenário. As medidas identificadas muitas vezes não possuíam procedimentos operacionais. Além disso, poucas medidas são específicas para o modelo de *features* de LPSDs e, geralmente, concentram-se no contexto e na variabilidade dinâmica do modelo de *features*.

Apesar de algumas medidas específicas para o modelo de *features* de LPSDs terem sido identificadas, a maioria dessas medidas não são mensuráveis, devido a ausência de informações, como, por exemplo: o Tempo de reconfiguração dinâmica, a Latência, dentre outras.

Mais precisamente, apenas cinco medidas são mensuráveis. Essas cinco medidas mensuráveis são apresentadas na Tabela 8.

Entretanto, as cinco medidas descritas na Tabela 8 são computadas para um determinado contexto e não para o modelo de *features* como um todo (incluindo todas as adaptações de contextos). Por esta razão, essas medidas foram adaptadas com a finalidade de produzir um único valor para o modelo de *features* como um todo, ou seja, levando em consideração todas as adaptações de contexto representadas no modelo. Essa adaptação produziu quatro novas medidas, as quais são ilustradas na Tabela 9.

Tabela 9 – Adaptações das medidas específicas para modelos de *features* de LPSDs.

Nome da Medida	**Descrição da Medida**
Features de Contexto nas Adaptações de Contexto (CFCA)	Número de *features* que estão sempre presentes nas adaptações de contexto ativadas do modelo de *features*
Features de Contexto nas Restrições (CFC)	Número de *features* que estão presentes nas adaptações de contexto e nas restrições do modelo de *features*
Features Ativadas por Adaptação de Contexto (AFCA)	Número de *features* ativadas em cada adaptação de contexto / Número de adaptações de contexto
Features Desativadas por Adaptação de Contexto (DFCA)	Número de *features* desativadas em cada adaptação de contexto / Número de adaptações de contexto

A Tabela 10 apresenta a consolidação das características, subcaracterísticas e medidas identificadas tanto no mapeamento sistemático discutido na Seção 5.1.1 quanto na revisão da literatura não-sistemática apresentada nesta seção.

Tabela 10 – Características de qualidade, subcaracterísticas de qualidade e medidas extraídas a partir de artigos da literatura - adaptado de (BEZERRA *et al.*, 2015).

Características	**Subcaracterísticas**	**Medidas**	**Referências**
Adequação Funcional	Corretude Funcional	*Precision*; *Recall*; *F-measure*; Taxa de Falso Positivo (RPF); Nível de Aceitabilidade (NAcce); Frequência de Observação do Contexto (FCO)	(BAGHERI *et al.*, 2012; LEE *et al.*, 2012; ALFÉREZ *et al.*, 2014a)
	Completude Funcional	-	(ZHANG *et al.*, 2010)
Manutenibilidade	Analisabilidade	Número de *features* folhas (NLeaf); Número de Mudanças (NC); Número de Features Adicionadas (NAddF); Número de Features Removidas (NRemF); Número de Features Atualizadas (NUpF); Número de Features Críticas (NCF); Número de Features Importantes (NIF); Número de Features Usáveis (NUF)	(BAGHERI E. E GASEVIC, 2011; MONTAGUD *et al.*, 2012; SALLER *et al.*, 2013; MIZOUNI *et al.*, 2014; CAPILLA *et al.*, 2014a; QUINTON *et al.*, 2015)
	Modificabilidade	Flexibilidade da mudança (FM); Índice de manutenibilidade da *feature*(IMF)	(MONTAGUD; ABRAHãO, 2009; GONZALEZ-HUERTA *et al.*, 2012; ETXEBERRIA; SAGARDUI, 2008a; BAGHERI E. E GASEVIC, 2011; ZHANG *et al.*, 2014; CHEN; BABAR, 2011; MONTAGUD *et al.*, 2012; ETXEBERRIA *et al.*, 2008)
	Complexidade Cognitiva / Entendabilidade	Complexidade Cognitiva do Modelo de *Features* (CogC); Taxa de *Features* Booleanas (RSwitch); Taxa de *Features* com o Valor do Domínio Arbitrário (RData); Taxa de Features que tem o Valor Atribuído ou Pertence a comunalidade do modelo (RNone)	(BAGHERI E. E GASEVIC, 2011; PATZKE *et al.*, 2012; MONTAGUD *et al.*, 2012; BERGER T. E GUO, 2014; GREENWOOD *et al.*, 2011)
	Extensibilidade	Extensibilidade da Feature (FEX)	(MONTAGUD; ABRAHãO, 2009; JANAKIRAM; RAJASREE, 2005; ETXEBERRIA; SAGARDUI, 2008a; MONTAGUD *et al.*, 2012; ETXEBERRIA *et al.*, 2008)
	Flexibilidade	Flexibilidade da Configuração (FoC)	(BAGHERI E. E GASEVIC, 2011; CHEN; BABAR, 2011; MONTAGUD *et al.*, 2012; DUAN *et al.*, 2013; ZHANG *et al.*, 2014; MIZOUNI *et al.*, 2014)
	Modularidade	*Features* Dependentes Cíclicas Únicas (SCDF); *Features* Dependentes Cíclicas Múltiplas (MCDF)	(JANAKIRAM; RAJASREE, 2005)

Tabela 10 – Características de qualidade, subcaracterísticas de qualidade e medidas extraídas a partir de artigos da literatura - adaptado de (BEZERRA *et al.*, 2015).

Características	**Subcaracterísticas**	**Medidas**	**Referências**
	Reusabilidade	-	(MONTAGUD *et al.*, 2012)
	Complexidade Estrutural	Número de *Features* (NF); Número de *Features* Mandatórias (NM); Número de *Features* Top (NTop); Profundidade da Árvore (DT Max, DT Mean and DT Median); Complexidade Ciclomática (CyC); Complexidade da Configuração (ConfC); Complexidade das Restrições (ConsC); Complexidade Estrutural (SC); Complexidade Composta (ComC); Complexidade da Variabilidade (CVy); Complexidade da Variante (CVt); Restrições *Cross-tree* (CTC); Restrições Variáveis *Cross-tree* (CTCV); Taxa de Conectividade do Grafo (RCon); Densidade do Grafo (RDen); Coeficiente de Densidade da Conectividade (CoC); Número de *Features* Agrupadas (NGF); Número de Filhos por *Feature* (BF Max); Taxa de *Features* com um Valor Modelado Explícito (RDefaut); Taxa de *Features* Derivadas (RDerived); Taxa de *Features* com menos de uma Condição de Visibilidade (RVisibility); Número de Dependências de Restrições de Integridade (NCC)	(CETINA *et al.*, 2009b; GAMEZ *et al.*, 2011; GREENWOOD *et al.*, 2011; NASCIMENTO *et al.*, 2011; BAGHERI E. E GASEVIC, 2011; GAMEZ *et al.*, 2012; PARRA *et al.*, 2012; PATZKE *et al.*, 2012; MONTAGUD *et al.*, 2012; ANJORIN *et al.*, 2013; SALLER *et al.*, 2013; MIZOUNI *et al.*, 2014; ALFÉREZ *et al.*, 2014a; BERGER T. E GUO, 2014; MOENS; TURCK, 2014; CAPILLA *et al.*, 2014a; LOCHAU *et al.*, 2015)
	Variabilidade Estática	Número de *Features* Opcionais (NO); *Single Hotspot Features* (SHoF); *Multiple Hotspot Features* (MHoF); *Rigid Nohotspot Features* (RNoF); Número de *Features* Variáveis (NVF); Número de Configurações Válidas (NVC); Taxa de Variabilidade (RoV); Número de Grupos Or (NGOr); Nùmero de Grupos XOr (NGXOr); Taxa de *Features* Or (ROr); Taxa de *Features* XOr (RXOr); Número de Transições Geradas através das Configurações (NRC); Número de Configurações Válidas (NVC); Número de Configurações Inválidas (NIC)	(JANAKIRAM; RAJASREE, 2005; WHITE *et al.*, 2010; BAGHERI E. E GASEVIC, 2011; NASCIMENTO *et al.*, 2011; CHEN; BABAR, 2011; MONTAGUD *et al.*, 2012; PATZKE *et al.*, 2012; PARRA *et al.*, 2012; GAMEZ *et al.*, 2012; ANJORIN *et al.*, 2013; BERGER T. E GUO, 2014; ALFÉREZ *et al.*, 2014a; CAPILLA *et al.*, 2014a; LOCHAU *et al.*, 2015)

Tabela 10 – Características de qualidade, subcaracterísticas de qualidade e medidas extraídas a partir de artigos da literatura - adaptado de (BEZERRA *et al.*, 2015).

Características	**Subcaracterísticas**	**Medidas**	**Referências**
	Variabilidade Dinâmica	Número de Contextos (NC); Número de *Features* Desativadas (NDF); Número de *Features* Ativadas (NAF); *Features* de Contextos em Restrições (CFC); Número de Restrições de Contexto (NCC); Número de *Features* de Contextos (CF); Número de *Features* Ativadas por Contexto (AFCA); Número de *Features* Desativadas por Contexto (DFCA)	(CETINA *et al.*, 2009b; GREENWOOD *et al.*, 2011; CETINA *et al.*, 2013; SALLER *et al.*, 2013; MIZOUNI *et al.*, 2014; CAPILLA *et al.*, 2014a; ALFÉREZ *et al.*, 2014a; MURGUZUR *et al.*, 2014; QUINTON *et al.*, 2015; SANCHEZ *et al.*, 2015)
Usabilidade	Facilidade de Uso	-	(ZHANG *et al.*, 2014)
Eficiência de Desempenho	Acurácia	-	(ETXEBERRIA; SAGARDUI, 2008a; LEE; KANG, 2010; LEE *et al.*, 2012; GAMEZ *et al.*, 2012; ALFÉREZ *et al.*, 2014a; LOCHAU *et al.*, 2015; SANCHEZ *et al.*, 2015)
	Utilização de Recursos	-	(ZHANG *et al.*, 2010; GONZALEZ-HUERTA *et al.*, 2012)
	Escalabilidade	-	(MONTAGUD; ABRAHãO, 2009; GAMEZ *et al.*, 2011; MONTAGUD *et al.*, 2012; WHITE *et al.*, 2014; ALFÉREZ *et al.*, 2014a; MURGUZUR *et al.*, 2014)
	Comportamento em Relação ao Tempo	Tempo de Documentação (DT); Tempo quando uma *feature* foi incluída dentro do escopo do projeto (TISP); Latência (Lat); Tempo de Reconfiguração (RecT); Tempo de Binding (BinT); Memória Requerida (RM); Tempo de Resposta (RT); Tempo de Resposta dentro de Circunstância de *Stress* (RTSC); Tempo de *Frame* entre eventos de Contexto Problemático (FTPC); Média de Tempo de Resposta por Acurácia do Modelo de *Feature* (RTMA); Média do Tempo de Resposta por *Features* Mortas não Existentes (RTMDF); Média do Tempo de Resposta por Estabilidade (RTMS); Média do Tempo de Resposta por SemiVivacidade (MTRS); Tempo de Execução (RunT)	(CETINA *et al.*, 2009b; ZHANG *et al.*, 2010; BELATEGI *et al.*, 2011; GAMEZ *et al.*, 2011; ALFEREZ; PELECHANO, 2011; GREENWOOD *et al.*, 2011; NASCIMENTO *et al.*, 2011; MONTAGUD *et al.*, 2012; LEE *et al.*, 2012; PARRA *et al.*, 2012; GAMEZ *et al.*, 2012; SALLER *et al.*, 2013; ANJORIN *et al.*, 2013; ZHANG *et al.*, 2014; ALFÉREZ *et al.*, 2014a; CAPILLA *et al.*, 2014a; SANCHEZ *et al.*, 2015)
Portabilidade	Adaptabilidade	Adaptabilidade Estática da *Feature* (FSA); Adaptabilidade Dinâmica da *Feature* (FDA)	(MATINLASSI *et al.*, 2002; BAGHERI E. E GASEVIC, 2011; MIZOUNI *et al.*, 2014)

Tabela 10 – Características de qualidade, subcaracterísticas de qualidade e medidas extraídas a partir de artigos da literatura - adaptado de (BEZERRA *et al.*, 2015).

Características	**Subcaracterísticas**	**Medidas**	**Referências**
Confiabilidade	Disponibilidade	-	(MONTAGUD; ABRAHãO, 2009; ALFEREZ; PELECHANO, 2011; LEE *et al.*, 2012; PARRA *et al.*, 2012; NASCIMENTO *et al.*, 2011; CETINA *et al.*, 2013; MIZOUNI *et al.*, 2014; CAPILLA *et al.*, 2014a; SANCHEZ *et al.*, 2015)
	Consistência	Taxa de Consistência (CR), Número de Eventos de Contexto Problemáticos (NEPC); Número de Contextos Núcleos que estão sempre Ativos (NACA); Número de Configurações Mortas (NDC); Número de Features Mortas (NDF)	(MONTAGUD *et al.*, 2012; SALLER *et al.*, 2013; ALFÉREZ *et al.*, 2014a; ZHANG *et al.*, 2014; QUINTON *et al.*, 2015; LOCHAU *et al.*, 2015)
	Tolerância a Falhas	Número de Falhas de Reconfiguração (NRF)	(NASCIMENTO *et al.*, 2011; CETINA *et al.*, 2013; QUINTON *et al.*, 2015)
Segurança	Autenticidade	-	(ZHANG *et al.*, 2010; ZHANG *et al.*, 2014)
	Integridade	-	(ZHANG *et al.*, 2010; MONTAGUD *et al.*, 2012)
	Não Repúdio	-	(PARRA *et al.*, 2009; ALFÉREZ *et al.*, 2014a; SANCHEZ *et al.*, 2015)

5.3 Revisão por Pares

A Tabela 10 agrupa o conjunto das medidas identificadas por meio das revisões da literatura discutidas nas seções 5.2 e 5.1 com a finalidade de suportar a avaliação da qualidade dos modelos de *features*. Contudo, é fundamental verificar se as medidas estão corretas e se as relações entre essas medidas e as características e subcaracterísticas de qualidade estão adequadas. Com esta finalidade, foi realizada uma revisão por pares. A técnica de revisão por pares é um tipo de avaliação em que um produto de trabalho (documento, código, etc.) é examinado pelo seu autor e por um ou mais revisores, a fim de avaliar o seu conteúdo técnico e a sua qualidade. A revisão por pares foi escolhida por ser uma técnica amplamente utilizada tanto pelo meio acadêmico quanto pela indústria.

A revisão por pares foi conduzida com os seguintes objetivos:

- Avaliar se as características e subcaracterísticas de qualidade identificadas estão alinhadas com a nomenclatura da norma de qualidade SQuaRE;
- Avaliar se a relação entre características de qualidade, subcaracterísticas e medidas estão adequadas;
- Avaliar se as características, subcaracterísticas e medidas de qualidade estão em um nível de granularidade adequado; e
- Avaliar se as medidas identificadas estão corretas (em relação à sua semântica, fórmula de cálculo, etc).

Quatro especialistas realizaram a revisão por pares das medidas identificadas (e ilustradas na Tabela 10): dois especialistas em qualidade de produtos de software e dois especialistas em LPSs e LPSDs.

No processo de revisão das medidas identificadas foram encontradas inconsistências em relação a alguns aspectos, tais como: nível de granularidade das características e subcaracterísticas de qualidade (e.g., tamanho e profundidade do modelo foram identificadas como subcaracterísticas de qualidade, foram excluídas); nomenclatura não alinhada com a norma SQuaRE (e.g., Informações de Reuso foi identificado como subcaracterística de qualidade, porém, Reusabilidade é o termo correto); relações incorretas entre as características de qualidade (e.g., a subcaracterística de qualidade Precisão foi relacionado com a característica Adequação Funcional, porém, o correto é que a subcaracterística Precisão estivesse relacionada com a característica de Eficiência de Desempenho) e medidas (e.g., medidas que não avaliam o modelo de *features* como Complexidade do código e Complexidade dos ativos do núcleo).

Os problemas encontrados pelos revisores foram agrupados em quatro tipos:

- Tipo 1: Características e subcaracterísticas de qualidade não alinhadas com a nomenclatura da norma de qualidade SQuaRE;
- Tipo 2: Relacionamentos inadequados entre características de qualidade, subcaracterísticas e medidas;
- Tipo 3: Nível inadequado de granularidade e
- Tipo 4: Erros nas definições das medidas (em relação à sua semântica, formula de cálculo, etc).

A Figura 13 ilustrada a quantidade de problemas encontrados pelos revisores por tipo de problema.

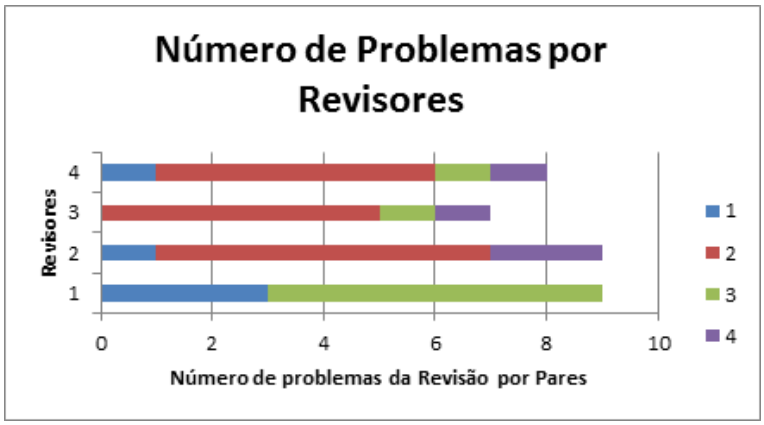


Figura 13 – Quantidade de problemas encontrados na revisão por pares por revisor e por tipo de problema nas medidas propostas para LPSs.

Na revisão das medidas específicas para LPSDs foram utilizados os mesmos critérios adotados anteriormente para as medidas voltadas para LPSs. Nesta revisão, foram identificadas inconsistências relacionadas à: nomenclatura da norma SQuaRE (e.g., subcaracterísticas que foram identificadas como Granularidade e Semi-Vivacidade foram excluídas), relacionamento das subcaracterísticas e medidas (e.g, a medida Número de Soluções Válidas que se encontrava relacionada à subcaracterística Acurácia, é a mesma medida que se encontra relacionada à subcaracterística Variabilidade Estática, logo foi excluída), nível de granularidade das subcaracterísticas de qualidade (e.g., Dependência foi identificada como subcaracterística, mas a medida de Dependência faz parte da subcaracterística Complexidade Estrutural), e medidas relacionadas ao modelo de *features* (e.g., foram excluídas medidas não relacionadas com o modelo de *features*). A quantidade de problemas identificados nesta revisão estão ilustrados na Figura 14. Todos os problemas encontrados na revisão por pares foram analisados e corrigidos.

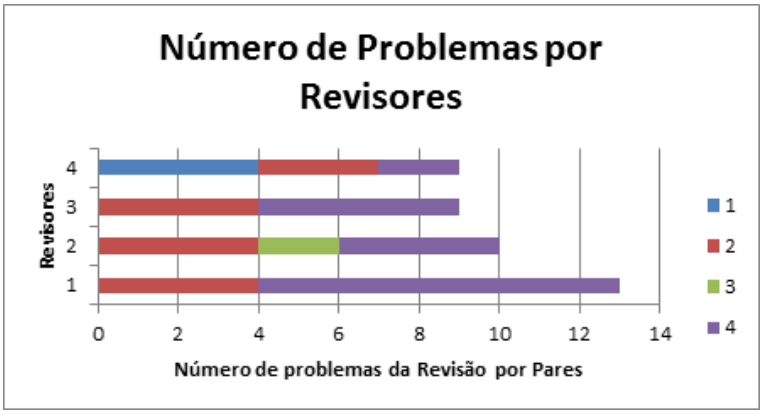


Figura 14 – Quantidade de problemas encontrados na revisão por pares por revisor e por tipo de problema nas medidas propostas para LPSDs.

5.4 O Catálogo COfFEE

Nesta seção é apresentado o catálogo de medidas, denominado COfFEE (*CatalOg of measures for Feature modEl quality Evaluation*), o qual é proposto nesta Tese e composto por 40 medidas de qualidade para avaliação da manutenibilidade do modelo de *features*. O COfFEE foi construído a partir das medidas identificadas pelas revisões da literatura (Seções 5.1 e 5.2) e validadas por meio da revisão por pares (Seção 5.3). Essas medidas foram ilustradas na Tabela 10.

O processo utilizado para a construção do catálogo COfFEE é descrito a seguir.

Após a revisão por pares chegou-se a um conjunto de medidas corretas e adequadamente relacionadas às características e subcaracterísticas de qualidade, as quais estão descritas na Tabela 10.

Contudo, muitas dessas medidas, apesar de serem mensuráveis, requerem informações adicionais, muitas vezes difíceis de se obter, para o seu cálculo.

Essas informações adicionais estão relacionadas a diferentes aspectos tais como mudanças, tempo, decisões do cliente, dentre outros. Como exemplos dessas medidas pode-se citar: Impacto da Mudança (IC), Flexibilidade da Mudança (FC), Índice de Manutenibilidade da *Feature* (MI), Número de mudanças (NC), Nível de aceitabilidade (NAcce), Frequência de observação do contexto (FCO), Número de *features* adicionadas (NAddF), Número de *features* removidas (NRemF), Número de *features* atualizadas (NUpF), Impacto Quantitativo de *Features* Variáveis Funcionais nos Atributos de Qualidade (QIFVF), Taxa de Consistência (RC), Taxa de *Features* Booleanas (*RSwitch*), Taxa de *Features* com o Valor do Domínio Arbitrário (*RData*), Taxa de *Features* que tem o Valor Atribuído ou Pertence a comunalidade do modelo (*RNone*),

Taxa de *Features* com um Valor Modelado Explícito (*RDefaut*), Taxa de *Features* Derivadas (*RDerived*), Taxa de *Features* com menos de uma Condição de Visibilidade (*RVisibility*), Tempo de Documentação (DT), Tempo quando uma *feature* foi incluída dentro do escopo do projeto (TISP), Latência (Lat), Tempo de reconfiguração (RecT), Tempo de *binding* (BinT), Memória requerida (RM), Tempo de resposta (RT), Tempo de resposta dentro de circunstância de *stress* (RTSC), Tempo de *Frame* entre eventos de Contexto Problemático (FTPC); Média de tempo de resposta por acurácia do modelo de *features* (RTMA); Média do tempo de resposta por *features* mortas não existentes (RTMDF), Média do tempo de resposta por estabilidade (RTMS), Média do tempo de resposta por semiVivacidade (MTRS) e Tempo de execução (RunT).

Adicionalmente, o cálculo de algumas medidas envolvem informações do contexto em tempo de execução. Como exemplo dessas medidas pode-se citar: Número de Reconfigurações (NRec), Número de Eventos de Contexto Problemático (NEPC), Número de configurações mortas (NDC), Número de *features* mortas (NDF) e Número de Falhas de Reconfiguração (NRF). Por fim, algumas medidas requerem, para o seu cálculo, informações relacionadas ao ciclo de vida e a evolução do modelo de *features*. Essas medidas são: *Precision*, *Recall*, *F-measure*, Taxa de falso positivo (RPF), Número de *features* críticas (NCF), Número de *features* importantes (NIF), Número de *features* usáveis (NUF), Número de transições geradas através das configurações (NRC), Valor da Importância da *Feature* (VFI), Complexidade da Variabilidade (CVy), Complexidade da Variante (CVt), Tempo de Documentação (DT), Tempo quando uma *Feature* foi Incluída no Escopo do Projeto (TISP), Adaptabilidade Estática da *Feature*(FSA) e Adaptabilidade Dinâmica da *Feature* (FDA).

Devido a essas dificuldades, decidiu-se por não incluir essas medidas no catálogo COfFEE. As medidas selecionadas para compor o catálogo COfFEE estão ilustradas na Tabela 11. Vale ressaltar que todas essas medidas estão relacionadas à característica de qualidade Manutenibilidade.

Tabela 11 – Catálogo COfFEE - Um catálogo de medidas de qualidade para suportar a avaliação da manutenibilidade do modelo de *features*.

Característica	**Subcaracterísticas**	**Medidas**
Manutenibilidade	Analisabilidade	Número de *features* folhas (NLeaf)
	Complexidade Cognitiva / Entendabilidade	Complexidade Cognitiva do Modelo de *Features* (CogC)
	Extensibilidade	Extensibilidade da Feature (FEX)
	Flexibilidade	Flexibilidade da Configuração (FoC)
	Modularidade	*Features* Dependentes Cíclicas Únicas (SCDF); *Features* Dependentes Cíclicas Múltiplas (MCDF)

Tabela 11 – Catálogo COfFEE - Um catálogo de medidas de qualidade para suportar a avaliação da manutenibilidade do modelo de *features*.

Característica	Subcaracterísticas	Medidas
	Complexidade Estrutural	Número de *Features* (NF); Número de *Features* Mandatórias (NM); Número de *Features* Top (NTop); Profundidade da Árvore (DT Max, DT Mean and DT Median); Complexidade Ciclomática (CyC); Complexidade Composta (ComC); Restrições *Cross-tree* (CTC); Restrições Variáveis *Cross-tree* (CTCV); Taxa de Conectividade do Grafo (RCon); Densidade do Grafo (RDen); Coeficiente de Densidade da Conectividade (CoC); Número de *Features* Agrupadas (NGF); Número de Filhos por *Feature* (BF Max)
	Variabilidade Estática	Número de *Features* Opcionais (NO); *Single Hotspot Features* (SHoF); *Multiple Hotspot Features* (MHoF); *Rigid Nohotspot Features* (RNoF); Número de *Features* Variáveis (NVF); Número de Configurações Válidas (NVC); Taxa de Variabilidade (RoV); Número de Grupos Or (NGOr); Número de Grupos XOr (NGXOr); Taxa de *Features* Or (ROr); Taxa de *Features* XOr (RXOr)
	Variabilidade Dinâmica	Número de Contextos (NC); Número de *Features* Desativadas (NDF); Número de *Features* Ativadas (NAF); *Features* de Contextos em Restrições (CFC); Número de Restrições de Contexto (NCC); Número de *Features* de Contextos (CF); Número de *Features* Ativadas por Contexto (AFCA); Número de *Features* Desativadas por Contexto (DFCA)

5.5 Aplicando o Catálogo COfFEE

A fim de avaliar a utilização do catálogo de COfFEE (apresentado na Tabela 11), aplicou-se as 40 medidas do catálogo em cinco modelos de *features*. Os modelos de *features* foram extraídos do repositório S.P.L.O.T.[1] (MENDONCA *et al.*, 2009a), um repositório de modelos de *features* mantidos pela comunidade acadêmica de LPS, e os modelos de *features* de LPSDs foram extraídos da literatura.

As fórmulas de cálculo das 40 medidas do catálogo COfFEE são apresentadas na Tabela 12. As fórmulas de cálculo das medidas estão descritas de acordo com as descrições dos artigos originais. Os procedimentos operacionais para cada medida do catálogo COfFEE são apresentados no Apêndice A.

Os procedimentos operacionais foram construídos com base nas descrições das medidas nos artigos identificados, e as medidas novas de acordo com a descrição dos especialistas.

[1] http://www.splot-research.org/

Para construção do intervalo de dados foram utilizados os dados das medidas dos *datasets* descritos no Capítulo 6, para definição dos limites máximo e mínimo. Já o procedimento de medição utiliza a ferramenta DyMMer, o qual foi desenvolvida nesta Tese e é detalhada no Capítulo 6.

Para aplicar as medidas foram identificados modelos de *features* de LPSs no repositório S.P.L.O.T. e de LPSDs na literatura. A escolha desses modelos foi baseada no domínio de aplicações móveis e/ou ubíquas. Os modelos de *features* de LPSs e LPSDs selecionados foram:

- *Strategy Mobile Game*: uma LPS de um jogo móvel para estratégia de múltiplos jogadores;
- *Mobile Media 2*: uma LPS para aplicações que manipulam fotos, música e vídeo de dispositivos móveis, tais como os telefones móveis;
- *Mobile Tourist Planner*: uma LPSD para aplicação de planejamento de turismo baseado na previsão do tempo;
- *MobiHome*: uma LPSD de aplicações adaptativas para acessar diversos serviços de uma *smarthome*; e
- *Service-Based Systems*: uma LPSD para sistemas de automação de casa baseados em serviços.

As medidas apresentadas na Tabela 12 foram selecionadas devido ao fato de serem de possível coleta. Estas medidas estão relacionadas à característica de qualidade de manutenibilidade. Os resultados dessas medidas para os 5 modelos de *features* selecionados são apresentados na Tabela 13. Apenas as medidas NAF, NAF, NDF, NCC, CF, CFC, AFCA e DFCA não foram coletadas para os modelos *Strategy Mobile Game* e *Mobile Media 2*, pois estas medidas são específicas para LPSDs e não para LPSs tradicionais. Para análise desses resultados, foram agrupadas as medidas para cada subcaracterística de manutenibilidade, de acordo com o catálogo COfFEE aparentado na Seção 11.

A subcaracterística de qualidade Analisabilidade está relacionada a medida Número de *Features* Folhas (NLeaf). Quanto maior for a NLeaf, maior é sua analisabilidade. Assim, o modelo de *features Mobile Media 2* tem maior analisabilidade (NLeaf = 32).

A subcaracterística de Modularidade está relacionada as medidas *Features* Dependentes Cíclicas Únicas (SCDF) e *Features* Dependentes Cíclicas Múltiplas (MCDF). Quanto menor o número de SCDF e MCDF, melhor é a modularidade. Portanto, o modelo de *features* com a melhor modularidade foram os modelos *Mobile Tourist Planner* e o *Mobi Home*.

Outras medidas relacionadas a subcaracterísticas de qualidade de manutenibilidade

Tabela 12 – Catálogo COfFEE: Fórmulas de cálculo das medidas.

Acrônimo	**Nome da Medida**	**Fórmula de Cálculo**
NF	Número de *Features*	Quantidade de *features* do modelo
NO	Número de *Features* Opcionais	Quantidade de *features* opcionais do modelo
NM	Número de *Features* Mandatórias	Quantidade de *features* mandatórias do modelo
NTop	Número de *Features* Top	Quantidade de *features* descendentes da raiz da árvore
NLeaf	Número de *Features* Folhas	Quantidade de *features* sem filhos da árvore
DT Max	Profundidade Máxima	Número de *features* do caminho mais longo a partir da raiz do modelo de *features*
DT Mean	Profundidade Média	Número de *features* do caminho médio a partir da raiz do modelo de *features*
DT Median	Profundidade Mediana	Número de *features* do caminho mediano a partir da raiz do modelo de *features*
CogC	Complexidade Cognitiva	Número de Pontos de Variação
FEX	Extensibilidade da *Feature*	NLeaf + SCDF + MCDF
FoC	Flexibilidade da Configuração	NO/NF
SCDF	Features Dependentes Cíclicas Únicas	∑(#*Features* participantes de restrições e que são filhas de pontos de variação com cardinalidade [1..1])
MCDF	Features Dependentes Cíclicas Múltiplas	∑(#*Features* participantes de restrições e que são filhas de pontos de variação com cardinalidade [1..*])
CyC	Complexidade Ciclomática	Quantidade de restrições de integridade
ComC	Complexidade Composta	$NF^2 + (NM^2 + 2*NOr^2 + 3*NXOr^2 + 3*NGF^2 + 3*R^2)/9$ R = NGF + CyC
NGF	Número de Agrupamento de *Features*	Número de agrupamentos de relações Or e XOr
CTCV	Restrições Variáveis de *Cross-Tree*	Número de variáveis distintas em restrições *cross-tree*
CTCR	Taxa de Restrições *Cross-Tree*	NFRI*/NF *Número de *features* envolvidas em restrições de integridade
RCon	Taxa de Conectividade do Grafo	Conectividade de grafo de dependência, o qual é a taxa de *features* que referenciam outras *features* (exceto pais) em suas restrições
RDen	Densidade do grafo	Média do número de *features* não-pais que são referenciadas em restrições
CoC	Coeficiente de Densidade de Conectividade	(Número de arestas)/NF
NVF	*Features* Variáveis	NA + NO
SHoF	*Single Hotspot Features*	Número de *features* filhas de pontos de variação com cardinalidade [1..1]
MHoF	*Multiple Hotspot Features*	Número de *features* filhas de pontos de variação com cardinalidade [1..*]
RNoF	*Rigid Nohotspot Features*	Número de *features* não filhas de pontos de variação
RoV	Taxa de Variabilidade	∑(Média do número de filhas dos nós)
NVC	Nùmero de Configurações Válidas	Número de possíveis configurações válidas do modelo de *features*
BF Max	Fator de Ramificação Máximo	Nùmero máximo de filhos por *feature*
NGOr	Número de Grupos Or	Número de pontos de variação com relacionamentos Or
NGXOr	Número de Grupos XOr	Número de pontos de variação com relacionamentos XOr
ROr	Taxa de *Features* Or	Taxa de *features* filhas de um relacionamento Or
RXOr	Taxa de *Features* XOr	Taxa de *features* filhas de um relacionamento XOr
NC	Número de Contextos	Número de contextos do modelo de *features*
NAF	Número de *Features* Ativadas	Número de *features* ativadas em cada contexto
NDF	Número de *Features* Desativadas	Número de *features* desativadas em cada contexto
NCC	Número de Restrições de Contexto	Número de restrições de um contexto
CF	Número de *Features* de Contextos	Número de *features* que estão sempre presentes nos contextos ativos do modelo de *features*
CFC	*Features* de Contextos em Restrições	Número de *features* que estão presentes nos contextos e nas restrições do modelo de *features*
AFCA	Número de *Features* Ativadas por Contexto	Número de *features* ativadas em cada contexto / NC
DFCA	Número de *Features* Desativadas por Contexto	Número de *features* desativadas em cada contexto / NC

não são apresentadas no modelo de qualidade da norma SQuaRE (ISO/IEC, 2011), apresentada no Capítulo 3.

As subcaracterísticas de manutenibilidade são específicas para avaliação do modelo de *features* (e.g. Extensibilidade, Flexibilidade, Complexidade Estrutural e Variabilidade).

Tabela 13 – Resultados das medidas para cada modelo de *features*.

Medidas	***Strategy Mobile Game***	***Mobile Media 2***	***Mobile Tourist Planner***	***Mobi Home***	***Service-Based Systems***
NF	33	43	10	14	20
NO	5	10	1	3	7
NM	6	12	3	4	5
NTop	6	15	4	2	6
NLeaf	22	32	7	7	14
DT Max	5	5	3	6	3
DT Mean	2.63	1.91	1.71	3.14	1.93
DT Median	3	1.5	2	4	2
CogC	8	7	2	3	3
FEX	30	34	7	7	21
FoC	0.15	0.23	0.1	0.21	0.35
SCDF	6	0	0	0	0
MCDF	2	2	0	0	7
CyC	4	3	0	1	7
ComC	1329.56	2024.56	110.33	247.44	482
NGF	11	11	3	7	6
CTCV	10	5	0	2	11
CTCR	0.30	0.12	0	0.14	0.55
RCon	0.24	0.07	0	0.07	0.3
RDen	1.5	0.67	0	1	0.85
CoC	0.97	0.98	0.9	0.93	0.95
NVF	26	30	6	9	14
SHoF	19	9	2	6	0
MHoF	2	11	3	0	7
RNoF	12	23	5	8	13
RoV	2.11	2.72	2.2	1.6	2.44
NVC	9198	2128896	28	21	213
BF Max	6	15	4	3	6
NGOr	1	4	1	0	3
NGXOr	7	3	1	3	0
ROr	0.06	0.26	0.3	0	0.35
RXOr	0.58	0.21	0.2	0.43	0
NC	-	-	2	2	2
NAF	-	-	-	-	-
NDF	-	-	-	-	-
NCC	-	-	-	-	-
CF	-	-	3	5	16
CFC	-	-	0	2	10
AFCA	-	-	5	8	15
DFCA	-	-	3	5	2

A subcaracterística Complexidade Cognitiva é relacionada à medida Complexidade Cognitiva (CogC). Complexidade Cognitiva denota o quão fácil o software pode ser entendido. Esta subcaracterística está relacionada a rastreabilidade da variabilidade da engenharia de uma LPS (ŠTUIKYS; DAMAŠEVICIUS, 2009). O modelo de *features* que apresentou a menor complexidade cognitiva foi o *Strategy Mobile Game* (CogC=8) e o *Mobile Media 2* (CogC=7).

A subcaracterística Extensibilidade pode ser atribuída à medida Extensibilidade da *feature* (FEX). Extensibilidade refere-se à capacidade de estender um modelo de *features* e o nível de esforço necessário para implementar a extensão. Quanto maior for o valor do FEX, maior a extensibilidade do modelo de *features* (MONTAGUD; ABRAHãO, 2009; JANAKIRAM; RAJASREE, 2005; ETXEBERRIA; SAGARDUI, 2008a; MONTAGUD *et al.*, 2012;

ETXEBERRIA *et al.*, 2008). Portanto, o modelo de *features* que apresenta melhor capacidade de extensão foi o *Mobile Media 2* (FEX =34).

A subcaracterística Flexibilidade é representada pela medida Flexibilidade da Configuração (FoC). Flexibilidade se refere à capacidade que um modelo de *features* tem para responder a possíveis mudanças internas ou externas que afetam o seu valor de entrega, de forma oportuna e rentável (DUAN *et al.*, 2013; BAGHERI E. E GASEVIC, 2011; ZHANG *et al.*, 2014; CHEN; BABAR, 2011; MONTAGUD *et al.*, 2012). Quanto maior for o valor do FoC, melhor será a flexibilidade. Portanto, o modelo de *features* que apresentou a melhor flexibilidade foi o *Service-Based Systems* (FoC = 0,35).

A subcaracterística Complexidade Estrutural pode ser atribuída por meio de um conjunto de medidas: Número de *Features* (NF), Número de *Features* Mandatórias (NM), Complexidade Ciclomática (CyC), Complexidade Composta (ComC), Restrições *Cross-tree* Variáveis (CTCV), Taxa de Restrições *Cross-tree* (CTCR), Coeficiente de Densidade de Conectivadade (CoC), Profundidade de Árvore (DT Max, DT Mean and DT Median), Número de *Features* Top (NTop), Número Máximo de *Features* por Filho (BF Max), Taxa de Conectividade do Grafo (RCon) e Densidade do Grafo (RDen). A Complexidade Estrutural está relacionada com a compreensão da estrutura do modelo de *features* (BAGHERI E. E GASEVIC, 2011; PATZKE *et al.*, 2012; MONTAGUD *et al.*, 2012). Quanto mais baixo for o valor das medidas de complexidade, menor é a complexidade do modelo de *features*. Assim, o modelo de *features* que apresentou o valor mais baixo da maioria das medidas de complexidade foi o *Mobile Tourist Planner* Logo, este modelo apresentou a menor complexidade estrutural. No entanto, isso decorreu devido ao fato deste modelo não possuir nenhuma restrição.

A subcaracterística de Variabilidade Estática pode ser atribuída por meio do conjunto de medidas: Número de Features Opcionais (NO), Número de Features Variáveis (NVF), *Single Hotspot Features* (SHoF), *Multiple Hotspot Features* (MHoF), *Rigid Nohotspot Features* (RNoF), Taxa de Variabilidade (RoV), Número de Configurações Válidas (NVC), Número de Grupos Or (NGOr), Número de Grupos XOr (NGXOr), Taxa de *Features* Or (ROr), e Taxa de *Features* XOr (RXOr). Variabilidade refere-se à capacidade de um artefato de ser configurado, personalizado, estendido, ou alterado, para utilização em um contexto específico. Quanto mais baixo for o valor das medidas de variabilidade estática, menor a variabilidade estática. Portanto, o modelo de *features* que apresentou a menor variabilidade foi o *Mobile Tourist Planner*.

Finalmente, a subcaracterística Variabilidade Dinâmica pode ser atribuída por meio

das medidas: Número de Contextos (NC), Número de *Features* Ativadas (NAF), Número de *Features* Desativadas (NDF), Número de Restrições de Contexto (NCC), Número de *Features* de Contextos (CF), *Features* de Contextos em Restrições (CFC), Número de *Features* Ativadas por Contexto (AFCA) e Número de *Features* Desativadas por Contexto (DFCA). A variabilidade dinâmica é caracterizada por presença de contextos e restrições de contextos no modelos de *features*. Quanto mais baixo for o valor das medidas de variabilidade dinâmica, menor a variabilidade dinâmica. A coleta das medidas NAF, NDF e NCC não foram levadas em consideração porque não foi selecionado um contexto específico para o cálculo dessas medidas. Além disso, só é possível coletar as medidas de variabilidade dinâmica em modelos de LPSDs. Portanto, o modelo de *features* que apresentou a menor variabilidade dinâmica foi o *Mobile Tourist Planner*.

Dessa forma, o modelo que apresentou melhores resultados, em relação a maioria das subcaracterísticas de manutenibilidade (Variabilidade Dinâmica, variabilidade estática, Complexidade Estrutural), foi o modelo *Mobile Tourist Planner*. No entanto, não é possível concluir, apenas utilizando as medidas, que o modelo de *features* possui melhor manutenibilidade, uma vez que a maior parte os outros quatro modelos também apresentam bons resultados para outras subcaracterísticas. Para isso, seria preciso investigar quais subcaracterísticas possuem maior impacto na qualidade do modelo de *features*.

É importante notar que não foram identificadas medidas para algumas características de qualidade, como por exemplo, usabilidade e segurança. Assim, é necessário propor novas medidas para estas características de qualidade. Outro problema complexo e relevante consiste em definir o objetivo da medida, isto é, os intervalos (*thresholds*) para análise da medição. Neste sentido, é necessário construir uma base histórica de medições de vários modelos de *features* e realizar uma análise estatística para definir o objetivo da medida.

O catálogo de medidas COfFEE é uma contribuição desta Tese, e suas medidas foram implementadas em uma ferramenta desenvolvida neste trabalho (ver Capítulo 6) para que o Engenheiro de Domínio possa realizar de forma automática a coleta das medidas em modelos de *features* de LPSs e LPSDs para suportar a avaliação do modelo com base nas medidas e propor melhorias no modelo. As medidas do catálogo serão utilizadas como base na validação destas medidas a partir dos estudos experimentais realizados nesta Tese.

5.6 Conclusões

Neste capítulo foi apresentado um mapeamento sistemático e uma revisão da literatura com a finalidade de construir um catálogo de medidas, denominado COfFEE, que pode ser utilizado para suportar a avaliação da manutenibilidade de modelos de *features*. O catálogo COfFEE é composto de 40 medidas voltadas para a avaliação da manutenibilidade de modelos de *features*. O COfFEE foi aplicado em 5 modelos de *features* no domínio de aplicações móveis. Por fim, os resultados da aplicação das medidas do catálogo COfFEE nos 5 modelos de *features* avaliados foram analisados e discutidos.

6 CONSTRUINDO *DATASETS* PARA SUPORTAR A AVALIAÇÃO DE QUALIDADE DE MODELOS DE *FEATURES*

Neste capítulo é apresentada uma ferramenta, denominada DyMMer, que possibilita a edição de modelos de *features* e a extração automática das 40 medidas de qualidade que compõem o catálogo COfFEE. Adicionalmente, são descritas as atividades realizadas com a finalidade de construir, a partir do catálogo COfFEE e da utilização da ferramenta DyMMer, três *datasets* de medidas de qualidade distintos, denominados: MAcchiATO, AFFOgaTO e ESPREssO.

Na Seção 6.1 são apresentados os principais *datasets* encontrados na literatura e uma metodologia para construção de *datasets*. Posteriormente, na Seção 6.2 é apresentada a ferramenta DyMMer. Na Seção 6.3 é apresentado o *dataset* MAcchiATO, o qual foi construindo com o objetivo de auxiliar a avaliação da qualidade de modelos de *features* de LPSs. O *dataset* AFFOgaTO, elaborado com a finalidade de possibilitar o estudo do impacto da evolução dos modelos de *features* na qualidade destes modelos, é discutido na Seção 6.4. Na Seção 6.5 é descrito o *dataset* ESPREssO, concebido para suportar a avaliação da qualidade de modelos de *features* de LPSDs. Por fim, na Seção 6.6 são apresentadas as conclusões deste capítulo.

6.1 Introdução

Recentemente, devido aos avanços proporcionados principalmente pela Internet, a quantidade de dados relacionados a projetos de software que são disponibilizados de forma aberta cresceu vertiginosamente. Diversos repositórios de dados sobre engenharia de software estão atualmente disponíveis gratuitamente, tais como o PROMISE e o Qualitas Corpus, dentre outros (MENZIES; ZIMMERMANN, 2013). Esses repositórios armazenam diferentes tipos de dados, tais como: código fonte, relatórios de erros (*it bugs*), revisões de código, relatórios sobre a execução de programas, *feedback* dos usuários, entre outros (KIM *et al.*, 2016). Um repositório pode ser formado por um ou mais *datasets*. Um *dataset* consiste em uma coleção de dados (ou observações) organizados e relacionados a um determinado assunto ou aspecto. *Datasets* são comumente utilizados em diversas pesquisas na área de engenharia de software experimental (GOUSIOS, 2013; GOUSIOS; ZAIDMAN, 2014; SPINELLIS, 2015; ORTU *et al.*, 2015).

Uma importante conferência denominada *International Conference on Mining Software Repositories - MSR*, tem incentivado pesquisas relacionadas à mineração de repositórios de dados sobre engenharia de software com a finalidade de descobrir padrões e suportar a tomada

de decisões (MENZIES; ZIMMERMANN, 2013). A utilização de *datasets* pode auxiliar a validação de pesquisas científicas e a realização de estudos experimentais. Em geral, um *dataset* inclui:

- Uma descrição dos dados, incluindo a sua fonte;
- A metodologia utilizada para coletar ou gerar os dados (de preferência com a ferramenta utilizada com esta finalidade);
- Uma descrição do esquema utilizado para armazenar os dados;
- Uma descrição de como os dados podem ser efetivamente utilizados por terceiros;
- Os problemas de pesquisa nos quais o *dataset* poderia ser utilizado; e
- Quaisquer limitações e/ou desafios envolvendo o *dataset*.

Apesar da crescente disponibilização de repositórios de dados sobre engenharia de software, apenas três *datasets* voltados para a área de LPS foram encontrados (ZHANG; BECKER, 2013a; PASSOS; CZARNECKI, 2014; DINTZNER *et al.*, 2013). Porém, nenhum destes *datasets* possui um conjunto de medidas para avaliação de modelos de *features* que seja estatisticamente relevante. Adicionalmente, nenhum *dataset* contendo medidas para avaliação de modelos de *features* em LPSD foi encontrado durante a realização desta Tese. A maior parte dos *datasets* encontrados na área de LPS se concentra na identificação de defeitos e análise de código.

Por estes motivos, neste trabalho foram desenvolvidos três *datasets* de medidas de qualidade para avaliação do modelo de *features*, denominados: MAcchiATO, AFFOgaTO e ESPREssO. Esses *datasets* foram construídos a partir das medidas do catálogo COfFEE e de um conjunto de modelos de *features* extraídos do repositório S.P.L.O.T (MENDONCA *et al.*, 2009a) e da literatura. Com a finalidade de apoiar a construção destes três *datasets*, foi desenvolvida uma ferramenta, denominada DyMMer. A ferramenta DyMMer possibilita a edição de modelos de *features* e a extração automática das 40 medidas de qualidade que compõem o catálogo COfFEE.

6.2 A Ferramenta DyMMer

Baseado no catálogo de medidas COfFEE, foi implementada uma ferramenta denominada DyMMer (*Dynamic feature Model tool based on Measures*). A ferramenta DyMMer suporta a edição de modelos de *features* e a extração automática das 40 medidas de qualidade que compõem o catálogo COfFEE. Diferentemente de outras ferramentas existentes (e.g., FAMI-

LIAR[1], FAMA[2], S.P.L.O.T.[3] e VariaMos[4]), a DyMMer coleta um número grande de medidas de qualidade com a finalidade de apoiar a avaliação da manutenibilidade do modelo de *features*, 40 medidas no total, utiliza medidas específicas para modelos de *features* de LPSDs e exporta, automaticamente, os valores das medidas para planilhas no formato *Microsoft Excel*.

A ferramenta DyMMer foi desenvolvida para extrair medidas de modelos de *features* representados segundo o formato XML proposto pelo repositório de modelo de *features* S.P.L.O.T. Desta forma, inicialmente, a DyMMer recebe como entrada um conjunto de modelos de *features*, onde cada modelo de *features* é representado por um arquivo XML. Em seguida, cada arquivo XML é processado individualmente e uma estrutura interna em memória principal (objeto Java) é criada para representá-lo. Assim, para cada modelo de *features* tem-se um arquivo XML e uma representação interna. Utilizando essas estruturas internas, a DyMMer calcula, automaticamente, para cada modelo de *features*, os valores das 40 medidas de qualidade presentes no catálogo COfFEE. A ferramenta DyMMer e sua documentação estão disponíveis *on-line* [5].

6.2.1 Arquitetura da Ferramenta DyMMer

A ferramenta DyMMer foi desenvolvida utilizando a plataforma Java e o S.P.L.A.R, que é um componente do S.P.L.O.T (MENDONCA *et al.*, 2009a) utilizado como uma biblioteca de baixo nível. A DyMMer extrai medidas apenas de modelos de *features* consistentes e corretos. Os modelos de *features* consistentes e corretos são automaticamente verificados pelo S.P.L.A.R utilizando uma resolução de satisfatibilidade (*SAT solving*) e diagramas de decisão binária (BDD) (MENDONCA *et al.*, 2008).

A arquitetura da ferramenta DyMMer é composta por três camadas principais, conforme ilustrado na Figura 15: (i) visualizador de modelo de *features*; (ii) editor de modelo de *features*; e (iii) exportação de dados. A camada de visualização torna possível ao engenheiro de domínio visualizar e analisar um determinado modelo de *features*. A camada de edição permite editar um modelo de *features* específico. Este editor torna possível, por exemplo, adicionar adaptações de contexto em modelos de *features* que não possuam essa informação. Vale destacar que a DyMMer considera que tanto as *features* de contexto quanto as *features* que não são de contexto são representadas em um único modelo de *features*. A camada de exportação de dados

[1] http://familiar-project.github.io/
[2] http://www.isa.us.es/fama/
[3] http://www.splot-research.org/
[4] http://www.variamos.com/
[5] https://github.com/DyMMerProject/DyMMerV2

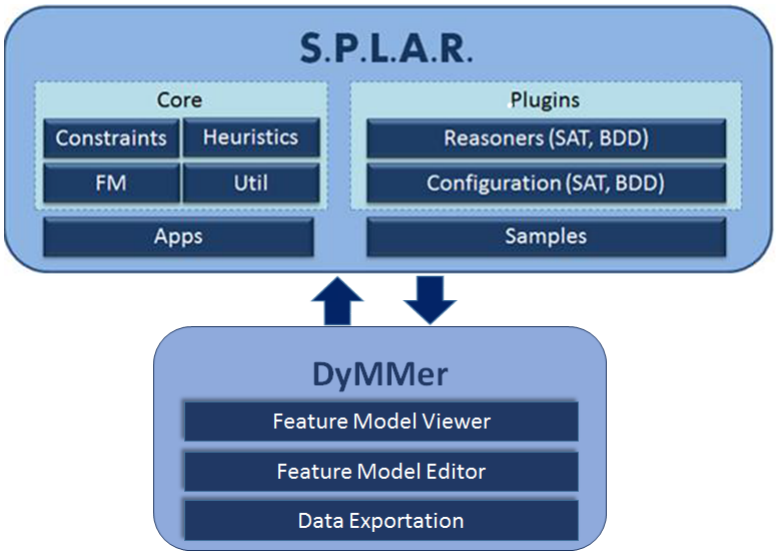


Figura 15 – Visão geral da arquitetura DyMMer e S.P.L.A.R.

exporta, automaticamente, os valores das 40 medidas de qualidade extraídas de cada um dos modelos de *features* recebidos como entrada para uma planilha no formato *Microsoft Office Excel*.

6.2.2 Principais Funcionalidades da DyMMer

Nesta Seção, são apresentadas as principais funcionalidades da ferramenta DyMMer, que são: importação, visualização, edição e exportação do modelo de *features*.

6.2.2.1 Importando um Modelo de Features

A DyMMer, diferentemente de outras ferramentas, não foi concebida com o objetivo de ser um editor de modelos de *features*, mas uma ferramenta para extração automática dos valores de medidas de qualidade. Desta forma, a DyMMer importa modelos de *features* já existentes, descritos no formato XML definido pelo repositório S.P.L.O.T., processa esses modelos e cria, para cada um dles, uma representação interna em memória principal. A Figura 16 ilustra a tela importação de modelos de *features* da ferramenta DyMMer.

6.2.2.2 Visualizando um Modelo de Features

A DyMMer torna possível visualizar e analisar um modelo de *features* específico, previamente importado, conforme ilustrado na Figura 17. Para os modelos de *features* que já possuem adaptações de contextos, é possível visualizar, para cada contexto, as *features* ativadas e

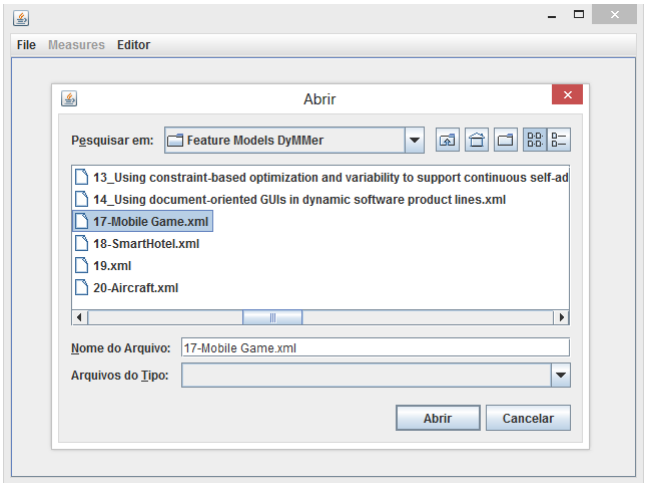


Figura 16 – Importando um modelo de *features*.

desativadas, de acordo com suas restrições. Além de visualizar a estrutura do modelo de *features*, a ferramenta DyMMer permite, por meio da aba de medidas, selecionar um subconjunto das medidas de qualidades suportadas, computar e visualizar os valores dessas medidas.

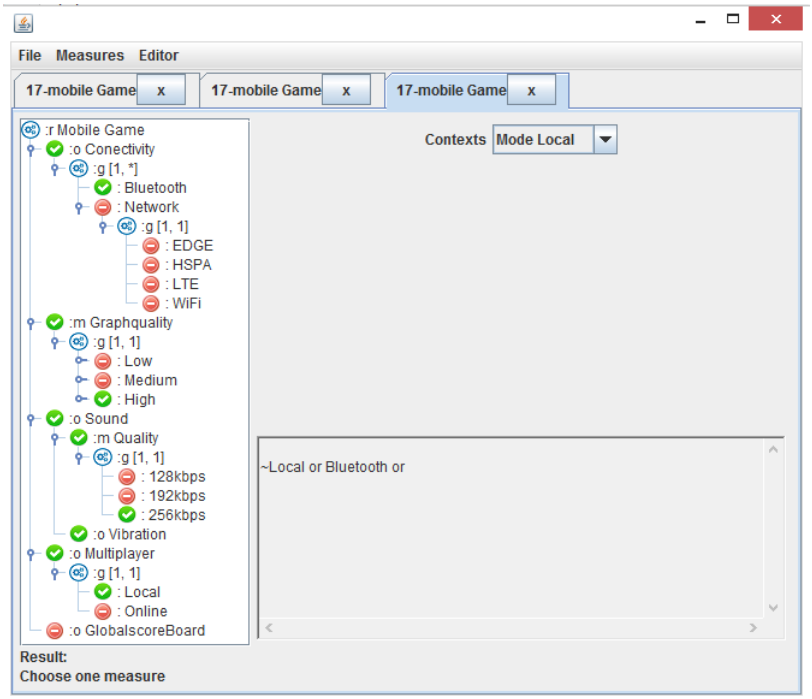


Figura 17 – Visualizando um modelo de *features*.

6.2.2.3 Editando um Modelo de Features

Além possibilitar a importação e visualização de modelos de *features*, a ferramenta DyMMer permite também a edição desses modelos, tornado possível, por exemplo, adicionar informações de contexto (em modelos de *features* sem contexto), por meio da adição ou remoção

de *features* ou restrições de contexto, ou ainda pela inclusão de informações sobre a ativação e desativação de *features* de contexto. Vale destacar que em LPSDs um modelo de *features* pode conter um ou mais contextos. Esta funcionalidade possibilita que o engenheiro de domínio possa lidar com modelos de *features* de LPSDs, já que o S.P.L.O.T. não suporta a modelagem deste tipo de modelo. A Figura 18 ilustra a edição de um modelo de *features* na ferramenta DyMMer.

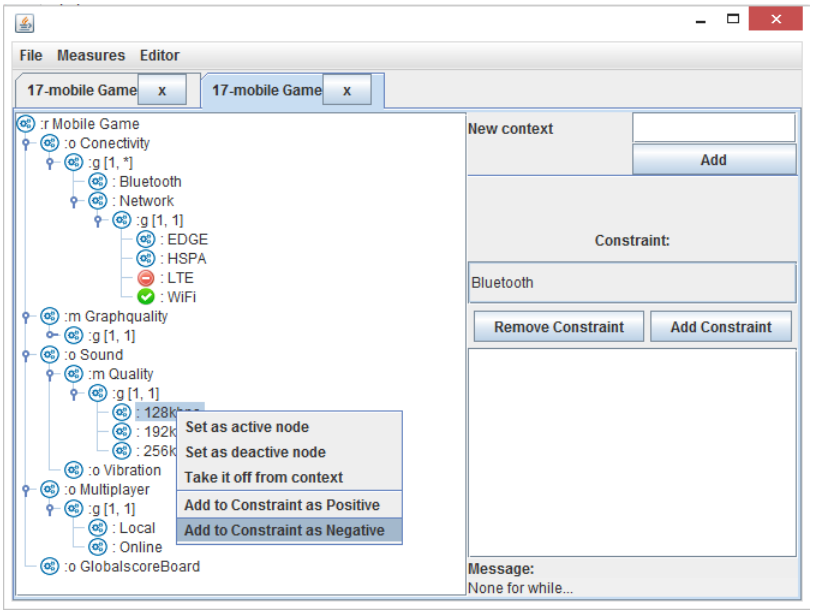


Figura 18 – Editando um modelo de *features*.

6.2.2.4 *Exportando Medidas*

Na DyMMer é possível exportar, de forma automática, os valores das medidas de qualidade para uma planilha no formato *Microsoft Office Excel*, conforme ilustrado na Figura 19. O engenheiro de domínio pode exportar todas as medidas de uma só vez para um ou mais modelos de *features*. Essa é uma grande vantagem da ferramenta DyMMer, pois possibilita a análise de vários modelos de *features* em conjunto. A DyMMer permite também que o engenheiro de domínio exporte apenas um subconjunto das 40 medidas de qualidade suportadas.

6.3 MAcchiATO: Um *Dataset* de Medidas para Modelos de *Features* de LPSs

Com a finalidade de suportar a avaliação da qualidade de modelos de *features* de LPSs, foi construído um *dataset* denominado MAcchiATO (*MeAsures dATaset for feaTure mOdel*). O *dataset* MAcchiATO contém os valores de 32 medidas pertencentes ao catálogo

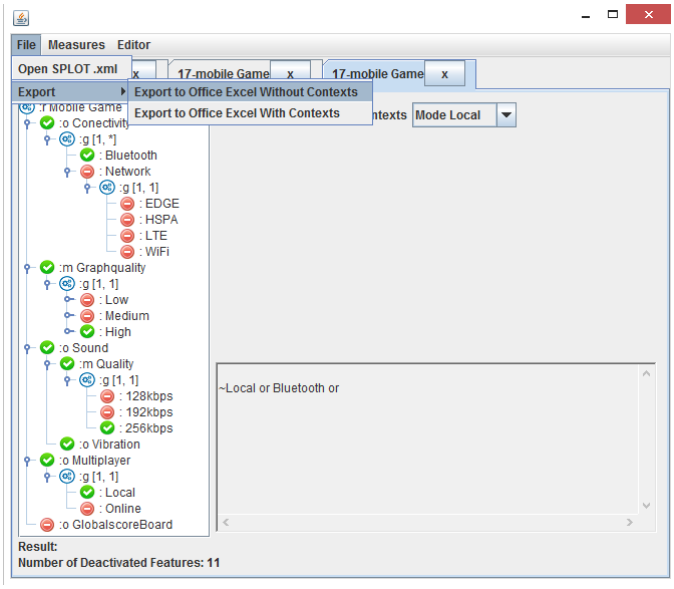


Figura 19 – Exportando os resultados das medidas extraídos a partir de um ou mais modelos de *features* para um arquivo XLS.

COfFEE [6] computadas para 218 modelos de *features* de LPSs extraídos do repositório S.P.L.O.T. Essas 32 medidas foram selecionadas por não envolverem informações de contexto, uma vez que o objetivo deste *dataset* é auxiliar a avaliação de modelos de *features* de LPSs, e não de LPSDs. O *dataset* MAcchiATO[7] está disponível de forma *online* e pode ser utilizado gratuitamente pela comunidade de engenharia de software.

A geração do *dataset* MAcchiATO seguiu as seguintes etapas:

- **Extraindo e Selecionando os Modelos de *Features*:** um subconjunto de todos os modelos de *features* armazenados no repositório S.P.L.O.T. foi selecionado manualmente. Esta seleção incluiu: (i) apenas modelos de *features* de software (por exemplo, modelos de *features* relacionados aos domínios de automóveis, motos, notebooks e bicicletas, por exemplo, foram descartados); e (ii) modelos de *features* não-duplicados. Esses filtros foram necessários, porque o repositório S.P.L.O.T. armazena modelos de *features* de diferentes domínios e, muitas vezes, duplicados. Após este processo, foram selecionados 218 modelos de *features*[8].
- **Importando os Modelos de *Features*:** os 218 modelos de *features* selecionados foram extraídos e armazenados em um *workspace*. Em seguida, os modelos foram importados automaticamente utilizando-se a ferramenta DyMMer. Vale destacar que a ferramenta

[6] As medidas do *dataset* MAcchiATO correspondem às primeiras 32 medidas da Tabela 12.
[7] http://carlabezerra.great.ufc.br/macchiato
[8] O processo de extração foi realizado até o dia 25/02/2015. Nesta data, o S.P.L.O.T. armazenava 605 modelos de *features*.

DyMMer computou automaticamente os valores das 32 medidas que compõem o MAcchiATO para os 218 modelos de *features* previamente selecionados, enquanto utilizando-se apenas o repositório S.P.L.O.T. seria possível calcular automaticamente apenas 11 das 32 medidas presentes no MAcchiATO, mais especificamente: Número de *features*, Número de *features* opcionais, Número de *features* obrigatórias, Número de *features* agrupadas, número de *features* Folha, Profundidade de Árvore, Restrições *Cross-Tree*, Número de *features* variáveis, Taxa de Variabilidade e Número de configurações válidas. Adicionalmente, utilizando o S.P.L.O.T., o engenheiro de domínio precisaria selecionar cada um dos 218 modelos de *features* individualmente e solicitar a computação das medidas, ou seja, ele deveria realizar esse processo uma vez para cada modelo de *features* a ser avaliado. Por outro lado, usando a ferramenta DyMMer, o engenheiro de domínio executa esse processo uma única vez, independentemente da quantidade de modelos de *features* a serem avaliados. Vale mencionar ainda que os valores computados pelo S.P.L.O.T. não são armazenados. Assim, para se obter o valor de uma determinada medida, que já foi computada anteriormente, para um modelo de *features* específico, é necessário executar novamente o processo de carga do modelo e computação da medida. Já a ferramenta DyMMer evita esse *overhead* armazenando os valores das medidas, de um ou mais modelos de *features* simultaneamente, em um arquivo no formato *Microsoft Excel*.

- **Exportando os Valores das Medidas:** Após a importação dos 218 modelos de *features* selecionados e do cálculo dos valores das 32 medidas que compõem o *dataset* para cada um destes modelos, esses valores foram exportados para uma planilha no formato *Microsoft Excel*, utilizando-se para isso a ferramenta DyMMer.

  Desta forma, o *dataset* MAcchiATO é composto de duas partes: (i) um conjunto de 218 modelos de *features* selecionados a partir do repositório S.P.L.O.T. (em formato XML); e (ii) uma planilha compilada que contém os valores de 32 medidas de qualidade para cada um dos 218 modelos de *features*.

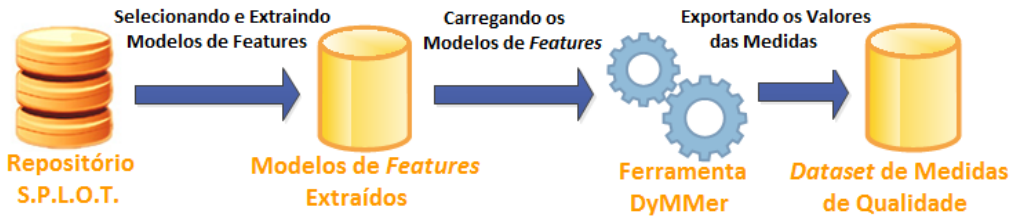


Figura 20 – Processo de geração do *dataset* MAcchiATO.

O *dataset* MAcchiATO foi utilizado em um estudo de caso (BEZERRA *et al.*, 2016) com a finalidade de identificar correlações entre essas 32 medidas de qualidade, que podem ser utilizadas para avaliar modelos de *features* de LPSs; tentar reduzir a quantidade de medidas necessárias para este tipo de avaliação, uma vez que utilizar 32 medidas poderia gerar uma sobrecarga de trabalho para o engenheiro de domínio; e definir limites de referências (*thresholds*) para essas medidas.

6.4 AFFOgaTO: Um *Dataset* de Medidas para Evoluções de Modelos de *Features* de LPSs

O modelo de *features* e a própria LPS evoluem e crescem em conjunto (LOTUFO *et al.*, 2010). Segundo Pussinen et al. (PUSSINEN, 2002), enquanto a evolução na Engenharia de Software tradicional ocorre na fase de manutenção, LPSs evoluem ao longo de todo o seu ciclo de vida. A evolução do modelo de *features* ocorre quando algumas mudanças são realizadas na estrutura do modelo (incluindo suas restrições), tais como: adição ou remoção de *features*; adição ou remoção de relacionamentos entre *features*; adição ou remoção de restrições; e modificação da variabilidade do modelo (GAMEZ; FUENTES, 2011). Contudo, a experiência envolvendo LPSs mostra que especificações e realizações de variabilidade nos modelos de *features* tendem a se desgastar no sentido em que estas tornam-se excessivamente complexas e inconsistentes entre si (ZHANG; BECKER, 2013b). Desta forma, acredita-se que a avaliação da qualidade dos modelos de *features* é necessária ao longo de todo o processo de evolução da LPS.

Neste cenário, foi desenvolvido um *dataset* denominado AFFOgaTO (*dAtaset For the Feature mOdel evoluTiOn*). O AFFOgaTO tem por finalidade possibilitar o estudo do impacto do processo de evolução dos modelos de *features* na qualidade destes modelos. Por este motivo, o *dataset* AFFOgaTO é composto por: (i) um conjunto de 16 modelos de *features*, com suas respectivas versões, obtidos a partir do repositório S.P.L.O.T.; e (ii) uma planilha compilada contendo os valores de 21 medidas estruturais, para cada versão de cada um dos 16 modelos selecionados. O *dataset* AFFOgaTO [9] está disponível para *download* e pode ser usado livremente pela comunidade de engenharia de software.

Para a construção do AFFOgaTO foram selecionadas, a partir do catálogo COfFEE, 21 medidas estruturais. Essas 21 medidas cobrem as principais subcaracterísticas relacionadas à manutenibilidade dos modelos de *features* (BEZERRA *et al.*, 2015), mais precisamente:

[9] https://goo.gl/gye5ma

analisabilidade (NLeaf), complexidade cognitiva (CogC), extensibilidade (FEX), flexibilidade (FoC), modularidade (SCDF e MCDF), complexidade estrutural (CyC, ComC, CTC, CoC, DT, NF, NTop) e variabilidade (MHoF, RNoF, SHoF, RoV, NVF, NVC, NO, NA). As medidas de manutenibilidade são importantes para avaliar quão bem modelos de *features* podem ser entendidos, alterados e analisados.

Para geração do *dataset* AFFOgaTO, foram executadas as seguintes etapas:

- **Identificando Evoluções de Modelos de *Features***: nesta etapa, um subconjunto de todos os modelos de *features* armazenados no repositório S.P.L.O.T. foi selecionado manualmente. Esta seleção incluiu: (i) apenas modelos de *features* de software; (ii) modelos de *features* não-duplicados; e (iii) modelos de *features* que possuíssem pelo menos duas versões distintas. Após este processo, foram selecionados 16 modelos de *features*[10].
  Vale destacar que o repositório S.P.L.O.T. não possui um controle de versões. Logo, foi necessário identificar manualmente as versões de um mesmo modelo de *features*. Para isso, as seguintes atividades foram realizadas:
  - **Agrupamento Inicial de Versões**: inicialmente, foram identificados os modelos de *features* com o mesmo nome, com o mesmo autor ou ainda com estrutura similar. Em geral, os modelos de *features* que representam versões de um mesmo modelo de *features* possuem nomes similares e são registrados pelo mesmo autor. Assim, os modelos de *features* com o mesmo nome, com o mesmo autor e com estrutura similar foram agrupados formando um conjunto de versões de um mesmo modelo de *features*, ou seja, uma evolução de um modelo de *features* .
  - **Verificação Estrutural**: para cada conjunto de versões do mesmo modelo de features (denominado evolução), previamente identificado, foi realizada uma verificação estrutural. Assim, para cada modelo de *features*, do mesmo agrupamento, foi verificada sua estrutura a fim de garantir que eles correspondem a diferentes versões do mesmo modelo de *features*. Os modelos de *features* cujas estruturas foram dissimilares, foram descartados.
  - **Ordenação dos Agrupamentos**: em seguida, os modelos de *features* pertencentes a um mesmo agrupamento (evolução) foram ordenados. Para isso, foi utilizada a data de gravação no repositório S.P.L.O.T. Ao final do processo, foram selecionadas 16 evoluções, onde cada evolução contém diferentes versões de um mesmo modelo de

[10] O processo de extração foi realizado até o dia 25/02/2015. Nesta data, o S.P.L.O.T. armazenava 605 modelos de *features*.

features.

- **Exportando os Valores das Medidas**: nesta etapa, para cada versão de cada agrupamento (evolução) identificado anteriormente foram computados os valores das 21 medidas selecionadas para compor o *dataset*. Essa atividade foi realizada utilizando-se a ferramenta DyMMer.

O *dataset* AFFOgaTO foi utilizado para suportar um estudo inicial sobre o impacto do processo de evolução dos modelos de *features* na qualidade destes modelos (BEZERRA *et al.*, 2016), conforme descrito no Capítulo 7 desta Tese. Este estudo discute, por exemplo, as subcaracterísticas de qualidade que tendem a permanecer estáveis durante a evolução de um modelo de *features* e aquelas que sofrem maiores variações.

6.5 ESPREssO: Um *Dataset* de Medidas para Modelos de *Features* de LPSDs

Uma das principais dificuldades enfrentadas atualmente na avaliação de modelos de *features* de LPSDs consiste na ausência de repositórios contendo esses artefatos. O repositório S.P.L.O.T. armazena apenas modelos de *features* de LPSs tradicionais, ou seja, que não incluem, por exemplo, informações de contexto. Durante a realização desta Tese, não foi identificado na literatura nenhum repositório ou *dataset* contendo modelos de *features* de LPSDs. Para suprir esta necessidade, foi construído um *dataset* denominado ESPREssO (*mEasures dataSet for dsPl featuRE mOdel*), o qual foi concebido para suportar a avaliação da qualidade de modelos de *features* de LPSDs. O *dataset* ESPREssO[11] está disponível *online* e pode ser utilizado gratuitamente pela comunidade de engenharia de software.

O *dataset* ESPREssO é composto por: (i) um conjunto de 30 modelos de *features* de LPSDs, extraídos da literatura; e (ii) uma planilha compilada contendo os valores de 13 medidas de qualidade, para cada um dos 30 modelos selecionados. As 13 medidas que compõem o *dataset* ESPREssO foram selecionadas a partir do catálogo COfFEE, sendo 9 medidas voltadas para LPSs e 4 específicas para LPSD. As 9 medidas relacionadas às LPSs abrangem 5 subcaracterísticas da manutenibilidade, mais especificamente: Analisabilidade (NLeaf), Complexidade Cognitiva (CogC), Flexibilidade (FoC), Complexidade Estrutural (DT, NF, RDen) e Variabilidade (RoV, NVC, NO). As 4 medidas específicas para LPSDs são: *Features* de Contexto (CF), *Features* de Contexto em Restrições (CFC), *Features* Ativadas por Contexto (AFCA) e *Features* Desativadas por Contexto (DFCA).

[11] https://goo.gl/ONfTL3

A geração do *dataset* ESPREssO seguiu as seguintes etapas:

- **Seleção de Modelos de *Features* de LPSDs:** inicialmente, foram identificados e selecionados, a partir da literatura, modelos de *features* de LPSDs. Esta seleção incluiu: (i) apenas modelos de *features* de LPSDs; e (ii) modelos de *features* não duplicados. Após este processo, foram selecionados 30 modelos de *features* de LPSDs. Em seguida, os modelos de *features* selecionados foram adicionados ao repositório S.P.L.O.T., sem as adaptações de contexto.
- **Adição das Adaptações de Contexto:** os modelos de *features* selecionados são recuperados do repositório S.P.L.O.T., em formato XML e ainda sem as adaptações de contexto, e, em seguida, importados para a ferramenta DyMMer. Depois disso, os modelos de *features* são editados, na própria ferramenta DyMMer, a fim de se inserir as adaptações de contexto, em cada modelo.
- **Exportação dos Valores das Medidas de LPSDs:** Os modelos de *features* são carregados e processados pela ferramenta DyMMer, a fim de se calcular os valores das medidas de qualidade. Em seguida, os valores das 13 medidas que compõem o *dataset* ESPREssO são compilados e armazenados em uma planilha no formato do *Microsoft Office Excel*.

O *dataset* ESPREssO foi utilizado para suportar a agregação de medidas de qualidade por meio da lógica *fuzzy*, conforme será discutido no Capítulo 9. A partir dessa agregação foram derivadas 4 novas medidas para avaliação de diferentes características dos modelos de *features* de LPSDs, mais especificamente: tamanho, estabilidade, flexibilidade e dinamicidade.

6.6 Conclusões

Neste capítulo foi apresentada a ferramenta DyMMer, desenvolvida com o objetivo de apoiar a avaliação da qualidade do modelo de *features*. A ferramenta realiza a coleta automática das medidas pertencentes ao catálogo COfFEE e também suporta a modelagem de adaptações de contextos em modelos de *features* de LPSDs. Com o suporte da ferramenta DyMMer, foram gerados três *datasets* de medidas distintos, com a finalidade de apoiar os estudos desenvolvidos nesta Tese, denominados de: MAcchiATO, AFFOgaTO e ESPREssO. O *dataset* MAcchiATO contém os valores de 32 medidas de qualidade extraídas de 218 modelos de *features* de LPSs tradicionais. O *dataset* AFFOgaTO é composto pelos valores de 21 medidas de qualidade extraídas de 16 evoluções de modelos de *features*. Já o *dataset* ESPREssO é formado pelos valores de 13 medidas de qualidade extraídas de 30 modelos de *features* de LPSDs.

7 ESTUDO EXPLORATÓRIO: EVOLUÇÃO DOS MODELOS DE *FEATURES*

Este capítulo apresenta um estudo exploratório que investiga o impacto da evolução dos modelos de *features* em relação à manutenibilidade destes modelos. Na Seção 7.1 é apresentada uma discussão acerca do processo de evolução dos modelos de *features*. A Seção 7.2 apresenta o planejamento deste estudo exploratório. A Seção 7.3 detalha a execução do estudo exploratório. A Seção 7.4 discute e resume os resultados obtidos neste estudo exploratório. A Seção 7.5 descreve as ameaças à validade do estudo. Finalmente, as conclusões deste capítulo são apresentadas na Seção 7.6.

7.1 Evolução dos modelos de *features*

Segundo Pussinen (2002), enquanto a evolução na Engenharia de Software clássica ocorre na fase de manutenção, LPSs evoluem ao longo de todo o seu ciclo de vida. No caso da evolução do modelo de *features*, ela ocorre quando algumas mudanças são realizadas na estrutura do modelo (incluindo suas restrições), tais como: adição ou remoção de *features*; adição ou remoção de relacionamentos entre *features*; adição ou remoção de restrições; e modificação da variabilidade do modelo (GAMEZ; FUENTES, 2011).

Entretanto, a experiência envolvendo LPSs mostra que o relacionamento entre as especificações representadas no modelo de *features* e os produtos derivados da linha tende a se desgastar ao longo do tempo, no sentido de que este modelo torna-se excessivamente complexo e surgem inconsistências entre o modelo de *features* e os produtos derivados da linha. O modelo de *features* e a própria LPS evoluem e crescem juntos (LOTUFO *et al.*, 2010). Desta forma, também existe um relacionamento entre o modelo de *features* e os demais artefatos intermediários da linha. Este relacionamento também se torna mais complexo à medida que linha evolui.

Assim, é de fundamental importância que a avaliação da qualidade de modelos de *features* seja realizada ao longo de todo o processo de evolução da LPS, a fim de garantir a qualidade do modelo de *features* ao longo do tempo em suas diferentes versões.

Contudo, avaliar a qualidade do modelo de *features* ao longo de toda a sua evolução pode configurar-se em uma atividade bastante complexa e custosa em termos financeiros. Por este motivo, decidiu-se explorar neste estudo apenas uma característica de qualidade: a manutenibilidade. Esta escolha deve-se à grande relevância desta característica de qualidade. Segundo a norma SQuaRE (ISO/IEC, 2011), descrita na Seção 3.2.1, a característica de qualidade de

manutenibilidade é composta pelas seguintes subcaracterísticas: modularidade, reusabilidade, analisabilidade, modificabilidade e testabilidade. Contudo, a literatura apresenta outras subcaracterísticas relacionadas à manutenibilidade, tais como: complexidade cognitiva, extensibilidade, flexibilidade, complexidade estrutural e variabilidade.

Para a execução deste estudo exploratório, foram utilizadas as 21 medidas e 16 modelos de *features* com suas respectivas versões do *dataset* AFFOgaTO, descrito na Seção 6.4. Essas 21 medidas cobrem as principais subcaracterísticas relacionadas à manutenibilidade dos modelos de *features* (BEZERRA *et al.*, 2015), mais precisamente: analisabilidade (NLeaf), complexidade cognitiva (CogC), extensibilidade (FEX), flexibilidade (FoC), modularidade (SCDF e MCDF), complexidade estrutural (CyC, ComC, CTC, CoC, DT, NF, NTop) e variabilidade (MHoF, RNoF, SHoF, RoV, NVF, NVC, NO, NA). As medidas de manutenibilidade são importantes para avaliar quão bem modelos de *features* podem ser entendidos, alterados e analisados. Neste capítulo, um conjunto de versões de um mesmo modelo de *features*, ordenadas pela ordem cronológica de sua criação, é denominado "evolução".

Este estudo, embora ainda seja uma investigação inicial, pode ser útil para melhorar o entendimento de como os modelos de *features* evoluem ao longo do tempo. Este estudo, discute, por exemplo, as subcaracterísticas de qualidade que tendem a permanecer estáveis durante a evolução de um modelo de *features* e aquelas que sofrem maiores variações.

7.2 Planejamento do Estudo Exploratório

Os estudos exploratórios são muito utilizados para realizar uma investigação preliminar acerca do tema principal da pesquisa que será realizada, permitindo ao pesquisador familiarizar-se com o fenômeno que está sendo estudado, de modo que a pesquisa subsequente possa ser concebida com uma maior compreensão e precisão. Além disso, permite ao pesquisador definir o seu problema de pesquisa, formular a sua hipótese com mais precisão, selecionar as técnicas mais adequadas para suas pesquisas e decidir sobre as questões que mais necessitam de atenção e investigação detalhada (PETTERSEN *et al.*, 2004).

Um estudo exploratório envolve o levantamento bibliográfico e/ou a análise de um conjunto de exemplos que estimulem a compreensão do tema da pesquisa. Assim, as pesquisas exploratórias visam proporcionar uma visão geral de um determinado problema. Em geral, um estudo exploratório é realizado sobre um problema ou questão de pesquisa que tem poucas ou nenhuma pesquisa anterior a seu respeito. O objetivo desse tipo de estudo é procurar

padrões, ideias ou hipóteses. A ideia não é testar ou confirmar uma determinada hipótese. Uma das principais técnicas utilizadas nas pesquisas exploratórias é o estudo de caso. Um estudo exploratório avalia quais teorias ou conceitos existentes podem ser aplicados a um determinado problema ou se novas teorias e conceitos devem ser desenvolvidos (ROBSON, 2002; RUNESON P. E HÖST, 2009).

O estudo exploratório descrito neste capítulo teve por objetivo investigar se a manutenibilidade do modelo de *features* tende a aumentar ou diminuir ao longo do seu processo de evolução. Para esta investigação, alguns fatores foram considerados importantes para a manutenibilidade do modelo de *features*, tais como: analisabilidade, modificabilidade, complexidade estrutural, variabilidade e flexibilidade dos modelos de *features*.

Para guiar este estudo exploratório foram definidas 6 questões de pesquisa (QPs), as quais são ilustradas na Tabela 14. As questões de pesquisa são relacionadas a evolução estrutural dos modelos de *features* baseado em medidas de tamanho, complexidade e variabilidade. Essas mudanças estruturais do modelo podem impactar diretamente a manutenibilidade do modelo de *features*.

Tabela 14 – Questões de pesquisa do estudo exploratório.

ID	Descrição das Questões de Pesquisa
QP1	O número de *features* tende a aumentar com a evolução do modelo de *features*?
QP2	O número de *features* filhas envolvidas nos relacionamentos alternativo (XOR) ou OR tende a aumentar com a evolução do modelo de *features*?
QP3	A profundidade do modelo de *features* tende a aumentar com a evolução do modelo?
QP4	A largura do modelo de *features* tende a aumentar com a evolução do modelo?
QP5	A variabilidade tende a aumentar com a evolução do modelo de *features*?
QP6	A complexidade cresce com a evolução do modelo de *features*?

Para responder essas questões de pesquisa, foi utilizado o catálogo de medidas COfFEE e também o *dataset* AFFOgaTO (ver Seção 6.4).

7.3 Execução do Estudo Exploratório

Inicialmente, para cada evolução do modelo de *features*, foram identificadas as mudanças principais realizadas de uma versão para outra de cada modelo de *features*. Em seguida, as mudanças foram agrupadas e classificadas. Estes grupos de mudanças são chamados de "tipos de evolução" e foram identificadas 16 tipos de mudanças. A Figura 21 apresenta estes tipos de evolução e o número de vezes que cada tipo de evolução ocorreu no conjunto das 16

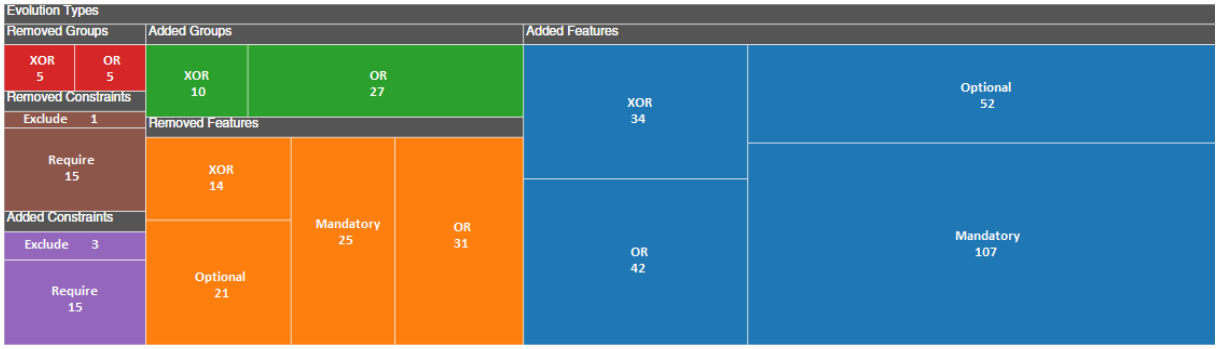


Figura 21 – Tipos de evolução e mudanças na análise de diferentes versões dos 16 modelos de *features*.

diferentes versões dos modelos de *features* analisados. Assim, pode-se observar na Figura 21, que o tipo de evolução mais frequente foi a inclusão de *features* obrigatórias, que ocorreu 107 vezes. Por outro lado, o tipo de evolução menos frequente foi a remoção da restrição *cross-tree* "*excludes*" (relacionamento), que ocorreu apenas uma vez. Pode-se observar também, nesta Figura, que 235 *features* foram adicionadas, enquanto 91 *features* foram removidas.

Após a identificação dos tipos de evolução foram analisados os valores das 21 medidas de manutenibilidade para cada evolução de modelo de *features*, definidas na Seção 6.4. Foram observados o padrão de comportamento para cada medida de manutenibilidade nas diferentes versões da mesma evolução do modelo de *features*. A Tabela 15 resume estes valores.

Para execução deste estudo exploratório, foram analisados os valores das medidas de manutenibilidade mostradas na Tabela 15 para diferentes versões do mesmo modelo de *features*. É importante entender como o processo de evolução impacta na manutenibilidade do modelo de *features*. Para análise dos dados foram definidas e analisadas as 6 questões de pesquisa da Tabela 14.

7.3.1 QP1. O número de features tende a aumentar com a evolução do modelo de features?

Como destaque prévio, considerando os 16 modelos de *features* e suas respectivas versões (Tabela 15), 235 *features* foram adicionadas, enquanto 91 *features* foram removidas (ver Figura 21). Além disso, analisando a medida "Número de Features" (NF) na Tabela 15, pode-se observar que apenas em duas evoluções de modelo de *features* de LPSs diferentes, esta medida permaneceu constante (*Eshop* e *Web Collaboration*). Em todos as outras 14 LPSs com suas respectivas evoluções, a medida NF aumentou. Em algumas evoluções, como "Toko", por exemplo, o valor da medida NF varia de 21 para 72, um crescimento significativo. Então, pode-se considerar que há uma forte evidência nestes dados que o número de *features* tende a aumentar

Tabela 15 – Resultados das medidas para diferentes versões dos modelos de *features*.

FM	V	Medidas																				
		NF	NA	NO	NLeaf	CogC	FoC	SCDF	MCDF	FEX	CyC	ComC	CTC	CoC	DT	NTop	MHoF	RNoF	SHoF	NVC	NVF	RoV
Eshop	V1	10	5	1	7	2	0,1	1	1	9	1	103,22	1	0,5	3	4	3	5	2	20	6	0,019
	V2	10	4	2	6	2	0,2	1	1	8	2	104,22	2	0,67	3	4	2	6	2	9	6	0,0088
Mobile Game	V1	10	4	4	6	1	0,4	0	0	6	0	100,7	0	0	3	3	4	6	0	135	8	0,13
	V2	16	4	10	9	1	0,62	0	4	13	1	257	1	0,06	3	6	4	12	0	3645	14	0,055
VODPlayer	V1	10	2	5	6	1	0,5	0	0	6	0	101,1	0	0	3	4	0	8	2	48	7	0,047
	V2	11	0	5	8	0	0,45	0	0	8	3	126,8	3	0,5	2	7	0	11	0	32	5	0,015
Mobile Media	V1	11	0	1	7	0	0,09	0	0	7	0	130	0	0	3	2	0	11	0	2	1	0,0009
	V2	14	0	2	9	0	0,14	0	0	9	0	209,4	0	0	3	2	0	14	0	4	2	0,00024
	V3	15	0	3	10	0	0,2	0	0	10	0	238,4	0	0	3	2	0	15	0	8	3	0,00024
	V4	18	0	4	12	0	0,22	0	0	12	0	342,8	0	0	3	2	0	18	0	16	4	6,10352E-05
	V5	23	2	5	14	1	0,22	0	0	14	1	509,8	1	0,5	4	3	2	21	0	60	7	7,15E-06
	V6	26	3	6	16	1	0,23	0	0	16	1	705,3	1	0,5	4	3	3	23	0	204	9	3,04E-06
Ecommerce	V1	12	0	2	8	0	0,17	0	0	8	0	153	0	0	3	2	0	12	0	3	2	0,000732422
	V2	53	5	3	35	4	0,057	0	0	35	0	3032,7	0	0	8	7	2	48	3	18	8	1,9984E-15
Software Product	V1	15	0	8	9	0	0,53	0	0	9	0	229	0	0	2	6	0	15	0	256	8	0,0078125
	V2	19	0	9	12	0	0,47	0	0	12	0	370	0	0	2	9	0	19	0	512	9	0,000976563
	V3	22	0	11	14	0	0,5	0	0	14	0	495,1	0	0	2	10	0	22	0	2048	11	0,000488281
Toko	V1	21	4	6	14	1	0,29	0	0	14	0	452,8	0	0	5	3	0	17	4	256	10	0,00012207
	V2	48	19	13	32	7	0,27	0	0	33	2	2353,2	2	0,5	5	7	10	29	9	2970240	32	1,05524E-08
	V3	49	24	7	32	9	0,14	0	0	32	1	2475	1	0,5	5	7	21	25	3	9031680	31	1,60435E-08
	V4	56	26	11	37	10	0,20	0	0	37	1	3224	1	0,5	5	9	23	30	3	232243200	37	3,22302E-09
	V5	63	29	14	43	11	0,22	0	0	43	1	1072,3	1	0,5	5	9	26	34	3	6531840000	43	7,08184E-10
	V6	72	37	14	49	14	0,19	0	0	49	1	5332	1	0,5	5	9	34	35	3	4,0824E+11	51	8,64482E-11
Mobile Phone	V1	20	4	6	13	2	0,3	0	0	13	1	411,2	1	0,5	4	4	2	16	2	162	10	0,000154495
	V2	25	10	6	17	4	0,24	0	0	17	2	642,33	2	0,5	4	4	10	15	0	3528	16	0,000105143
Scrolling Text	V1	27	19	0	19	7	0	4	7	30	12	808	13	0,70	3	7	8	8	11	1440	19	1,07288E-05
	V2	29	19	0	19	7	0	8	3	30	12	926,6	12	0,63	4	7	4	10	15	192	19	3,57628E-07
Reference	V1	3	14	10	23	4	0,32	1	10	34	7	989	7	0,5	4	4	12	17	2	87480	24	4,0736E-05
	V2	31	14	9	23	4	0,29	1	0	24	2	975,4	2	0,5	4	4	12	17	2	245760	23	0,000114441
SimulES	V1	32	15	8	25	7	0,25	0	0	25	0	1056,7	0	0	5	8	4	17	11	73728	23	1,71661E-05
	V2	53	26	12	35	11	0,23	0	0	35	0	2894,4	0	0	6	3	9	27	17	97327104	38	1,08055E-08
	V3	59	32	8	38	14	0,14	0	0	38	0	3638,9	0	0	5	3	5	27	27	41140224	40	7,13669E-11
B2B Website	V1	24	0	5	18	0	0,21	0	0	18	0	612	-	0	4	3	0	24	0	32	5	1,90735E-06
	V2	37	8	5	29	2	0,13	0	0	29	2	1431,33	3	0,5	4	3	8	29	0	7200	13	5,23869E-08
Web Collaboration	V1	40	9	21	29	2	0,525	0	0	29	0	1611,2	0	0	4	2	9	31	0	44236800	30	4,02331E-05
	V2	40	6	25	32	2	0,625	0	0	32	0	1609	0	0	4	16	3	34	3	35251200	31	3,20608E-05
FM ERP SPL	V1	42	8	10	29	3	0,24	0	2	31	4	1833,1	4	0,5	4	5	8	34	0	4320	18	9,82254E-10
	V2	57	14	14	37	6	0,25	0	2	39	4	3361,4	4	0,5	6	6	14	43	0	1292544	28	8,96883E-12
E-science	V1	61	52	7	43	16	0,11	1	2	46	2	3853,7	2	0,67	8	3	18	9	34	471859220	59	2,04636E-10
	V2	61	52	7	43	16	0,11	0	0	43	0	3852,3	0	0	8	3	18	9	34	754974740	59	3,27418E-10
Video Player	V1	53	21	17	33	9	0,32	0	0	33	2	2877,1	2	0,5	5	6	21	32	0	844462080	38	9,37541E-08
	V2	71	35	12	53	5	0,17	0	0	53	0	5113,7	0	0	5	6	35	36	0	4,50065E+13	47	1,9061E-08

FM - Modelos de *Features*; V - Versões

com a evolução do modelo de *features*.

7.3.2 QP2. O número de features filhas envolvidas em relacionamentos alternativos (XOR) ou OR tende a aumentar com a evolução do modelo de features?

Como os relacionamentos XOR e OR são pontos de variação, a impressão é que o número de *features* envolvidas nestes tipos de relacionamento tende a aumentar com a evolução do modelo de *features*. A fim de analisar este aspecto foi considerado os dados da Figura 21. Pode-se observar a partir desta Figura, que 34 *features* filhas foram adicionadas em relacionamentos alternativo (XOR), enquanto 14 foram removidas. Além disso, 42 *features* filhas foram adicionadas em relacionamentos OR, enquanto 31 foram removidas. Assim, pode-se argumentar que há uma evidência que o número de *features* filhas envolvidas em relacionamentos alternativos (XOR) e OR tende a aumentar com a evolução do modelo de *features*.

7.3.3 QP3. A profundidade do modelo de features tende a aumentar com a evolução do modelo?

Como mencionado previamente no Capítulo 2, o modelo de *features* é representado com uma estrutura de árvore, que é um conjunto de *features* hierarquicamente organizadas. Há uma forte evidência que o número de *features* cresce com a evolução do modelo. Entretanto, não foram identificadas evidências nesta direção; pelo contrário, a profundidade do modelo de *features* permanece constante.

A fim de analisar este aspecto, considere a medida Profundidade da Árvore (DT) na Tabela 15. Esta medida cresceu em apenas 3 evoluções dos seguintes modelos de *features*: *Mobile Media*, *Ecommerce* and *FM ERP*. No entanto, diminuiu na evolução do modelo de *features VODPlayer*. Além disso, o aumento mais significativo ocorreu na evolução do modelo *Ecommerce*, no qual o valor DT variou de 3 para 8. Esta informação corrobora que há evidência que a profundidade do modelo de *features* tende a permanecer constante. Portanto, como a profundidade da árvore permanece constante, pode-se assumir que as *features* inseridas ao longo do tempo estão distribuídas ao longo da estrutura da árvore.

7.3.4 QP4. A largura do modelo de features tende a aumentar com a evolução do modelo?

A partir das questões de pesquisa QP1 e QP3, pode-se observar que o número de *features* aumenta ao longo do tempo, ao mesmo tempo, a profundidade do modelo de *features* tende a permanecer constante. Assim, pode-se supor que a largura do modelo tende a aumentar.

A fim de analisar este aspecto, considere a medida NLeaf na Tabela 15. Esta medida aumenta em 12 modelos de *features* e suas respectivas evoluções, diminuindo em apenas uma evolução do modelo de *features* (*Eshop)* e permanecendo constante em 3 modelos (*Scrolling Text*, *Reference* e *E-science*). Além disso, o aumento mais significativo ocorreu na evolução do modelo *Toko*, onde o valor NLeaf variou de 14 a 49. Então, como o número de *features* folhas (NLeaf) está aumentando e a profundidade da árvore permanece constante, pode-se concluir pelos dados que a largura da árvore aumenta ao longo da evolução dos modelos. Esta informação reforça evidências que *features* inseridas ao longo do tempo são distribuídas ao longo da estrutura da árvore.

7.3.5 QP5. A variabilidade tende a aumentar com a evolução do modelo de features?

Como destacado previamente no Capítulo 2, relacionamentos opcionais, alternativos (XOR) e (OR) são chamados de pontos de variação. Um ponto de variação representa uma tomada de decisão para diferentes variantes de um ativo da LPS. Uma variação é simplesmente uma diferença entre dois ativos comparáveis (ou conjuntos). Um ponto de variação consiste de um conjunto de possíveis instanciações (variações legais do ponto de variação). Assim, variabilidade descreve as possíveis variações de ativo da LPS com pontos de variação.

Como o modelo de *features* evolui, é natural achar que sua variabilidade cresça. A fim de analisar este aspecto, há algumas medidas que podem ser usadas, tais como: MHoF, RNoF, SHoF, NVC, NVF e RoV. Analisando os valores mostrados na Tabela 15, pode-se observar que a medida RoV tende a diminuir em todas as 16 evoluções dos modelos de *features*. Como $RoV = NVC/NF^2$ e NF aumentam ao longo do tempo, os valores de RoV diminuem. A medida SHoF tende a permanecer estável. Os valores de SHoF permanecem constantes em 9 evoluções de modelos de features, aumentando em 4 e diminuindo em 3 modelos. Como SHoF é baseado no número de relacionamentos XOR e, como ilustrado na Figura 21, só 10 relacionamentos XOR foram adicionados, enquanto 5 relacionamentos XOR foram removidos, o SHoF varia pouco. Por outro lado, NVF aumentou em 11 evoluções de modelos de *features*, enquanto

NVC aumentou em 13 modelos. É importante enfatizar que NVF é uma simples e eficiente medida para variabilidade desde que computa o número de *features* alternativas (NA) e o número de *features* opcionais (NO). Além disso, a Figura 21 mostra que uma pequena quantidade de restrições *cross-tree* entre *features* foram adicionados. Por exemplo, 15 relacionamentos "*requires*" foram incluídos, enquanto a maioria foram excluídos. Só 3 relacionamentos "*excludes*" foram adicionados, enquanto um foi removido. Assim, há uma evidência forte que a variabilidade tende a aumentar com a evolução do modelo de *features*.

7.3.6 QP6. A complexidade cresce com a evolução do modelo de features?

Como o modelo de *features* evolui, é lógico imaginar que sua complexidade aumente. A fim de analisar este aspecto, há algumas medidas que podem ser usadas, tais como: CyC, CTC, DT, NTop, CoC, NF e ComC. Analisando os valores mostrados na Tabela 15, pode-se observar que o valor da medida CyC cresceu em 7 evoluções dos modelo de *features*, mas diminuiu em 6 modelos. A medida CTC aumentou seu valor em 6 modelos e diminuiu em 4 evoluções dos modelos de *features*. As medidas CyC e CTC são baseadas no número de restrições, e como o número de restrições sofre uma variação pequena, mantém um comportamento estável. A medida DT permanece praticamente constante, como mencionado previamente. A medida NTop aumentou em 4 evoluções de modelos de *features*, desde que não é esperado um grande número de inserções de *features* na raiz. A medida CoC aumentou seu valor em 6 modelos, diminuiu em 3 modelos de *features* e permaneceu constante em 7 evoluções de modelos de *features*, mostrando que o número de bordas aumenta e diminui de maneira similar ao número de *features*.

Por outro lado, a medida do número de *features* (NF) aumentou em 12 modelos. Entretanto, a medida mais completa e importante relacionada com complexidade é a medida ComC, desde que diferentes parâmetros são usados em seu cálculo. A medida ComC aumenta em 13 evoluções de modelos de *features*, enquanto diminui em apenas 3 modelos. Assim, pode-se sustentar que há uma evidência que a complexidade tende a aumentar com a evolução do modelo de *features*.

7.4 Discussão dos Resultados

Baseado na investigação das questões de pesquisas, foram analisadas as evoluções do modelo de *features* do ponto de vista da manutenibilidade.

A primeira conclusão do estudo foi que o número de *features* tende a aumentar com a evolução do modelo de *features*. Posteriormente, este crescimento impacta diretamente a capacidade de análise e de modificabilidade do modelo de *features*. A largura do modelo de *features* tende a aumentar ao longo do tempo. Também foi verificado que modelos de *features* largos tendem a ser mais difíceis de se alterar. Assim, a modificabilidade do modelo de *features* diminui. A variabilidade e a complexidade estrutural do modelo de *features* tende a aumentar com o tempo. Dessa forma, a evolução do modelo de *features* gera um impacto em sua capacidade de ser analisável e modificável.

Por exemplo, a medida FoC (Flexibilidade de Configuração) diminui em 9 das 16 evoluções dos modelos de *features*, permanecendo constante em dois modelos e aumentando em 5 evoluções dos modelos de *features*. Posteriormente, a flexibilidade do modelo de *features* tende a aumentar.

Nesse sentido, pode-se concluir, a partir do estudo exploratório realizado, que o processo de evolução dos modelos de features de LPSs tende a diminuir a manutenibilidade do modelo de *features*.

7.5 Ameaças à Validade

Na execução do estudo exploratório foram identificadas algumas ameaças à validade baseadas em Wholin et al. (WOHLIN *et al.*, 2012). Segundo Wholin et al. (WOHLIN *et al.*, 2012) as ameaças à validade de um estudo podem ser classificadas em: validade interna, validade externa, validade de construção e validade de conclusão.

Validade Interna ameaça os fatores que podem influenciar as observações do estudo. Nas observações do estudo, a identificação dos tipos de evolução em cada versão de cada modelo de *features* (veja Figura 21) foi conduzida manualmente. Mesmo esta tarefa tendo sido realizada criteriosamente, alguns erros podem ter acontecido durante este processo.

Validade Externa ameaça o que diz respeito a generalização das descobertas do estudo. O estudo considera apenas 16 modelos de *features* e suas respectivas versões. Este número de modelos não tem significância estatística e pode ser vista como uma ameaça à validade interna. Além do mais, apenas quatro dos modelos de *features* têm ao menos três tipos de diferentes versões (*Mobile Media 6*, *Toko 6*, *Software Product 3* e *SimulES 3*). Esta limitação também pode ser vista como uma ameaça à validade interna de nosso estudo.

Validade de Construção ameaça o que diz respeito ao relacionamento entre teoria e

observação. A preocupação principal do estudo é relacionada a informação do modelo de *features* provida pelo repositório S.P.L.O.T. Primeiro, é assumido no estudo que os modelos de *features* selecionados pelo S.P.L.O.T. são reais (isto é, representam LPSs reais). Muitos modelos do S.P.L.O.T. são exemplos "*toys*" e não são suficientes para esboçar conclusões significativas. Uma outra ameaça está relacionada à identificação da evolução do modelo de *features* (sequência de versões). No estudo é considerado a data de inclusão, nome e ID do modelo de *features* para guiar a identificação de cada evolução de modelo de *features*. Esta informação é extraída diretamente do S.P.L.O.T. e foi assumido que é confiável. Além disso, este processo foi conduzido de maneira manual e alguns erros podem ter acontecido.

7.6 Conclusões

Este capítulo apresentou um estudo exploratório realizado com o objetivo de investigar o impacto do processo de evolução dos modelos de *features* na manutenibilidade destes modelos. Durante este estudo, foi identificado um conjunto de mudanças que são recorrentes na evolução de modelos de *features*. O estudo procurou responder seis questões de pesquisa relacionadas à diferentes subcaracterísticas da manutenibilidade, tais como: analisabilidade, modificabilidade, complexidade estrutural, variabilidade e flexibilidade. Embora não sejam conclusivas, as descobertas indicam que a manutenibilidade dos modelos de *features* tende a diminuir quando os modelos evoluem.

8 ESTUDO DE CASO EXPLORATÓRIO: UTILIZAÇÃO DE MEDIDAS NA AVALIAÇÃO DA QUALIDADE DO MODELO DE *FEATURES*

Este capítulo apresenta um estudo de caso exploratório sobre a utilização de medidas de qualidade na avaliação da manutenibilidade do modelo de *features*. A Seção 8.1 discute o planejamento deste estudo de caso, detalhando as questões de pesquisa, os objetivos definidos, além dos procedimentos de coleta e análise de dados. Os resultados deste estudo de caso são apresentadas na Seção 8.2. Na Seção 8.3 faz-se uma discussão acerca dos resultados obtidos e das implicações para os pesquisadores e profissionais. A Seção 8.4 apresenta as ameaças à validade do estudo e, por fim, a Seção 8.5 conclui este capítulo.

8.1 Planejamento do Estudo de Caso

Um dos objetivos deste estudo de caso exploratório é investigar como as medidas de qualidade do catálogo COfFEE podem ser aplicadas com a finalidade de suportar a avaliação da qualidade do modelo de *features* em LPSs. Para isso, foi utilizado o *dataset* MAcchiATO. O *dataset* MAcchiATO contém os valores de 32 medidas pertencentes ao catálogo COfFEE computadas para 218 modelos de *features* de LPSs extraídos do repositório S.P.L.O.T. Essas 32 medidas foram selecionadas por poderem ser computadas para LPSs, foco deste estudo de caso.

Este estudo de caso foi baseado em Jedlitschka e Pfahl (2005), Kitchenham *et al.* (2008), Robson e McCartan (2016) e seguiu as orientações definidas em Runeson P. e Höst (2009). Os procedimentos de pesquisa foram descritos em um protocolo, seguindo as indicações apontadas por Runeson P. e Höst (2009). Este protocolo detalha a execução do estudo de caso, os métodos e ferramentas utilizadas para a coleta de dados, as técnicas aplicadas para análise de dados e os procedimentos de validade.

8.1.1 Questões de Pesquisa

O principal objetivo deste estudo de caso consiste em explorar como as medidas de qualidade pertencentes ao catálogo COfFEE podem ser utilizadas na avaliação da manutenibilidade do modelo de *features*. Contudo, a utilização dessas medidas envolve alguns desafios, como, por exemplo:

- Aplicar todas as 32 medidas de qualidade do catálogo COfFEE que são voltadas exclusivamente para LPSs (e que compõem o *dataset* MAcchiATO) pode proporcionar uma elevada

sobrecarga de trabalho para o engenheiro de software;

- Algumas medidas distintas, no conjunto de 32 medidas presentes no *dataset* MAcchiATO, podem avaliar aspectos semelhantes. Logo, nem todas precisam ser utilizadas para avaliar a manutenibilidade do modelo de *features*;
- Para que se possa utilizar uma determinada medida é fundamental que esta possua *thresholds*. Porém, muitas das 32 medidas selecionadas não possuem *thresholds*.

Para atingir o objetivo proposto, foram definidas três questões de pesquisa:

- **QP1:** Existem correlações entre as medidas utilizadas para avaliar a qualidade dos modelos de *features* em LPSs? Respondendo a QP1, será possível reduzir a quantidade de medidas a serem utilizadas para avaliar a manutenibilidade de modelos de *features*.
- **QP2:** Como agrupar as medidas relacionadas? Respondendo a QP2, será possível agrupar medidas que cobrem aspectos semelhantes e, consequentemente, reduzir a quantidade de medidas a serem utilizadas para avaliar a manutenibilidade de modelos de *features*.
- **QP3:** Como definir *thresholds* para uma determinada medida? Responder a QP3 é essencial para permitir que as medidas sejam utilizadas na prática, ou seja, em cenários reais.

8.1.2 Seleção dos Objetivos

A fim de especificar o contexto para a realização deste estudo de caso, o primeiro passo foi definir as medidas de qualidade e os modelos de *features* a serem utilizados. Esta seleção foi baseada no método de amostragem de conveniência (WOHLIN *et al.*, 2012), que é um tipo de técnica de amostragem não probabilística com base no julgamento do pesquisador. Este método foi utilizado devido ao fato das medidas serem selecionadas com base em revisões da literatura e os modelos de *features* terem sido extraídos de um repositório público. Assim, o *dataset* MAcchiATO foi concebido por meio da utilização deste método.

8.1.3 Procedimentos para a Coleta dos Dados

Os dados utilizados neste estudo foram extraídos do *dataset* MAcchiATO. A construção do *dataset* MAcchiATO foi descrita na Seção 6.3.

8.1.4 Procedimentos para Análise de Dados

Os dados coletados em um estudo de caso podem ser quantitativos ou qualitativos. Os dados quantitativos envolvem classes e números, enquanto os dados qualitativos envolvem descrições, figuras e diagramas. Os dados quantitativos são analisados utilizando métodos estatísticos, enquanto os dados qualitativos são analisados utilizando a classificação e categorização. Neste estudo de caso, foi utilizada uma combinação de dados qualitativos e quantitativos, que é conhecido como método misto (RUNESON P. E HÖST, 2009).

A análise dos dados é executada de forma distinta para os dados quantitativos e qualitativos. A análise dos dados quantitativos, em geral, envolve métodos baseados em estatística descritiva e análise de correlação. Os métodos de estatística descritiva envolvem médias, desvios-padrão, histogramas e gráficos de dispersão, com a finalidade de facilitar a compreensão dos dados coletados. A análise de correlação é realizada com o objetivo de descrever a forma como uma medida se relaciona com outra. Para os dados qualitativos, o principal objetivo da análise é extrair conclusões, mantendo uma clara cadeia de provas. Isto é, um leitor deve ser capaz de seguir a derivação dos resultados e chegar às conclusões a partir dos dados coletados. Além disso, os avanços das técnicas de aprendizagem de máquina e análise de dados têm fornecido meios eficazes de extrair informações úteis e conhecimento a partir dos dados. Neste estudo de caso exploratório foram utilizadas diferentes técnicas de análise de dados, tais como: estatística descritiva, análise de correlação, Análise dos Componentes Principais (PCA) (Análise de Componentes Principais), além de métodos paramétricos e não paramétricos para definir os *thresholds*.

8.2 Respondendo as Questões de Pesquisa

Nesta seção, as conclusões do estudo de caso são discutidas e as respostas das questões da pesquisa são apresentadas.

8.2.1 Análise dos Dados

O objetivo da Análise Exploratória de Dados (AED) consiste em examinar os dados previamente à aplicação de qualquer técnica estatística. Desta forma, o analista consegue um entendimento básico de seus dados e das relações existentes entre as variáveis analisadas. A análise exploratória pode empregar grande variedade de técnicas gráficas e quantitativas,

buscando maximizar a obtenção de informações acerca dos dados. A análise descritiva permite ao pesquisador familiarizar-se com os dados, organizá-los e sintetizá-los de forma a obter as informações necessárias do conjunto de dados para responder as questões que estão sendo estudadas.

Neste sentido, conhecer a distribuição dos dados é essencial para escolher o método estatístico mais adequado para sua análise. Contudo, esta atividade requer julgamento e experiência e, normalmente, implica em um processo iterativo que envolve a seleção de uma distribuição, a estimativa de parâmetros e a verificação da conformidade desta distribuição aos dados coletados. Neste estudo de caso exploratório, foi utilizada a ferramenta *EasyFit*[1], a fim de determinar a função de distribuição de probabilidade que pode ser utilizada para melhor descrever cada uma das 32 medidas no *dataset* MAcchiATO. A ferramenta *EasyFit* utiliza o teste de *Kolmogorov-Smirnov* (a partir de uma ou duas amostras de teste K-S), que é um teste não paramétrico de igualdade de distribuição de probabilidades que pode ser utilizado para comparar uma amostra com uma distribuição de referência (uma amostra de teste K-S) ou para comparar duas amostras (duas amostras de teste K-S). Esta ferramenta foi escolhida para o estudo por ser uma ferramenta livre e bastante utilizada pela academia para execução de análises estatísticas.

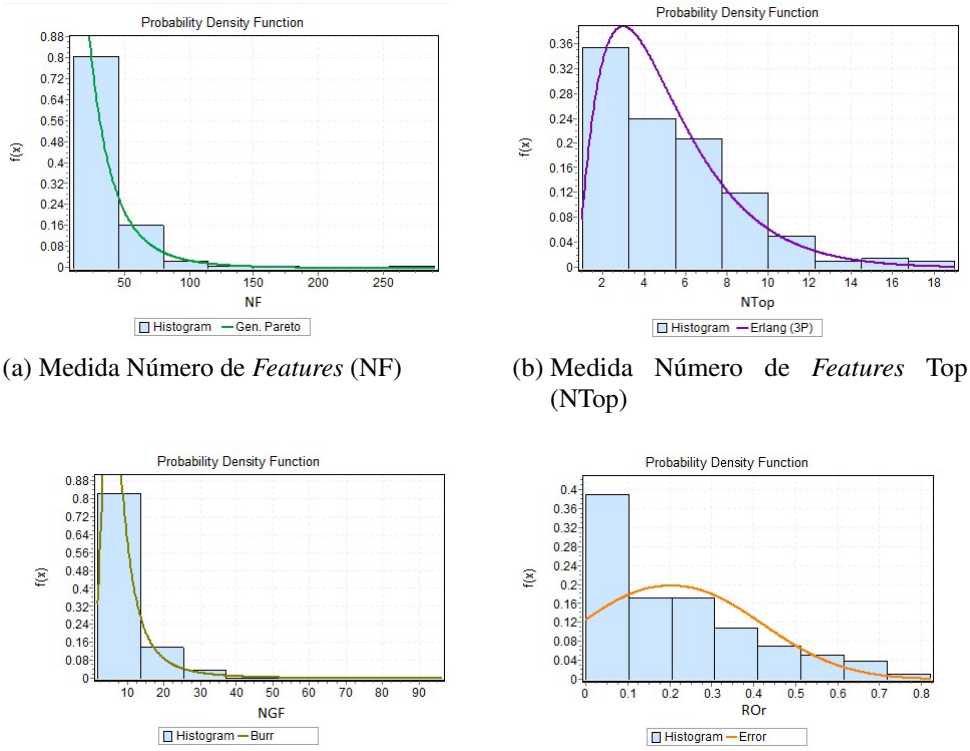
A Figura 22 ilustra o histograma e a função de distribuição de probabilidade que melhor descreve (de acordo com a ferramenta *EasyFit*) as medidas: Número de *Features* (NF), Número de *Features* Top (NTop), Número de Agrupamentos de *Features* (NGF) e Taxa de *Features* Or (ROr).

A Figura 22 mostra que as medidas NF, NTop, NGF e ROr seguem diferentes funções de distribuição de probabilidade. Mais precisamente, NF, NTop, NGF e ROr seguem as funções de distribuições e probabilidade Gen. Pareto, Erlang (3P), Burr e Erro. É importante ressaltar que nenhuma das 32 medidas no *dataset* MAcchiATO possui uma distribuição normal (ou gaussiana).

8.2.2 QP1: Existem correlações entre as medidas utilizadas para avaliar a qualidade dos modelos de features em LPSs?

O coeficiente de correlação mede o grau em que duas variáveis tendem a mudar juntas. Assim, ele descreve tanto a força quanto a direção da relação entre estas duas variáveis. As correlações entre as variáveis podem ser medidas com a utilização de diferentes coeficientes. Os dois coeficientes mais comuns são: o coeficiente de *Pearson* e o coeficiente de correlação de

[1] http://www.mathwave.com/



(a) Medida Número de *Features* (NF)

(b) Medida Número de *Features* Top (NTop)

(c) Medida Número de Agrupamento de *Features* (NGF)

(d) Medida Taxa de *Features* Or (ROr)

Figura 22 – Histograma e a função de distribuição de probabilidade que melhor descreve as medidas NF, NTop, NGF e ROr.

Spearman.

O coeficiente de correlação de *Pearson* mede a força da relação linear entre duas variáveis contínuas e com distribuição normal (Gaussiana). Uma relação é linear quando uma mudança de uma variável está associada com uma variação proporcional na outra variável.

O coeficiente de correlação de *Spearman* avalia a relação monotônica entre duas variáveis contínuas ou ordinais. Em uma relação monotônica, as variáveis tendem a mudar em conjunto, mas não necessariamente, a uma taxa constante. Além disso, o coeficiente de correlação de *Spearman* é um teste não-paramétrico (MCCRUM-GARDNER, 2008).

Assim, ao contrário do coeficiente de correlação de *Pearson*, o coeficiente de *Spearman* não requer a suposição de que a relação entre as variáveis é linear, nem faz quaisquer suposição sobre a distribuição da frequência das variáveis (WOHLIN *et al.*, 2012).

A primeira hipótese desse estudo de caso é que existe uma correlação significativa entre duas medidas no *dataset* MAcchiATO (H1). Assim, a hipótese nula (H0) é que não há nenhuma correlação entre essas medidas. A fim de testar a hipótese H1, a utilização do coeficiente de *Pearson* é revelada inadequada, uma vez que exige que as variáveis sigam uma distribuição

normal (Gaussiana) e nenhuma medida no *dataset* MAcchiATO segue essa distribuição. Por esse motivo, foi utilizado um teste de correlação com base nos estudos propostos em (BAGHERI E. E GASEVIC, 2011; BERGER T. E GUO, 2014; COURTNEY; GUSTAFSON, 1993). Desta forma, para indicar a correlação estatística entre duas medidas foi utilizado o coeficiente de correlação de *Spearman* (com nível de significância de 0.05), o qual mede o grau de correlação linear entre duas variáveis quantitativas (x e y). Este coeficiente é definido pela seguinte equação:

$$\rho = 1 - \frac{6\sum d_i^2}{n(n^2-1)} \tag{8.1}$$

onde:

$d_i = x_i - y_i$

n = o número de pares dos valores.

Para entender melhor o coeficiente de correlação, foi utilizada a seguinte escala (SALKIND; RAINWATER, 2003; BERGER T. E GUO, 2014):

- 0.8 to 1.0 or -0.8 to -1.0: correlação muito forte;
- 0.6 to 0.8 or -0.6 to -0.8: correlação forte;
- 0.4 to 0.6 or -0.4 to -0.6: correlação moderada;
- 0.2 to 0.4 or -0.2 to -0.4: correlação fraca;
- 0.0 to 0.2 or 0.0 to -0.2: correlação muito fraca ou não existe relação.

Para realizar a análise de correlação utilizou-se o *dataset* MAcchiATO. Estes dados foram normalizados para cada uma das 32 medidas. O processo de normalização utilizou a seguinte fórmula de cálculo:

$$\mathrm{Z} = \frac{x - \mu}{\sigma} \tag{8.2}$$

onde:

Z é o valor absoluto da normalização;

x é o valor da medida;

μ é a média da população;

σ é o desvio padrão da população.

Conforme mencionado anteriormente, a análise de correlação utilizada nesta pesquisa é qualitativa e quantitativa. Primeiro, foram filtradas as estimativas fora dos coeficientes de correlação sem significância estatística (ρ-value ≥ 0.05). Então, foram inspecionadas as correlações fortes para encontrar explicações baseadas no conhecimento dos modelos de *features*. Em seguida, foram discutidas tais correlações significativas e fornecidas interpretações cuidadosas.

A análise da correlação entre duas medidas é importante para descobrir se estas cobrem aspectos similares do modelo de *features*. Os resultados deste estudo são apresentados na Tabela 75 do Apêndice **??**. Um resumo destes resultados é apresentado nas Tabelas 16 e 17. As tabelas foram divididas por questão de tamanho, para facilitar a análise de correlação. Nessas Tabelas, cada célula mostra o grau de correlação entre duas medidas. Note que apenas os valores classificados com correlação muito forte e forte são apresentados.

Tabela 16 – Correlação de *Spearman* para medidas de qualidade (parte 1).

Medidas	NF	NO	NLeaf	DTMax	DTMean	CogC	FEX	SCDF	MCDF	CyC	ComC
NLeaf	0.98	-	1	-	-	-	0.98	-	-	-	0.97
DTMax	-	-	-	1	0.70	0.61	-	-	-	-	-
DTMean	-	-	-	0.70	1	-	-	-	-	-	-
DTMedian	-	-	-	-	0.75	-	-	-	-	-	-
CogC	0.60	-	-	0.61	-	1	0.60	-	-	-	0.61
FEX	0.95	-	0.98	-	-	0.60	1	-	-	-	0.95
FoC	-	0.76	-	-	-	-	-	-	-	-	-
SCDF	-	-	-	-	-	-	-	1	-	0.61	-
MCDF	-	-	-	-	-	-	-	-	1	0.64	-
CyC	-	-	-	-	-	-	-	0.61	0.64	1	-
ComC	1	-	0.97	-	-	0.61	0.95	-	-	-	1
NGV	0.88	-	0.78	0.69	0.69	-	0.62	0.75	-	-	0.89
CTCV	-	-	-	-	-	-	-	0.61	0.66	0.98	-
CTC	-	-	-	-	-	-	-	0.61	0.64	0.94	-
RCon	-	-	-	-	-	-	-	0.61	0.64	0.93	-
RDen	-	-	-	-	-	-	-	-	-	0.81	-
CoC	1	-	0.98	-	-	0.60	0.95	-	-	-	1
NVF	0.85	-	0.83	-	-	0.76	0.84	-	-	-	0.85
SHoF	-	-	-	-	-	0.67	-	-	-	-	-
MHoF	-	-	-	-	-	0.75	-	-	-	-	-
RNoF	0.69	0.76	0.67	-	-	-	0.60	-	-	-	0.68
BF Max	0.67	-	0.76	-	-	-	0.75	-	-	-	0.65
NGOr	-	-	-	-	-	0.78	-	-	-	-	-
NGXOr	-	-	-	-	-	0.67	-	-	-	-	-
ROr	-	-	-	-	-	0.62	-	-	-	-	-

Pode-se observar na Tabela 11 que todas as medidas no *dataset* MAcchiATO estão relacionadas com a característica de qualidade de manutenibilidade. A manutenibilidade é subdividida em 7 subcaracterísticas de qualidade: analisabilidade, complexidade cognitiva, extensibilidade, flexibilidade, modularidade, complexidade estrutural e variabilidade. Assim, pode-se analisar os resultados das correlações, apresentados nas Tabelas 16 e 17, agrupando as medidas no *dataset* MAcchiATO pelas subcaracterísticas da manutenibilidade.

Analisabilidade e Extensibilidade. O NLeaf (uma medida de analisabilidade) e

FEX (uma medida de extensibilidade) possuem forte correlação (0,98), porque FEX usa NLeaf em sua fórmula de cálculo. Ambas as medidas NLeaf e FEX tem forte correlação com as medidas ComC, NGV, CoC, NVF, RNoF e BFMax (medidas de complexidade estrutural e variabilidade). Assim, se um modelo de *features* tem um alto valor para NLeaf e FEX, provavelmente ele terá um valor alto para as medidas ComC, NGV, CoC, NVF, RNoF e BFMax. Então, este resultado evidencia que essas medidas abrangem aspectos similares do modelo de *features* e que é possível usar apenas o NLeaf e/ou o FEX para medir esses aspectos.

Complexidade Cognitiva. CogC possui forte correlação com diferentes medidas de variabilidade, complexidade e extensibilidade. CogC é a única medida de complexidade cognitiva. Então, este aspecto é coberto por essa medida.

Flexibilidade. FoC é uma medida de flexibilidade e tem uma forte correlação com a medida NO, uma medida de variabilidade, porque FoC utiliza NO em sua fórmula de cálculo. Então, este resultado mostra que as medidas FoC e NO cobrem aspectos similares do modelo de *features* e que pode-se usar apenas a medida FoC para medir esse aspecto.

Modularidade. SCDF e MCDF são medidas de modularidade e possuem forte correlação positiva com algumas medidas estruturais, como: CyC, CTCV, CTC e RCon. É importante notar que todas esses medidas estão relacionadas com as restrições do modelo de *features*, cobrindo aspectos semelhantes. Assim, pode-se utilizar apenas SCDF e MCDF em substituição as medidas CyC, CTCV, CTC e RCon.

Complexidade Estrutural. NF é uma medida de complexidade estrutural que tem uma forte correlação com NLeaf (uma medida relacionada com a largura do modelo de *features*). No entanto, NF não tem correlação com as medidas de profundidade, tais como, DTMax, DTMean e DTMedian. Assim, NF e DTMax cobrem diferentes aspectos.

Além disso, NF tem uma forte correlação positiva com NGV e NGF (medidas de variabilidade). Isto indica que quando o NF aumenta, a variabilidade do modelo de *features* também aumenta. NF também tem uma forte correlação positiva com ComC e CoC (medidas de complexidade). O valor 1 referente a correlação entre ComC e CoC indica que eles crescem na mesma proporção. Isso pode ocorrer porque as medidas de ComC e CoC utilizam a medida NF na sua fórmula de cálculo. Assim, pode-se usar apenas NF em vez do conjunto composto pelas medidas NLeaf, CogC, FEX, ComC, NGF, CoC, NVF, RNoF e BF Max. Por outro lado, DTMax e DTMean tem forte correlação com DTMedian e NGF. Então, este resultado evidencia que essas medidas abrangem aspectos similares do modelo de *features* e que pode-se usar apenas DTMax

para medir esses aspectos. Da mesma forma, pode-se ver na Tabela 17 que as medidas RDen CyC, CTCV, CTC e RCon possuem alta correlação entre elas. Estas medidas estão relacionadas com a complexidade do modelo de *features*.

É importante ressaltar que todas estas medidas consideram as restrições do modelo de *features* na sua fórmula de cálculo. Isto explica a forte correlação entre essas medidas. Como as medidas SCDF e MCDF já possuem uma forte correlação com as medidas CyC, CTCV, CTC e RCon, poderíamos usar apenas RDen para medir esses aspectos semelhantes. A medida BFMax possui forte correlação com as medidas NF, NLeaf, FEX, ComC e BFMean. Todas estas medidas estão relacionadas com o número de nós na árvore do modelo de *features*. No entanto, como NF tem uma forte correlação com BFMax, pode-se utilizar apenas NF ao invés de todas estas medidas.

Variabilidade. O número de *features* obrigatórias (NM) possui uma forte correlação positiva com a medida RNoF. Ambas as medidas estão relacionadas com a variabilidade do modelo de *features*. Então, este resultado evidencia que pode-se simplesmente usar NM vez de RNoF. Outros exemplos de medidas de variabilidade são NGOr, NGXOr, ROr e RXOr. Estas medidas utilizam na sua fórmula de cálculo o número de grupos Or e XOr e o número de *features* que participam de grupos Or e XOr. De acordo com a Tabela 17, existe uma forte correlação entre NGOr e ROr, e entre NGXOr e RXOr. Além disso, NGOr e ROr possuem uma correlação muito forte com MHoF. As medidas NGXOr e RXOr são fortemente correlacionadas com a medida SHoF. Então, este resultado evidencia que pode-se utilizar apenas NGOr e NGXOr, em vez de ROr, RXOr, SHoF e MHoF. Por outro lado, NTop, RoV e NVC não têm forte correlação com qualquer outra medida. Isto sugere que estas medidas abrangem aspectos específicos do modelo de *features* e elas não estão sobrepondo outras medidas.

Analisando as correlações entre as 32 medidas do catálogo COfFEE, pode-se argumentar que não é necessário o uso das 32 medidas, a fim de avaliar a manutenibilidade do modelo *feature*. Seria suficiente usar o conjunto composto pelas seguintes medidas: NF, NM, NLeaf, NTop, DTMax, CogC, FEX, FoC, SCDF, MCDF, RDen, RoV, NVC, NGOR e NGXOr.

8.2.3 QP2: Como agrupar as medidas relacionadas?

Compreende-se que as medidas individualmente não são suficientes para a caracterização da qualidade de um modelo de *features*. Estas medidas são muitas vezes calculadas de forma individual e elas não dão indicação da qualidade adequada. Para lidar com este pro-

Tabela 17 – Correlação de *Spearman* para medidas de qualidade (parte 2).

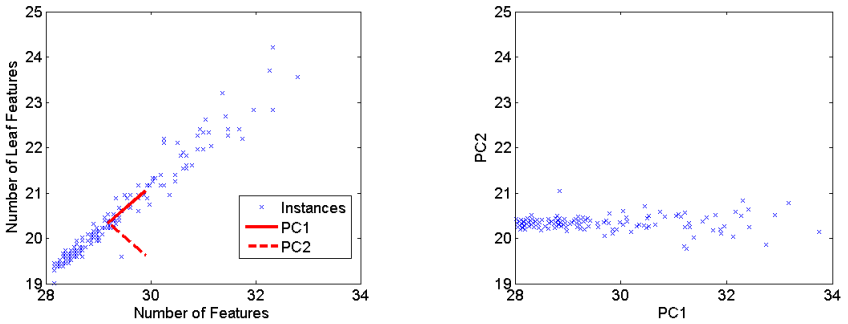
Medidas	NGF	CTCV	CTC	RCon	RDen	CoC	SHoF	MHoF	RNoF	NGOr	NGXOr	ROr	RXOr
NF	0.88	-	-	-	-	1	-	-	0.69	-	-	-	-
NO	-	-	-	-	-	-	-	-	0.76	-	-	-	-
NM	-	-	-	-	-	-	-	-	0.79	-	-	-	-
NLeaf	0.78	-	-	-	-	0.98	-	-	0.67	-	-	-	-
CogC	0.62	-	-	-	-	0.60	0.67	0.75	-	0.78	0.67	0.62	-
FEX	0.75	-	-	-	-	0.95	-	-	0.60	-	-	-	-
SCDF	-	0.61	0.61	0.61	-	-	-	-	-	-	-	-	-
MCDF	-	0.66	0.64	0.64	-	-	-	-	-	-	-	-	-
CyC	-	0.98	0.94	0.93	0.81	-	-	-	-	-	-	-	-
ComC	0.89	-	-	-	-	1	-	-	0.68	-	-	-	-
CTCV	-	1	0.96	0.95	0.85	-	-	-	-	-	-	-	-
CTC	-	0.96	1	0.99	0.86	-	-	-	-	-	-	-	-
RCon	-	0.95	0.99	1	0.87	-	-	-	-	-	-	-	-
RDen	-	0.85	0.86	0.87	1	-	-	-	-	-	-	-	-
CoC	0.88	-	-	-	-	1	-	-	0.69	-	-	-	-
SHoF	-	-	-	-	-	-	1	-	-	-	0.98	-	0.92
MHoF	-	-	-	-	-	-	-	1	-	0.97	-	0.92	-
RNoF	-	-	-	-	-	0.69	-	-	1	-	-	-	-
BF Max	-	-	-	-	-	0.67	-	-	-	-	-	-	-
NGOr	-	-	-	-	-	-	-	0.98	-	1	-	0.91	-
NGXOr	-	-	-	-	-	-	0.98	-	-	-	1	-	0.91
ROr	-	-	-	-	-	-	-	0.92	-	0.91	-	1	-
RXOr	-	-	-	-	-	-	0.92	-	-	-	0.91	-	1

blema, têm sido propostas estratégias de agrupamento de medidas para avaliação da qualidade (SHLENS, 2014; BERKHIN, 2006; WANG, 2005). Dessa forma, foi utilizada neste trabalho a técnica de agrupamento denominada Análise de Componentes Principais (PCA), por ser uma técnica bastante utilizada pela comunidade acadêmica para agrupamento de dados. Com o PCA, um número de combinações de medidas não correlacionadas são selecionadas para capturar informações sobre o modelo de *features* como um todo.

PCA é um procedimento matemático que utiliza transformação ortogonal para converter um conjunto de variáveis (dimensões), possivelmente correlacionadas, para um conjunto de variáveis não correlacionadas linearmente chamadas de componentes principais (Componentes Principais (PCs)). Seguindo o exemplo de PCA ilustrado em Bro e Smilde (2014), foi selecionado um exemplo de PCA utilizando duas medidas do catálogo COfFEE, número de *features* (NF) e número de *features* folhas (NLeaf), e os dados armazenados no *dataset* MAcchiATO.

A Figura 23 mostra a plotagem das observações de NF e NLeaf. Aparentemente, observando a Figura 23, NF e NLeaf são aproximadamente correlacionados. Assim, pode-se argumentar que os modelos de *features* com um alto valor de NF tendem a ter um valor alto de NLeaf e vice-versa (Figura 23a). Na Figura 23, cada modelo de *features* no *dataset* MAcchiATO é representado nas coordenadas (NF, NLeaf). Note que pela inclinação, estes dois eixos (NF e NLeaf) são aproximadamente 45 graus em que pode capturar a maior parte da variabilidade ao longo de um único eixo (Figura 23b). Assim, para o exemplo ilustrado na Figura 23, NF e NLeaf podem ser substituídos pelo Componente Principal 01 (PC01), enquanto ainda capturam a

distribuição completa. Na verdade, se NF e NLeaf são altamente correlacionados, seria possível utilizar, por exemplo, a média ou a soma destas duas medidas (NF e NLeaf) como uma nova variável (PC01) que poderiam substituir eles. Nenhuma informação seria perdida, uma vez que seria sempre possível ir por exemplo para a média das duas variáveis originais.



(a) NF vs NLeaf.

(b) Uma transformação Ortogonal de NF and NLeaf.

Figura 23 – Um exemplo de PCA (Figure 23a) para NF vs NLeaf. A Figura 23b mostra uma transformação ortogonal de NF e NLeaf.

Portanto, se um algoritmo encontra um ângulo de rotação dos eixos de tal forma que a variabilidade máxima é preservada, ele pode ajudar a descobrir onde a correlação reside e quais eixos podem ser abandonados, removendo redundâncias nos dados. O PCA faz exatamente isso. Além disso, o PCA diz quanto de variabilidade os eixos rodados tendem a capturar (ABDI; WILLIAMS, 2010). Assim, um benefício do PCA é quantificar a importância de cada variável (dimensão) para descrever a variabilidade de um conjunto de dados. Em particular, a medida da variância ao longo de cada componente principal (PC) proporciona um meio para comparar a importância relativa de cada uma das variáveis (dimensão).

É esperado com aplicação deste método que a variação utilizando um pequeno número de componentes principais proporcione uma razoável caracterização do *dataset* completo. Com o mínimo esforço, o PCA prevê um roteiro de como reduzir um conjunto de dados complexos para uma dimensão inferior para revelar estruturas escondidas, simplificadas e que estão subjacentes (SHLENS, 2014).

Na Seção 8.2.2 foi apresentada a solução de um subconjunto de 15 medidas a partir das 32 medidas do catálogo COfFEE. Neste trabalho, o PCA foi executado utilizando estas 15 medidas como entrada. Além disso, os dados extraídos do MAcchiATO foram normalizados. Para isso, foi usada uma regra de seleção para os componentes principais que respondem por

mais de 95% da variação acumulada. Foi utilizado o software estatístico R[2] para implementar a técnica de PCA e correlação.

Assim, para selecionar os componentes principais (PCs) foi calculado o desvio padrão, a proporção de variação e a proporção acumulada de variação, de acordo com a Tabela 18. O PCA calcula a combinação das variáveis (medidas de qualidade), tais que as novas variáveis (PCs) possuem um grande desvio padrão. Assim, em geral, um desvio padrão maior significa uma variável melhor. Na Tabela 18, a linha "desvio padrão" significa o desvio padrão das novas variáveis (PC01 até PC09). A próxima linha na Tabela 18 mostra a proporção de variação, isto é, a relação entre a variação do PC e a variação total (a soma das variações). A terceira linha na Tabela 18, chamada proporção cumulativa, indica a quantidade de informação relativa aos dados originais que podem ser descritos pela combinação das novas variáveis.

Tabela 18 – Proporção cumulativa dos PCs.

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9
Desvio Padrão	2.387	1.425	1.265	1.087	1.034	0.970	0.919	0.755	0.647
Proporção da Variação	0.380	0.135	0.107	0.079	0.071	0.063	0.056	0.038	0.028
Proporção Cumulativa	0.380	0.515	0.622	0.701	0.772	0.835	0.891	0.929	0.957

A Figura 24 apresenta a proporção cumulativa para cada um dos componentes principais. É necessário selecionar os PCs com proporção acumulada de variação de até 95%, conforme mencionado anteriormente. Dessa forma, foram selecionados os primeiros 9 PCs (PC1, PC2, ..., PC8, PC9).

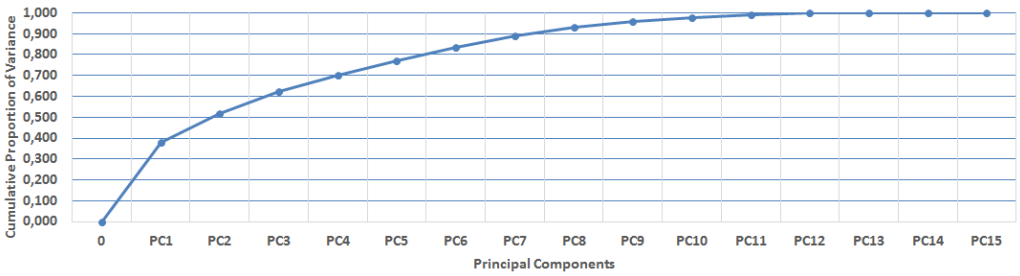


Figura 24 – Seleção dos componentes principais.

O resultado do PCA é apresentado na Tabela 19. Nesta Tabela, as colunas representam os 9 PCs selecionados e as linhas representam as 15 medidas de qualidade selecionadas na análise de correlação (ver Seção 8.2.2). Para cada célula é atribuído um valor entre -1 e 1, que significa a representatividade da medida no PC. Se este valor é próximo de 1 ou -1, a medida pode ser considerada como representante no PC.

[2] http://www.r-project.org/

Para destacar os coeficientes mais significativos para cada medida em um PC, estes foram destacados em negrito (ver Tabela 19). Cada PC representa o agrupamento das 15 medidas, compondo as novas medidas agrupadas (9 PCs).

A maioria dos PCs tem os percentuais distribuídos em várias medidas conforme ilustrado na Tabela 19. Por exemplo, PC01 mostra 5 coeficientes representativos (ver NF, NLeaf, CogC, FEX e NGOr). O agrupamento destas medidas permite a explicação da manutenibilidade do modelo de *features*, uma vez que não existem medidas relacionadas com a complexidade do modelo de *features* (e.g., NF e CogC), a variabilidade do modelo de *features* (e.g., NGXOr), a extensibilidade do modelo (e.g., FEX) e a analisabilidade do modelo (e.g., NLeaf).

Outros exemplos de várias medidas que podem ser adicionadas são apresentadas no PC02 (e.g., RoV e NGXoR), relacionadas com a variabilidade do modelo de *features*, PC05 (e.g., NTop, FoC e RoV), relacionadas com a flexibilidade do modelo, e PC08 (e.g., DTMax, SCDF e RDen), relacionadas com a complexidade do modelo de *features*. Para que estas medidas tenham uma maior explicação, elas precisam ser listadas em um modelo preditivo, que deriva uma nova medida.

Tabela 19 – Componentes principais (PCs).

	PC01	**PC02**	**PC03**	**PC04**	**PC05**	**PC06**	**PC07**	**PC08**	**PC09**
NF	**0.403**	0.119	0.022	-0.114	0.057	-0,084	0.007	0.022	-0.098
NM	0.269	0.317	0.019	-0.188	-0.289	-0.023	0.335	-0.259	**-0.441**
NTop	0.076	0.078	0.386	-0.167	**0.517**	**0.600**	-0.099	-0.116	-0.162
NLeaf	**0.393**	0.157	0.074	-0.139	0.015	-0.065	-0.057	0.035	-0.066
DT Max	0.271	-0.257	-0.260	-0.131	0.061	-0.273	-0,042	**-0.438**	0.037
CogC	**0.354**	-0.297	-0.091	-0.037	0.069	0.176	-0.010	0.109	0.287
FEX	**0.394**	0.122	0.141	-0.026	-0.024	-0.086	-0.156	0.024	-0.095
FoC	-0.045	0.336	0.038	-0.107	**0.609**	**-0.553**	-0.176	0.143	0.123
SCDF	0.053	-0.302	**0.566**	0.096	-0.164	-0.177	-0.036	**0.463**	-0.285
MCDF	0.159	0.025	0.130	**0.581**	-0.145	-0.061	**-0.621**	-0.308	-0,049
RDEn	0.019	-0.146	**0.53**8	0.212	0.107	-0.224	**0.509**	**-0.415**	0.333
RoV	0.024	**0.468**	0.250	-0.225	**-0.415**	0.101	-0.146	0.058	**0.607**
NVC	0.281	0.137	-0.166	0.369	0.029	-0.082	0.324	0.441	0.061
NGOr	**0.339**	-0.053	-0.161	0.279	0.156	0.306	0.112	0.072	0.254
NGXOr	0.163	**-0.467**	0.065	**-0.466**	-0.096	-0.116	-0.179	0.096	0.163

A fim de ilustrar como um engenheiro de domínio pode utilizar os resultados do PCA, foi elaborado um exemplo para utilizar a medida agrupada representada pelo PC03 ilustrado na Tabela 19.

A fórmula de cálculo para PC03, derivada a partir dos resultados da Tabela 19, é:

$$\begin{aligned} PC03 = {} & 0.022NF + 0.0199NM + 0.386NTop + 0.074NLeaf - 0.260DTMax \\ & - 0.091CogC + 0.141FEX + 0.038FoC + 0.566SCDF + 0.130MCDF \\ & + 0.538RDen + 0.250RoV - 0.166NVC - 0.161NGOr + 0.065NGXOr \end{aligned} \tag{8.3}$$

Em seguida, considere um modelo de *features* específico, da LPS "Mobile Phone Software", contido no *dataset* MAcchiATO. A Tabela 20 apresenta os valores das 15 medidas utilizadas como entrada para o PCA do modelo de *features* da LPS "Mobile Phone Software". Note que o valor do PC03 para este modelo é 2.70.

Tabela 20 – Valores das medidas para o modelo de *features* "Mobile Phone Software".

Medida	Valor da medida	Peso da medida no PC03
NF	10	0.22
NM	2	0.04
NTop	4	1.54
NLeaf	7	0.52
DT Max	3	0.78
CogC	2	0.18
FEX	10	1.41
FoC	0.2	0.00
SCDF	2	1.13
MCDF	1	0.13
RDen	1	0.54
RoV	2.42	0.60
NVC	2.2	0.55
NGOr	1	0.16
NGXOr	1	0.06
Valor do PC03		**2.70**

8.2.4 QP3: Como definir thresholds para uma determinada medida?

Conforme mencionado no Capítulo 5, as medidas de qualidade de software fornecem os meios para valoração das subcaracterísticas de qualidade do modelo de *features*. No entanto, para que uma medida possa ser utilizada na prática é fundamental que sejam conhecidos seus limites superiores e inferiores. Somente com esses limites definidos é possível afirmar se um determinado valor para uma medida específica é adequado ou não.

Nos últimos anos, foram propostas várias estratégias para derivar *thresholds* para medidas de código, tais como: *thresholds* derivados da experiência, *thresholds* para análise de medidas, *thresholds* que utilizam modelos de erros e *thresholds* usando técnicas de *cluster*. No entanto, não foram identificadas abordagens para definir *thresholds* para medidas de qualidade do modelo de *features*. Alguns estudos (BAGHERI E. E GASEVIC, 2011; BERGER T. E GUO,

2014) definem medidas para o modelo de *features* e analisam estas medidas, mas eles não definem *thresholds*. Normalmente, os *thresholds* das medidas são definidos com base na experiência pessoal dos especialistas.

Uma possível estratégia para definir *thresholds* é usar a "regra de três-sigma", que é usada para definir os limites de controle superior e inferior em gráficos de controle estatísticos. Estes gráficos são usados para estabelecer limites para um processo industrial ou de negócios que está em um estado de controle estatístico. Em estatística, a chamada regra $68-95-99.7$ é uma abreviatura usada para relacionar o percentual de valores que se encontram em torno da média de uma distribuição normal com uma largura de um (68,27%), dois (95,45%) e três (99,73%) desvios-padrão, respectivamente. Assim, a "regra de três-sigma" expressa uma heurística em que "quase todos" os valores são levados a estar dentro de três desvios padrão da média. A "regra de três-sigma" está relacionada a um resultado que afirma que, mesmo para variáveis não distribuídas normalmente, pelo menos 98% dos casos deve cair dentro de intervalos de três sigma devidamente calculados (KAN, 2002).

Inicialmente, foi definida uma estratégia dirigida pelos dados que pode ser usada para o cálculo dos *thresholds* para um conjunto de medidas de qualidade. A estratégia proposta baseia-se em três requisitos fundamentais: i) respeitar as propriedades estatísticas das medidas, tais como a escala e distribuição; ii) ser orientada a dados, isto é, ter como base a análise de dados a partir de um conjunto representativo de modelos de *features*; e iii) ser simples de executar e repetível.

Com base na "regra de três sigma" e no *dataset* MAcchiATO, foram definidos *thresholds* para os 9 PCs selecionados na Seção 8.2.3. O limite superior, a média e o limite inferior de cada PC é ilustrado na Tabela 21. Esses limites podem ajudar nas avaliações de qualidade iniciais dos modelos de *features* em LPSs.

No entanto, como é ilustrado na Tabela 21, os limites superior e inferior definidos por esta primeira estratégia tornaram-se muito frouxos. Por exemplo, apenas 1 dos 218 modelos de *features* no *dataset* MAcchiATO estava fora do intervalo definido para os *thresholds*, mais especificamente, o modelo de *features* da LPS "Electronic Shopping" estava fora do alcance para os 9 PCs.

Desta forma, foi então investigada uma segunda estratégia para definir os *thresholds* dos PCs, na qual é possível definir os *thresholds* com a utilização de um intervalo de tolerância. Em estatística, um intervalo de tolerância é um intervalo estatístico em que uma determinada

Tabela 21 – *Thresholds* definidos baseados na "regra de três-sigma".

Medida	Limite Inferior (LL)	Média	Limite Superior (UL)
PC01	2.52E+48	5.82E+46	-2.63E+48
PC02	-1.23E+48	2.84E+46	1.29E+48
PC03	-1.56E+48	-3.45E+46	1.49E+48
PC04	-3.31E+48	7.65E+46	3.46E+48
PC05	-2.63E+47	6.08E+45	2.75E+47
PC06	-7.66E+47	-1.69E+46	7.32E+47
PC07	-2.91E+48	6.73E+46	3.05E+48
PC08	-3.96E+48	9.14E+46	4.14E+48
PC09	-5.47E+47	1.26E+46	5.73E+47

proporção de uma amostra de população cai com algum nível de confiança. Assim, neste trabalho foi utilizado um intervalo de tolerância de 95%, que é um valor típico para problemas semelhantes (ALTMAN *et al.*, 2013), para definir os *thresholds* para os 9 PCs selecionados na Seção 8.2.3. O limite superior, a média e o limite inferior de cada PC é apresentado na Tabela 22. Esses *thresholds* podem ajudar a avaliações de qualidade inicial dos modelos de *features* em LPSs. No entanto, um valor diferente pode ser escolhido pelo engenheiro de domínio (especialista em LPS).

Tabela 22 – *Thresholds* definidos baseados na estratégia em que 95% dos dados estão dentro dos intervalos.

Medida	Limite Inferior (LL)	Limite Superior (UL)
PC01	15.47	4.64E+12
PC02	4.87	2.27E+12
PC03	-2.75E+12	4.99
PC04	-8.16	6.10E+12
PC05	-2.75	4.85E+11
PC06	-1.35E+12	-0.98
PC07	1.35	5.37E+12
PC08	-3.34	7.30E+12
PC09	-9.22	1.01E+12

A fim de ilustrar como um engenheiro de domínio pode utilizar esses *thresholds*, considere o seguinte exemplo. Considerando-se um modelo de *features* específico,o da LPS "Mobile Phone Software", no *dataset* MAcchiATO os valores das 15 medidas utilizadas como entrada do PCA para este modelo de *features* são apresentados na Tabela 20. Note que o valor do PC03 para este modelo é 2,70. De acordo com a Tabela 22, 2,70 está dentro dos *thresholds* definidos para o PC03.

Finalmente, é importante ressaltar que ambas as estratégias para derivar *thresholds*, "regra de três-sigma" e intervalo de tolerância, são baseadas apenas nos dados, isto é, não utiliza a opinião dos especialistas, apenas os resultados das médias para os modelos de *features*. Portanto, não é possível afirmar que os limites são os melhores valores para cada PC. Neste sentido, são necessários mais estudos a fim de acrescentar a experiência dos especialistas.

8.2.5 Validação Cruzada

Validação cruzada é uma técnica de validação para avaliar a forma como os resultados de uma análise estatística vai ser generalizada para um conjunto de dados independentes (BROWNE, 2000). Ela tenta estimar a precisão com que um método de aprendizagem irá executar na prática. Normalmente, em um problema de aprendizagem de máquina, é utilizado um *dataset* conhecido, em que a formação do modelo de aprendizagem é executado (*dataset* de treinamento) e um *dataset* de dados desconhecidos, no qual o modelo de aprendizagem é testado (*dataset* de teste). O objetivo da validação cruzada é limitar problemas com sobreposição e compreender como o modelo de aprendizagem vai ser generalizada para um *dataset* independente (isto é, um conjunto de dados desconhecidos).

Uma rodada da técnica de validação cruzada envolve particionar aleatoriamente um conjunto de dados em subconjuntos complementares, realizar a análise de um subconjunto (chamado *dataset* de treinamento), e validar a análise do outro subconjunto (chamado *dataset* de teste ou validação). A fim de reduzir a variabilidade, várias rodadas de validação cruzada são realizadas utilizando partições diferentes, e os resultados de validação são calculados sobre as rodadas. Esta estratégia tem a vantagem de que os conjuntos de treinamento e de teste são grandes (BROWNE, 2000).

Além disso, uma das principais razões para a utilização da validação cruzada, em vez de usar a validação clássica (por exemplo, dividindo o conjunto de dados em dois conjuntos de 70% para a formação e 30% para o teste) é que o erro no conjunto de treinamento na validação clássica não é um estimador útil da precisão do modelo e, portanto, o erro no *dataset* de teste não representa adequadamente a avaliação da precisão do modelo. Este fato pode ocorrer porque não há dados suficientes disponíveis ou não há uma boa distribuição e disseminação de dados para particioná-lo em conjuntos de treinamento e teste.

A fim de avaliar o modo como a abordagem proposta vai generalizar um *dataset* independente, foi utilizada a técnica de validação cruzada, com 10 rodadas. Em cada rodada, temos:

- Geração aleatória do *dataset* de treinamento e do *dataset* de teste;
- Execução do PCA, tendo como entrada as 15 medidas selecionadas na Seção 8.2.2 e usando o *dataset* de treinamento;
- Definição dos *thresholds* para os PCs selecionados (gerado na etapa de execução do PCA); e

- Análise dos *thresholds* utilizando o *dataset* de teste, verificando a quantidade de amostras que estão dentro ou fora dos limites definidos.

A Tabela 23 ilustra o percentual (em média) dos modelos de *features* fora dos limites definidos para os *thresholds*, considerando todas as 10 rodadas. É importante notar que o percentual de modelos de *features* fora dos limites definidos estão perto de 5%, como era de esperar, uma vez utilizado um intervalo de tolerância de 95%, o que indica que a abordagem proposta pode ser generalizada para um *dataset* independente.

Tabela 23 – Percentual de modelos de *features* fora dos limites definidos para os *thresholds*.

Amostras	**Percentual de modelo de *features* fora dos limites dos *thresholds***								
	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8	PC9
Rodada 1	2,8%	2,8%	4,6%	4,6%	5,5%	2,8%	4,6%	4,6%	8,3%
Rodada 2	5,5%	3,7%	4,6%	1,8%	3,7%	2,8%	0,0%	2,8%	5,5%
Rodada 3	5,5%	3,7%	0,9%	3,7%	3,7%	2,8%	3,7%	2,8%	3,7%
Rodada 4	0,0%	1,8%	0,9%	2,8%	1,8%	0,9%	0,0%	0,9%	0,9%
Rodada 5	6,4%	1,8%	2,8%	1,8%	1,8%	5,5%	1,8%	1,8%	1,8%
Rodada 6	11,9%	6,4%	9,2%	6,4%	6,4%	7,3%	8,3%	6,4%	10,1%
Rodada 7	3,7%	7,3%	4,6%	4,6%	8,3%	6,4%	2,8%	6,4%	11,9%
Rodada 8	1,8%	1,8%	2,8%	3,7%	9,2%	1,8%	1,8%	5,5%	2,8%
Rodada 9	3,7%	9,2%	2,8%	2,8%	3,7%	2,8%	4,6%	2,8%	2,8%
Rodada 10	2,8%	2,8%	3,7%	2,8%	3,7%	3,7%	0,9%	4,6%	5,5%
Média	**4%**	**4%**	**4%**	**3%**	**5%**	**4%**	**3%**	**4%**	**5%**

8.3 Discussões

Nesta Seção, são discutidos os resultados do estudo de caso exploratório comparando com trabalhos relacionados e são apresentadas as implicações para a pesquisa e prática.

8.3.1 Relação com a Literatura Existente

Dois outros trabalhos relacionados, apresentados em Bagheri E. e Gasevic (2011) e Berger T. e Guo (2014), investigaram a correlação entre as medidas de qualidade em modelos de *features* de LPSs.

Bagheri E. e Gasevic (2011) aplicaram apenas 10 medidas de qualidade em um pequeno conjunto de 14 modelos de *features* extraídos do repositório S.P.L.O.T., no entanto, muitos desses modelos de *features* não são de software. Semelhante ao estudo realizado neste trabalho, Bagheri E. e Gasevic (2011) utilizaram o coeficiente de correlação de *Spearman* (com um nível de significância de 0,05) para investigar as correlações entre as medidas de qualidade. Alguns resultados apresentados em Bagheri E. e Gasevic (2011) são muito próximos

aos resultados deste estudo, tais como uma correlação muito forte entre NF e NLeaf, uma forte correlação entre CoC e NLeaf, uma fraca correlação entre NF e NTop, e uma fraca correlação entre NLeaf e NTop. No entanto, alguns resultados apresentados em Bagheri E. e Gasevic (2011) não foram confirmados neste trabalho, por exemplo, a forte correlação entre: i) CoC e CTC; ii) CoC e RoV; e iii) NVC e NTop. Isto pode ser devido à grande diferença entre o tamanho dos *datasets* utilizados. Este trabalho avalia uma quantidade maior de medidas (32 medidas) e executa a análise de correlação usando 218 modelos de *features* enquanto o trabalho de Bagheri E. e Gasevic (2011) utilizam apenas 10 medidas e 14 modelos de *features*.

Berger T. e Guo (2014) realizaram uma análise de correlação usando 12 medidas de qualidade para os modelos de *features* de LPS (9 medidas estruturais e 3 medidas de dependência) e um pequeno conjunto de 8 modelos de *features*. Semelhante a este trabalho e ao de Bagheri E. e Gasevic (2011), Berger T. e Guo (2014) utilizaram o coeficiente de correlação de Spearman para indicar a correlação estatística entre duas medidas de qualidade. Alguns resultados apresentados em Berger T. e Guo (2014) são muito próximos aos achados neste trabalho e os resultados de Bagheri E. e Gasevic (2011). Os autores detectaram uma relação muito forte entre NF e NLeaf, por exemplo, e uma forte correlação entre RCon e RDen. No entanto, diferente deste trabalho e do de Bagheri E. e Gasevic (2011), eles identificaram uma correlação muito forte entre NF e NTop, além de uma correlação muito forte entre NLeaf e NTop. Isto pode ter acontecido devido ao pequeno tamanho do *dataset* utilizado.

Já Abilio *et al.* (2015) apresentaram uma estratégia para definir *thresholds* para 13 medidas que foram utilizadas para detecção de 3 tipos diferentes de *code smells* em LPSs. Este estudo envolveu 26 participantes de duas instituições diferentes. Os participantes foram divididos em três grupos. Cada grupo trabalhou com um tipo de *code smell* e não possuía acesso ao código fonte. Cada participante recebeu um documento com: i) a descrição da LPS, medidas e *code smell* do código-alvo; ii) *thresholds*; e iii) 60 componentes e suas respectivas medidas. Os participantes começaram indicando os *thresholds* e as medidas que eles usaram para inspecionar os componentes. Assim, os participantes selecionaram os *thresholds* para cada *code smell*. Os autores utilizaram uma medida de concodância entre os avaliadores *Cohen's kappa* (KRAEMER, 1982) para avaliar a concordância entre os participantes a fim de definir *thresholds* para as medidas. Diferente de Abilio *et al.* (2015), neste trabalho foi utilizada uma abordagem centrada em dados para definir *thresholds* para as medidas, que não precisam de inspeções manuais.

Por último, Vale *et al.* (2015) e Vale G. e Figueiredo (2015) propuseram um método

para derivar *thresholds* para medidas de código que foram utilizadas para a detecção de 2 tipos diferentes de *code smells* em LPSs. A solução proposta foi avaliada utilizando uma *benchmark* composto por 33 LPSs. O método para derivação *thresholds* apresentados em Vale G. e Figueiredo (2015) é muito semelhante à estratégia baseada em intervalo de tolerância usada no estudo apresentado por este trabalho. Em Vale *et al.* (2015), Vale G. e Figueiredo (2015), os *thresholds* são obtidos por escolher o percentual dos valores da medida que se deseja representar. Assim, os *thresholds* são derivados escolhendo 3%, 15%, 90% e 95% do valor da medida, resultando em 4 *thresholds* (T1, T2, T3 e T4) para cada medida. Isso permite identificar o valor de medida que deve ser fixada em longo prazo, médio prazo e curto prazo. Além disso, esses percentuais são usados em perfis de qualidade para caracterizar o valor das medidas de acordo com cinco categorias: valores muito baixos (entre 0-3%), valores baixos (3-15%), valores moderados (15-90%), valores elevados (90-95%) e valores muito altos (95-100%). Neste estudo de caso, as medidas foram definidas para o modelo de features de LPSs e o método utilizado para definição dos *thresholds* foi o de intervalo de tolerância.

8.3.2 Implicações para os Pesquisadores e Engenheiros de Domínio

Os resultados do estudo mostraram que as correlações entre as medidas de qualidade existem. Assim, nem todas as medidas são necessárias para revelar as características de qualidade de um modelo de *features*. Além disso, as constatações obtidas neste trabalho indicam que a técnica de PCA pode ser usada para construção de novas medidas agrupadas mais representativas do que as medidas individuais. Finalmente, se a inspeção manual nos modelos de *features* para definir *thresholds* não é possível, estratégias centradas em dados podem ser aplicadas. Entretanto, se os *thresholds* não estão devidamente definidos, é difícil realmente saber se um determinado valor da medida indica um problema potencial no modelo de *features*.

Uma vez que tenha sido estabelecido que as correlações entre as medidas de qualidade existem, acredita-se que as abordagens de seleção de característica podem ser usadas para selecionar um subconjunto de medidas relevantes de qualidade (também conhecidas em aprendizagem de máquina e estatísticas como característica, subcaracterística ou variável), a fim de construir um modelo (por exemplo, um modelo de previsão). As abordagens de seleção de característica são utilizados por três razões: simplificação de modelos, menor tempo de treinamento e redução à sobreposição. A premissa principal quando se utiliza uma abordagem de seleção de características é que os dados contém muitas características que são redundantes

ou irrelevantes, e assim pode ser removido sem muita perda de informações, que foi exatamente o que aconteceu no estudo. Além disso, acredita-se que a teoria de redes, tais como a lei de potência, poderia ajudar a identificar possíveis medidas que têm muitas dependências ou das quais muitas medidas dependem.

O processo de projeto e execução do trabalho de investigação necessário para este estudo resultou em importantes reflexões sobre muitos aspectos da realização da investigação empírica sobre o campo de LPSs.

O primeiro desafio enfrentado foi obter um conjunto de modelos de *features* de LPSs bem estabelecidas e acessíveis ao público. Foi encontrado apenas um repositório público, o S.P.L.O.T. No entanto, a maioria dos modelos de *features* compartilhados foram concebidos para fins acadêmicos e de investigação. É fundamental reunir os modelos de *features* industriais e reais dentro desses repositórios para fortalecer estudos empíricos futuros e avaliações. Para trabalhos futuros, o ideal seria selecionar modelos de *features* em escala real e industriais.

O segundo desafio foi no que diz respeito à complexidade das medidas estruturais. Os resultados apoiam a ideia de que medidas simples, que são correlacionadas com medidas complexas, são bastante úteis para indicar atributos de qualidade externos.

O terceiro desafio está relacionado com a ausência do grau de qualidade nos modelos de *features*, em outras palavras, se cada modelo de *features* tivesse, pelo menos, uma indicação de qualidade como boa ou má, seria possível usar algoritmos mais sofisticados de aprendizagem de máquina para definir os *thresholds* das medidas ou classificar novos modelos de *features*.

Por último, de acordo com os resultados deste trabalho, não é possível provar que a melhoria da qualidade dos modelos de *features* implica em uma melhor reutilização de software. Para isso, é necessário monitorar a qualidade e o nível de reutilização de algumas LPSs reais. Além disso, este trabalho não forneceu diretrizes para melhoria da qualidade dos modelos de *features*. Estes aspectos serão investigados em trabalhos futuros.

8.4 Ameaças à Validade

Mesmo com o projeto cuidadoso do estudo de caso, esta pesquisa pode ser afetada por diferentes fatores que podem invalidar suas principais conclusões. Essas ameaças e as ações executadas, a fim de mitigar o impacto desses fatores sobre os resultados da investigação são descritas de acordo com Wohlin *et al.* (2012), como se segue.

Validade Interna. Para aumentar a validade interna, foram coletadas todas as medi-

das automaticamente utilizando a ferramenta DyMMer. Além disso, foi realizada a análise de correlação seguindo um processo de cálculo padrão do coeficiente de correlação de *Spearman* e foram escolhidas apenas os resultados com significância estatística (p <0,05) para análise. No entanto, não se pode garantir que os resultados da análise do experimento deste estudo dependem de definições específicas das medidas de qualidade no catálogo COfFEE. O repositório S.P.L.O.T. possui muitos modelos de *features* que foram projetados para fins acadêmicos e de investigação (exemplos fictícios). Este fator pode influenciar nos *thresholds* definidos. Além disso, as medidas no catálogo COfFEE possuem diferentes distribuições probabilísticas. Para mitigar essas ameaças, foi utilizado um grande número de modelos de *features* (218) e foram escolhidas estratégias não-paramétricas (coeficiente de correlação de *Spearman*, PCA e intervalo de tolerância).

Validade de Construção. A validade de construção está preocupada com a relação entre a teoria e a observação. Neste contexto, a principal preocupação do estudo é a redução do número de medidas de qualidade executadas na etapa de análise de correlação, a partir das 32 medidas iniciais no catálogo COfFEE para 15 medidas. Este conjunto resultante de 15 medidas de qualidade foi usado como entrada na etapa PCA. Este processo de seleção foi realizado manualmente e com base nas subcaracterísticas qualidade. O resultado de um conjunto diferente de medidas resultou em diferentes componentes principais. Como trabalho futuro, pretende-se utilizar um algoritmo de seleção de característica para produzir um pequeno conjunto de medidas para ser usado como entrada na etapa de PCA.

Validade Externa. A validade externa é a extensão que os resultados obtidos de um estudo de caso podem ser generalizados para outros cenários de investigação relevantes. Os resultados do estudo de caso válido externamente podem ser generalizados e aplicados com segurança para a prática de engenharia de software e serem recomendados como padrões para avaliação da qualidade do modelo de *features*. Para mitigar esta ameaça, foi utilizado um grande conjunto de modelos de *features*, incluindo modelos de *features* com diferentes tamanhos e complexidades. Além disso, todas as medidas de qualidade utilizadas neste trabalho foram extraídas a partir da literatura existente. No entanto, os resultados dos experimentos deste estudo não são automaticamente transferíveis para todos os outros *datasets* de modelos de *features*.

Validade de Conclusão. A validade de conclusão é a extensão em que as conclusões sobre a presença de uma relação estatisticamente significativa entre os tratamentos e os resultados, são válidos. Para mitigar as ameaças à validade da conclusão e aumentar a confiabilidade deste

estudo de caso exploratório, foi utilizada a técnica de validação cruzada, com 10 rodadas. Esta validação é importante para avaliar como a abordagem proposta vai generalizar para um *dataset* independente.

8.5 Conclusões

Este capítulo apresentou um estudo de caso exploratório, utilizando o *dataset* MAcchiATO. Este estudo de caso exploratório teve como objetivo investigar como as medidas de qualidade podem ser aplicadas para a avaliação da manutenibilidade dos modelos de *features* em LPSs. Foram exploradas três técnicas de análise de dados diferentes, a fim de identificar relações entre as medidas, agrupar medidas de qualidade e definir *thresholds* para as medidas.

Inicialmente, foi aplicado o coeficiente de correlação de *Spearman* para identificar correlações entre as medidas presentes no *dataset* MAcchiATO. A análise mostrou que existem fortes correlações entre essas medidas. Desta forma, nem todas as 32 medidas são necessárias para revelar a manutenibilidade de um modelo de *features*, mais precisamente, é possível utilizar somente 15 medidas. Em seguida, a técnica PCA (Análise de Componentes Principais) foi utilizada com a finalidade de agrupar as 32 medidas avaliadas. Como resultado foram produzidas 9 novas medidas agrupadas. Essas medidas agrupadas têm praticamente a mesma cobertura das 32 medidas iniciais. Posteriormente, duas técnicas diferentes, "regra de três-sigma" e intervalo de tolerância, foram utilizadas com o objetivo de definir *thresholds* estatísticos para as 9 medidas agregadas, produzidas anteriormente.

Os resultados deste estudo sugerem que medidas de qualidade podem ser efetivamente utilizadas para apoiar a avaliação da qualidade do modelos de *features* em LPSs.

9 AGREGAÇÃO DE MEDIDAS DE MANUTENIBILIDADE UTILIZANDO LÓGICA *FUZZY*

Este capítulo apresenta um estudo utilizando lógica *fuzzy* com a finalidade de agregar as medidas propostas para avaliar a manutenibilidade do modelo de *features*. A Seção 9.1 apresenta uma motivação para agregação de medidas e avaliação da qualidade de modelo de *features* de LPSDs. A Seção 9.2 apresenta o processo de execução da lógica *fuzzy*. A Seção 9.3 apresenta a abordagem, baseada em lógica *fuzzy*, que foi utilizada para agregar as medidas. A Seção 9.4 apresenta a aplicação das 4 novas medidas agregadas nos 30 modelos de *features* de LPSDs do *dataset* ESPREssO. A Seção 9.5 discute as ameaças à validade o estudo. Finalmente, a Seção 9.6 conclui este capítulo.

9.1 Motivação

Conforme citado anteriormente no Capítulo 2, um dos aspectos mais importantes de uma LPSD é o modelo de *features*. Geralmente, em LPSDs, duas estratégias são utilizadas para modelar a variabilidade dinâmica. Na primeira estratégia, o projetista modela as *features* de contexto separadamente das *features* que não são de contexto, em dois modelos modelos de *features* relacionados, mas distintos. Na segunda estratégia, tanto as *features* de contexto quanto as *features* que não são de contexto são modeladas em um único modelo de *features* (CAPILLA *et al.*, 2014b).

O catálogo COfFEE disponibiliza 40 medidas para avaliar a manutenibilidade do modelo de *features*. Entretanto, sabe-se que, muitas vezes, as medidas tomadas individualmente não são suficientes para caracterizar a manutenibilidade do modelo de *features*. Para lidar com este desafio, uma possível solução consiste em utilizar alguma técnica que possibilite a agregação destas medidas. Uma técnica bastante utilizada para agregar medidas é denominada agregação *fuzzy*. Esta técnica baseia-se na utilização da teoria de lógica *fuzzy* (KUNCHEVA; KRISHNAPURAM, 1996; YADAV; YADAV, 2015).

De Acordo com Pizzi (2013), a agregação *fuzzy* tem sido usada em vários problemas quantitativos da engenharia de software, tais como: automação da matriz de rastreabilidade de requisitos de software (THOMMAZO *et al.*, 2013), estimativa de manutenibilidade de software (PRATAP *et al.*, 2014), priorização de casos de testes baseados em risco (HETTIARACHCHI *et al.*, 2016), seleção de casos de teste de regressão (XU *et al.*, 2014) e predição de defeitos de software (YADAV; YADAV, 2015).

Neste trabalho, inicialmente, foram selecionadas 13 medidas do catálogo COfFEE. Essas medidas foram consideradas as mais relevantes para avaliar a manutenibilidade do modelo de *features* e compõem o *dataset* ESPREssO. Em seguida, a teoria de lógica *fuzzy* foi aplicada com a finalidade de agregar as 13 medidas previamente selecionadas. Como resultado foram produzidas 4 novas medidas agregadas. As novas medidas agregadas podem ser aplicadas para avaliar diferentes aspectos relacionados aos modelos de *features* em LPSDs, tais como: tamanho, estabilidade, flexibilidade e dinamicidade. Com o objetivo de avaliar o uso das novas medidas agregadas, estas foram aplicadas aos 30 modelos de *features* de LPSDs presentes no *dataset* ESPREssO. As descobertas deste estudo sugerem que a agregação de medidas pode ser efetivamente utilizada para apoiar a avaliação da manutenibilidade de modelos de *features* em LPSDs.

9.2 Lógica *Fuzzy*

Na teoria clássica os conjuntos são denominados *crisp* e um determinado elemento do universo em discurso (domínio) pertence ou não pertence ao referido conjunto. Ainda na teoria dos conjuntos *fuzzy* existe um grau de pertinência de cada elemento no intervalo de [0,1] a um determinado conjunto (e.g., conjuntos das pessoas altas) (BARCELLOS, 2009). Para cada conjunto, então, é criada uma função de pertinência, que indica o grau de pertinência de seus elementos a este conjunto. O grau de pertinência permite valores imprecisos. Outro conceito importante da lógica *fuzzy* é a variável linguística que consiste em uma variável cujos valores são nomes de conjuntos *fuzzy* (e.g., a variável linguística temperatura).

A Figura 25 ilustra uma arquitetura típica de um sistema especialista *fuzzy*. O sistema especialista de lógica *fuzzy* é dividido em quatro componentes (HETTIARACHCHI *et al.*, 2016):

- ***Fuzzificação*:** É responsável por traduzir a informação de entrada para o domínio *fuzzy*, ou seja, as informações numéricas para variáveis linguísticas. Este processo transforma a entrada *crisp* em uma entrada para as funções de pertinência.
- **Engenharia de Inferência *Fuzzy***: O processo de inferência usa a entrada das funções de pertinência para determinar a saída das funções de pertinência usando regras formuladas a partir da base de conhecimento. O processo de composição agrega todas as saídas *fuzzy* em um conjunto *fuzzy* único.
- **Base de Conhecimento:** A base de conhecimento apresentada na Figura 25 contém o conjunto de regras *fuzzy* selecionadas. Em um sistema especialista *fuzzy*, as regras *fuzzy*

executam um papel vital uma vez que esta são formuladas com base no conhecimento dos especialistas sobre o domínio de interesse. As regras *fuzzy* definem a região *fuzzy* do espaço de entrada e, consequentemente, definem a região *fuzzy* do espaço de saída. As regras *fuzzy* podem suportar não somente múltiplas variáveis de entradas, mas também múltiplas variáveis de saída (HETTIARACHCHI *et al.*, 2016).

- ***Defuzzificação***: A *defuzzificação* é a conversão de um conjunto *fuzzy* em um valor ou um vetor de valores. O processo de *defuzzificação* calcula uma saída usando o conjunto *fuzzy* produzido pelo processo de composição.

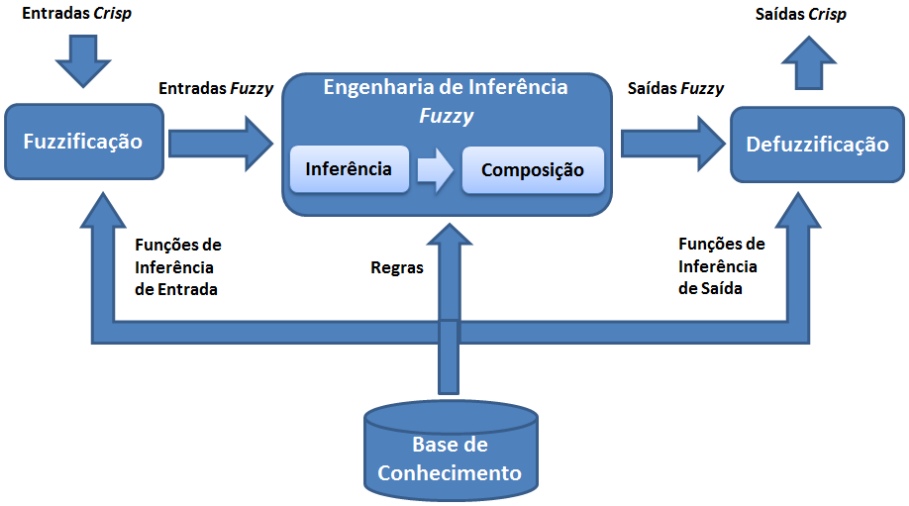


Figura 25 – Estrutura do sistema especialista *fuzzy* (HETTIARACHCHI *et al.*, 2016).

9.3 Agregando Medidas

A ideia principal do processo de agregação consiste em obter um nível de consenso entre a informação disponível, calculando um valor final. Se estes dados são extraídos por especialistas, então eles terão uma taxa de aceitação ou rejeição desses dados, que é o grau para qual os especialistas concordam em suas estimativas, assim como pode facilitar o desenvolvimento de classificações das avaliações (KUNCHEVA; KRISHNAPURAM, 1996).

Neste estudo, a lógica *fuzzy* foi utilizada com a finalidade de agregar as 13 medidas pertencentes ao *dataset* ESPREssO. Para isso, utilizou-se a metodologia proposta por Yadav e Yadav (2015) para agregação de medidas baseadas na lógica *fuzzy*. Esta metodologia se adéqua bem a problemas de engenharia de software. A seguir, são apresentadas as etapas que compõem a metodologia utilizada:

1. Seleção de medidas;

2. Definição da função de pertinência de entrada e saída das medidas;
3. Projeção das regras *fuzzy*; e
4. Execução da inferência de *fuzzificação* e *defuzzificação*.

Nas próximas seções essas etapas são discutidas em detalhes.

9.3.1 Seleção de Medidas

Neste estudo, buscou-se definir medidas agregadas para avaliar os seguintes aspectos: tamanho, estabilidade, flexibilidade e dinamicidade. Logo, deseja-se criar uma medida agregada para cada um desses aspectos. É importante destacar que as medidas agregadas para tamanho, estabilidade e flexibilidade podem ser aplicadas tanto para o modelo de *features* de LPSs quanto de LPDSs. Já a medida para dinamicidade é específica para modelos de *features* de LPSDs. Para cada uma dessas 4 medidas agregadas foram definidos o intervalo *fuzzy* e os estados linguísticos.

O tamanho tem um grande impacto na manutenibilidade do modelo de *features*. O tamanho de um modelo de *features* pode ser medido por meio da quantidade de *features*, além da largura e profundidade do modelo. Um modelo de *features* com um valor elevado para a medida agregada tamanho pode apresentar problemas em seu processo de configuração. A configuração do modelo de *features* é um processo que envolve a seleção das *features* desejadas para um produto final (THÜM *et al.*, 2009). Adicionalmente, um valor elevado para esta medida indica uma maior complexidade.

Tabela 24 – Medidas de tamanho do modelo de *features*.

Medidas de tamanho do modelo de *features*		**Intervalo *Fuzzy***	**Termos Linguísticos**
Medidas de Entrada	*DTMax*	{0, 50}	{L,M, H}
	NLeaf	{0, 100}	{L,M, H}
	NF	{0, 100}	{L,M, H}
Medida de Saída	*Índice de Tamanho do modelo de features (SIFM)*	{0, 100}	{VL, L, M, H, VH}

VL-Muito Baixo, L-Baixo, M-Médio, H-Alto, VH-Muito Alto.

Durante a realização desta pesquisa de doutorado, foram identificadas diversas medidas relacionadas ao tamanho do modelo de *features*. Contudo, nenhuma medida agregada voltada para este aspecto foi encontrada. Assim, para compor uma nova medida agregada relacionada ao tamanho, denominada "Índice do Tamanho do modelo de *features* (SIFM)", foram selecionadas três medidas, as quais são ilustradas na Tabela 24.

Estabilidade é a habilidade do software de minimizar efeitos inesperados provocados

por mudanças (ALFÉREZ *et al.*, 2014b). Para compor uma nova medida agregada relacionada à estabilidade, denominada "Índice de Estabilidade do modelo de *features*" (STIFM), foram selecionadas três medidas, as quais são ilustradas na Tabela 25.

Tabela 25 – Medidas de estabilidade do modelo de *features*.

Medidas de estabilidade do modelo de *features*		**Intervalo *Fuzzy***	**Termos Linguísticos**
Medidas de Entrada	*RDen*	{0, 10}	{L,M, H}
	RoV	{0, 10}	{L,M, H}
	CogC	{0, 10}	{L,M, H}
Medida de Saída	*Índice de Estabilidade do modelo de features (STIFM)*	{0, 10}	{VL, L, M, H, VH}

VL-Muito Baixo, L-Baixo, M-Médio, H-Alto, VH-Muito Alto.

A flexibilidade está relacionada à habilidade do modelo de *features* de responder a potenciais mudanças internas ou externas afetando sua entrega de valor, em tempo hábil e custo-efetividade (BAGHERI E. E GASEVIC, 2011; DUAN *et al.*, 2013; ZHANG *et al.*, 2014). Para compor uma nova medida agregada relacionada à flexibilidade, denominada "Índice de Flexibilidade de modelo de *features*" (FIFM), foram selecionadas três medidas, as quais são ilustradas na Tabela 26.

Tabela 26 – Medidas de flexibilidade do modelo de *features*.

Medidas de flexibilidade do modelo de *features*.		**Intervalo *Fuzzy***	**Termos Linguísticos**
Medidas de Entrada	*NM*	{0, 100}	{L,M, H}
	FoC	{0, 1}	{L,M, H}
	NF	{0, 100}	{L,M, H}
Medida de Saída	*Índice de Flexibilidade do modelo de features (FIFM)*	{0, 100}	{VL, L, M, H, VH}

VL-Muito Baixo, L-Baixo, M-Médio, H-Alto, VH-Muito Alto.

A dinamicidade está relacionada ao nível de adaptação do modelo de *features* em relação às restrições e adaptações de contexto presentes no modelo de *features*. Para compor uma nova medida agregada relacionada à dinamicidade, denominada "'Índice de Dinamicidade do modelo de *features*" (DIFM), foram selecionadas quatro medidas, as quais são ilustradas na Tabela 27.

9.3.2 Definição da Função de Pertinência

Uma função de pertinência pode ser gerada com a ajuda de um especialista do domínio ou a partir de dados reais. Neste estudo, as funções de pertinência foram definidas utilizando a primeira abordagem, ou seja, com o suporte de especialistas. Desenvolver uma

Tabela 27 – Medidas de dinamicidade do modelo de *features*.

Medidas de dinamicidade do modelo de *features*		**Intervalo *Fuzzy***	**Termos Linguísticos**
Medidas de Entrada	*CF*	{0, 50}	{L,M, H}
	CFC	{0, 50}	{L,M, H}
	AFCA	{0, 50}	{L,M, H}
	DFCA	{0, 50}	{L,M, H}
Medida de Saída	*Índice de Dinamicidade do modelo de features (DIFM)*	{0, 50}	{VL, L, M, H, VH}

VL-Muito Baixo, L-Baixo, M-Médio, H-Alto, VH-Muito Alto.

função de pertinência com ajuda do conhecimento de especialistas do domínio é um dos passos básicos quando se deseja projetar uma solução baseada na teoria dos conjuntos *fuzzy* para um determinado problema (YADAV; YADAV, 2015).

Uma função de pertinência pode ter uma variedade de formas, tais como: poligonal, trapezoidal e triangular (ROSS, 2009). Entretanto, as formas triangulares e trapezoidais proporcionam uma representação conveniente do conhecimento de especialista do domínio, além de simplificar o processo de computação (YADAV; YADAV, 2015). Nesta pesquisa, foram utilizadas as funções de pertinência triangular e trapezoidal. O trabalho proposto por Hettiarachchi *et al.* (2016) indica que a função de pertinência triangular é fácil e simples de se aplicar comparada às demais funções de pertinência.

Na etapa de *fuzzificação*, os valores das variáveis de entrada são usados para determinar o nível para o qual estes valores se encaixam em cada função de pertinência usada pelas regras *fuzzy*. Neste estudo, funções de pertinência para todas as medidas de entrada e saída foram definidas com a ajuda dos especialistas do domínio.

Foram selecionados 4 especialistas em modelagem de variabilidade de LPSs e LPSDs para definir as regras e funções de pertinência da lógica *fuzzy*. A revisão por pares foi conduzida por dois doutores e dois estudantes de doutorado. Todos os especialistas possuem mais de 5 anos de experiência em Engenharia de Domínio de LPSs, e apenas um dos especialistas não possui experiência em LPSDs. O conhecimento em Engenharia de Domínio de LPSs e LPSDs é considerado alto para os três especialistas, e apenas um dos especialistas possui baixo conhecimento em LPSDs. Todos os especialistas têm trabalhado com 3 ou mais projetos de LPSs e 3 especialistas tem trabalhado com até dois projetos de LPSDs.

O processo de decisão dos especialistas consistiu das seguintes fases: (i) os especialistas preencheram um formulário de revisão com o perfil dos especialistas (ver Apêndice C) e informações sugeridas às funções de pertinência e regras da lógica *fuzzy* para cada medida agregada; (ii) os especialistas julgaram as informações de cada medida usando como parâmetro

de medições coletadas no *dataset* ESPREssO e suas próprias experiências; (iii) as informações em contradição foram julgadas de acordo com quatro categorias (omissão, fato incorreto, ambiguidade e inconsistência); e (iv) em concordância com as modificações sugeridas pelos especialistas chegou-se a uma consolidação da saída final das funções pertinência e regras de da lógica *fuzzy* para cada medida agregada.

A Tabela 28 ilustra os valores de entrada e saída para cada função de pertinência.

Tabela 28 – Parâmetros das medidas para as funções de pertinência.

Medidas de Tamanho Parâmetros		**Medidas de Estabilidade Parâmetros**		**Medidas de Flexibilidade Parâmetros**		**Medidas de Dinamicidade Parâmetros**	
Medidas	**Parâmetros**	**Medidas**	**Parâmetros**	**Medidas**	**Parâmetros**	**Medidas**	**Parâmetros**
DTMax	Baixo <3	RDen	Baixo <= 1	NM	Baixo <4	CF	Baixo <6
	3 <= Médio <5		1 <Médio <1.6		4 <= Médio <11		6 <= Médio <13
	Alto >= 5		Alto >= 1.6		Alto >= 11		Alto >= 13
Nleaf	Baixo <10	RoV	Baixo <2	FoC	Baixo <0.12	CFC	Baixo <2
	10 <= Médio <28		2 <= Médio <4		0.12 <= Médio <0.36		2 <= Médio <5
	Alto >= 28		Alto >= 4		Alto >= 0.36		Alto >= 5
NF	Baixo <15	CogC	Baixo <2	NF	Baixo <15	AFCA	Baixo <7
	15 <= Médio <40		2 <= Médio <7		15 <= Médio <40		7 <= Médio <16
	Alto >= 40		Alto >= 7		Alto >= 40		Alto >= 16
						DFCA	Baixo <5
							5 <= Médio <11
							Alto >= 11
SIFM	Muito Baixo <10	***STIFM***	Muito Baixo <2	***FIFM***	Muito Baixo <10	***DIFM***	Muito Baixo <=1
	10 <= Baixo <20		2 <= Baixo <3		10 <= Baixo <25		1 <Baixo <3
	20 <= Médio <40		3 <= Médio <4		25 <= Médio <40		3 <= Médio <4
	40 <= Alto <51		4 <= Alto <5		40 <= Alto <55		4 <= Alto <5
	Muito Alto >= 51		Muito Alto >= 5		Muito Alto >= 55		Muito Alto >= 5

Um exemplo de função de pertinência para a medida relacionada à dinamicidade do

modelo de *features* é apresentado na Figura 26. A Figura representa os intervalos da função de pertinência para as 4 medidas de entrada CF, CFC, AFCA e DFCA, e a medida de saída DIFM.

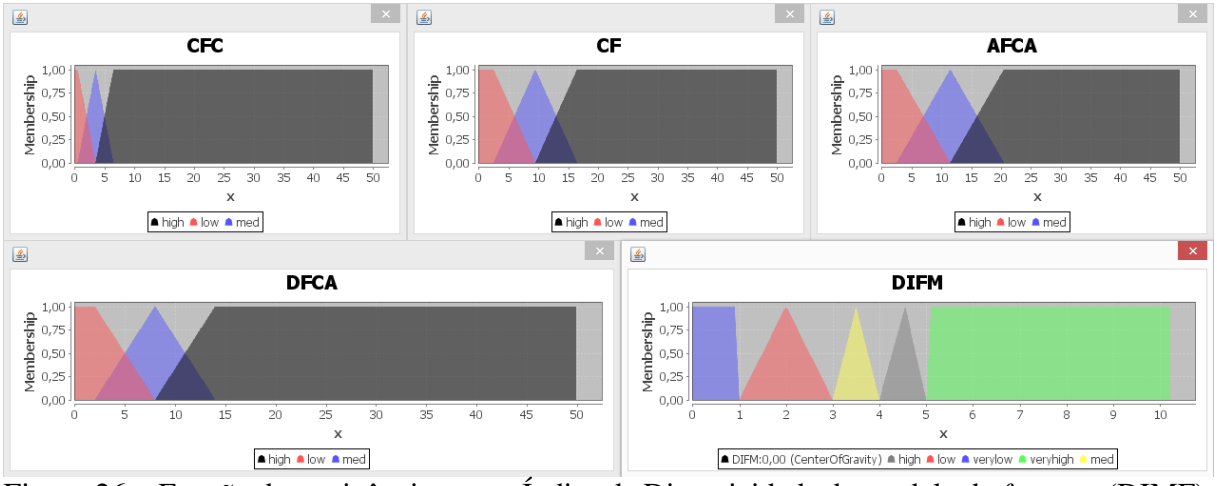


Figura 26 – Função de pertinência para o Índice de Dinamicidade do modelo de *features* (DIMF).

9.3.3 Projeto das Regras Fuzzy

Segundo (YADAV; YADAV, 2015), as regras *fuzzy* são definidas na forma de uma declaração condicional SE-ENTÃO. A parte SE da regra é conhecida previamente, e a parte ENTÃO posteriormente. A base da regra *fuzzy* pode ser projetada a partir de diferentes fontes, tais como domínio, especialistas, análise de dados históricos, e engenharia do conhecimento da literatura existente. Em geral, as regras *fuzzy* são projetadas com a ajuda dos especialistas do domínio.

Regras *Fuzzy* de Tamanho. Se os valores para DTMax, NLeaf e NF são altos, então o valor de SIMF também é alto. Para esta medida agregada, foram selecionadas três medidas de entrada do tamanho do modelo de *features*. Cada medida de entrada possui três estados linguísticos, ou seja, baixo (L), médio (M) e alto (H). Portanto, o número de total de regras é 12. Uma amostra dessas regras *fuzzy* são apresentadas na Tabela 29.

Tabela 29 – Regras *fuzzy* de tamanho.

Regras	Regras *Fuzzy*
1	SE DTMax É L E NLeaf É L E NF É L ENTÃO SIFM É VL
2	SE DTMax É M E NLeaf É L E NF É L ENTÃO SIFM É L
...	...
10	SE DTMax É M E NLeaf É H E NF É M ENTÃO SIFM É H
11	SE DTMax É M E NLeaf É H E NF É H ENTÃO SIFM É H
12	SE DTMax É H E NLeaf É H E NF É H ENTÃO SIFM É VH

Regras *Fuzzy* de Estabilidade. Se os valores de RDen, RoV e CogC são altos,

então o valor de STIFM é baixo. Para esta medida agregada, foram selecionadas três entradas para as medidas de estabilidade do modelo de *features*. Cada medida de entrada possui três estados linguísticos baixo (L), médio (M) e alto (H), e cada medida de saída tem cinco estados linguísticos, ou seja, muito baixo (VL), baixo (L), médio (M), alto (H), muito alto (VH). Portanto, o número total de regras é 27. Uma amostra dessas regras *fuzzy* de estabilidade é apresentada na Tabela 30.

Tabela 30 – Regras *fuzzy* de estabilidade.

Regras	**Regras *Fuzzy***
1	SE RDEn É L E RoV É L E CogC É L ENTÃO STIFM É VL
2	SE RDEn É L E RoV É L E CogC É M ENTÃO STIFM É H
...	...
25	SE RDEn É H E RoV É H E CogC É L ENTÃO STIFM É L
26	SE RDEn É H E RoV É H E CogC É M ENTÃO STIFM É VL
27	SE RDEn É H E RoV É H E CogC É H ENTÃO STIFM É VL

Regras *Fuzzy* de Flexibilidade. Se os valores de FoC e NF são altos e o valor de NM é baixo, então o valor de FIFM é alto. Para esta medida agregada, foram selecionadas três medidas de entrada de flexibilidade do modelo de *features*. Cada medida de entrada possui três estados linguísticos, ou seja, baixo (L), médio (M) e alto (H) e cada medida de saída possui cinco estados linguísticos, ou seja, muito baixo (VL), baixo (L), médio (M), alto (H), muito alto (VH). Portanto, o número total de regras é 22. Uma amostra dessas regras *fuzzy* é apresentada na Tabela 31.

Tabela 31 – Regras *fuzzy* de flexibilidade.

Regras	**Regras *Fuzzy***
1	SE NM É L E FoC É L E NF É L ENTÃO FIFM É L
2	SE NM É L E FoC É M E NF É L ENTÃO FIFM É M
...	...
20	SE NM É H E FoC É H E NF É L ENTÃO FIFM É H
21	SE NM É H E FoC É H E NF É M ENTÃO FIFM É H
22	SE NM É H E FoC É H E NF É H ENTÃO FIFM É H

Regras *Fuzzy* de Dinamicidade. Se os valores de CF, CFC e AFCA são altos e o valor de DFCA é baixo, então o valor de DIFM é alto. Para esta medida agregada, foram selecionadas três medidas de entrada da dinamicidade do modelo de *features*. Cada medida de entrada possui três estados linguísticos, ou seja, baixo (L), médio (M) e alto (H) e cada medida de saída possui cinco estados linguísticos, ou seja, muito baixo (VL), baixo (L), médio (M), alto (H), muito alto (VH). Portanto, o número total de regras é 25. Uma amostra dessas regras *fuzzy* é apresentada na Tabela 32.

Tabela 32 – Regras *fuzzy* de dinamicidade.

Regras	Regras *Fuzzy*
1	SE CF É L E CFC É L E AFCA É L E DFCA É L ENTÃO DIFM É VL
2	SE CF É M E CFC É L E AFCA É L E DFCA É L ENTÃO DIFM É L
...	...
23	SE CF É H E CFC É H E AFCA É M E DFCA É H ENTÃO DIFM É H
24	SE CF É H E CFC É H E AFCA É H E DFCA É M ENTÃO DIFM É VH
25	SE CF É H E CFC É H E AFCA É H E DFCA É H ENTÃO DIFM É VH

9.3.4 Execução da Inferência de Fuzzificação e Defuzzificação

Para realizar a *fuzzificação* e *defuzzificação*, foi utilizada a ferramenta *jFuzzyLogic*, que é uma biblioteca *open source* para sistemas *fuzzy* a qual permite projetar controladores de Lógica *Fuzzy* (CINGOLANI; ALCALA-FDEZ, 2012). A biblioteca foi utilizada porque está disponível como código aberto[1].

No processo de inferência, as entradas *fuzzy* (ou seja, o nível de funções de pertinência apropriadas para os valores de entrada das variáveis) são aplicadas em cada regra previamente para determinar o nível de veracidade de cada regra. O nível de veracidade é aplicado para os valores posteriores de cada regra, então a variável de saída obtém uma função de pertinência apropriada. As funções de pertinência das variáveis de saída definidas neste estudo são apresentadas na Tabela 28 (HETTIARACHCHI *et al.*, 2016).

A etapa de composição é responsável por combinar todas as funções de pertinência obtidas a partir de cada variável de saída e por formar uma única função de pertinência para cada variável de saída (HETTIARACHCHI *et al.*, 2016).

O processo de *defuzzificação* produz saídas *crisp* para cada variável de saída. Portanto, neste passo, a função resultante de pertinência do passo anterior é defuzzificada em um único número (HETTIARACHCHI *et al.*, 2016). A etapa de *defuzzificação* é apresentada na Seção 9.4.

A implementação das quatro medidas agregadas foi realizada utilizando-se a ferramenta *jFuzzyLogic*. O código desta implementação está disponível de forma *online* e pode ser utilizado livremente pela comunidade acadêmica[2].

[1] jfuzzylogic.sourceforge.net
[2] https://goo.gl/ONfTL3

9.4 Avaliação do Uso das Medidas Agregadas

Com a finalidade de validar a utilização das quatro medidas agregadas, estas medidas foram aplicadas nos 30 modelos de *features* de LPSDs presentes no *dataset* ESPREssO (ver Seção 6.5). Os resultados desta aplicação são ilustrados na Tabela 33.

Tabela 33 – Resultado da aplicação das medidas agregadas

modelo de *features*	**SIFM**	**STIFM**	**FIFM**	**DIFM**
Smart-phone	24.64	6.60	25.67	2.11
WSN	14.75	6.39	30.31	1.34
Movie System	24.26	6.87	25.78	1.40
WindFarm	30.00	6.13	11.46	3.50
Online-book-shopping	47.30	4.00	14.18	6.40
Dynamic Software Product Lines for Service-Based Systems	21.62	6.41	61.90	4.00
ADS	34.03	4.16	47.50	3.50
Smart Home	27.44	4.00	31.22	3.50
ConferenceContext	26.53	4.12	28.73	2.00
Robot experiment	25.00	3.65	36.07	3.63
Wireless sensor actuator network	38.55	4.16	73.63	3.50
MobiHome	23.77	6.41	28.93	1.89
Scenario Application	24.64	6.95	23.49	1.38
Mobile Game	26.89	2.94	31.32	6.18
SmartHotel	24.35	6.42	74.59	1.63
VsSystem	29.71	4.00	22.79	4.00
Aircraft	22.39	6.75	72.08	2.00
Heidelberg Ion-Beam Therapy Center	77.55	4.01	26.51	6.66
PPU	25.40	4.00	23.46	2.88
Navigation Protocol	25.43	6.89	22.85	1.53
Bikesharing	30.00	4.32	61.79	6.76
CarSensors	25.19	5.87	27.52	2.71
Congress Assistant	24.32	6.95	26.98	0.82
DSPLFamily	65.96	4.00	16.81	7.41
Nexus DSPL	25.90	4.13	25.70	5.43
SalesScenario	50.99	3.90	60.04	5.89
Mobile Tourist Planner	16.95	7.00	24.16	0.75
Linux	35.41	6.14	66.09	7.52
Content Store	24.64	6.95	23.49	1.86
Visual Data Graph	20.71	6.71	29.87	1.86

A Tabela 34 apresenta, para cada medida agregada, o percentual dos modelos de *features* que foram classificados de acordo com os intervalos: muito baixo, baixo, médio, alto e muito alto. Em relação ao tamanho, pode-se notar que 80% dos modelos de *features* apresentam um tamanho médio. Isto indica que esses modelos de *features*, em sua grande maioria, possuem complexidade e variabilidade média.

Em relação a estabilidade do modelo de *features*, a Tabela 34 mostra que a maioria dos modelos de *features* (53.33%) apresentaram estabilidade muito alta. Este resultado indica que a maioria dos modelos de *features* são estáveis, e mudanças nestes modelos não terão um

impacto negativo elevado. Quanto ao índice de flexibilidade do modelo de *features* (FIFM), pode-se observar que a maioria dos modelos apresenta uma flexibilidade classificada como média (43.33%) ou baixa (30%) (ver Tabela 34). Este resultado está consistente com as conclusões obtidas a partir da análise do índice de estabilidade.

Em relação à dinamicidade do modelo de *features*, a Tabela 34 mostra que a maioria dos modelos possui uma dinamicidade baixa (43.33%), indicando que estes modelos têm poucas *features* de contexto e adaptação de contexto. Entretanto, (26.67%) dos modelos de *features* analisados têm dinamicidade muito alta, indicando que estes modelos tem um nível alto de adaptação de contexto.

Tabela 34 – Percentual dos modelos de *features* por intervalo para medidas de tamanho, estabilidade, flexibilidade e dinamicidade.

Intervalos	**% modelos de *features***			
	SIFM	**STIFM**	**FIFM**	**DIFM**
Muito Baixo	0%	0%	0%	6.67%
Baixo	6.67%	3.33%	30%	43.33%
Médio	80%	23.33%	43.33%	16.67%
Alto	6.67%	20%	3.33%	6.67%
Muito Alto	6.67%	53.33%	23.33%	26.67%

A Figura 27 apresenta um exemplo do cálculo da medida de dinamicidade (DIFM) para o modelo de *features Bikesharing*. Os valores de entrada do modelo de *features* para esta medida são: CF (15), CFC (5), ADCA (17) e DFCA (11). O modelo tem várias *features* de contexto, restrições com *features* de contexto e ativação de *features*. Devido a estes fatores, o valor de saída de *fuzzy* do modelo é muito alto (6.73). Pode-se ver na Figura 27 que o centro de gravidade da medida DIFM é localizado no intervalo muito alto.

Figura 27 – Índice de dinamicidade do modelo de *features* de *bikesharing*.

O índice de dinamicidade (DIFM) pode ajudar a caracterizar se o modelo de *features* é dinâmico ou não. No caso do modelo de *bikesharing*, pode-se ver que o modelo realmente tem muitas adaptações e *features* de contexto, apesar de ter um tamanho médio. Outros modelos apresentados na Tabela 33 têm um valor de DIFM muito baixo (e.g., *WSN, movie system, Congress Assistant, Mobile Tourist Planner*, etc.). Apesar dos trabalhos na literatura introduzirem estes modelos de *features* como modelos de LPSDs, os resultados dessa medida mostram que não se pode considerar estes modelos como dinâmicos.

Dessa forma, os engenheiros de domínio podem utilizar as medidas agregadas para maior inferência sobre os 4 fatores, em vez de utilizar o resultado de várias medidas individuais. No entanto, outros fatores de manutenibilidade não foram atacados nesse estudo, possibilitando outros trabalhos futuros para agregação de novas medidas de acordo com a metodologia seguida neste trabalho.

9.5 Ameaças à Validade

Esta pesquisa pode ser afetada por diferentes fatores, que podem invalidar as descobertas principais deste trabalho.

Validade Interna. Validade interna está preocupada com o ambiente dado e a confiabilidade dos resultados. Foram derivadas novas medidas para avaliar modelos de *features* de LPSDs. Entretanto, estas medidas são aplicáveis somente em modelos de *features* únicos de LPSDs, ou seja, nos modelos de *features* que apresentam as *features* de contexto e features que não são de contexto em um modelo único. Foi utilizado o *dataset* chamado ESPREssO, que contém uma coleção de 13 medidas para 30 modelos de *features* de LPSDs extraídos da literatura.

Para mitigar possíveis valores errados do *dataset*, foram coletadas todas as medidas automaticamente pela ferramenta DyMMer. Foram utilizados os modelos de *features* de LPSDs extraídos da literatura que foram projetados para fins acadêmicos e de pesquisa. Para derivação das medidas baseadas na lógica *fuzzy*, foram utilizados especialistas em LPSs e LPSDs. Foram mitigadas estas ameaças por meio do uso de 4 especialistas e do *dataset* ESPREssO para definir os parâmetros das funções de pertinência e regras *fuzzy* para as medidas agregadas (SIFM, STIFM, FIFM e DIFM).

Validade Externa. Validade externa é a extensão em que os resultados obtidos em estudo de caso podem ser generalizados para outros cenários de pesquisa relevantes. Os

resultados do estudo de caso validado externamente pode ser generalizado e aplicado com segurança para a prática da engenharia de software e ser recomendado como padrão. Para mitigar a validade externa, foram escolhidos um amplo conjunto de modelos de *features*, com diferentes tamanhos e complexidades. Além disso, medidas de qualidade usadas neste trabalho foram extraídas de medidas existentes na literatura e de outras medidas definidas para modelos de *features* de LPSDs. Este estudo considera apenas 30 modelos de *features* de LPSDs para validação das 4 medidas agregadas. Este número de modelos é pequeno e pode ser visto como uma ameaça à validade externa.

Validade de Construção. Validade de construção está preocupada com o relacionamento entre teoria e observação. As novas medidas agregadas derivadas pela lógica *fuzzy* são compostas de outras medidas para avaliar fatores de tamanho, estabilidade, flexibilidade e dinamicidade de modelos de *features* de LPSDs. Entretanto, outras medidas poderiam compor estes fatores, os quais podem influenciar em um resultado diferente para avaliação dos modelos de *features*. Para mitigar esta ameaça, foram validados os resultados de entrada e saída e as regras *fuzzy* com quatro especialistas de LPSs e LPSDs para compor as medidas agregadas.

9.6 Conclusões

Neste capítulo, foi discutida a investigação realizada com a finalidade de agregar medidas de manutenibilidade voltadas para modelos de *features* de LPSDs. Para isso, foram utilizadas as 13 medidas de qualidade pertencentes ao *dataset* ESPREssO. Em seguida, foi aplicada a teoria de lógica *fuzzy*, a fim de agregar estas medidas, produzindo 4 novas medidas agregadas. As novas medidas agregadas podem ser aplicadas para avaliar características específicas de modelos de *features* em LPSDs, tais como: tamanho, estabilidade, flexibilidade e dinamicidade.

Adicionalmente, a utilização das novas medidas agregadas foi validada por meio da aplicação dessas medidas nos 30 modelos de *features* de LPSDs presentes no *dataset* ESPREssO.

As descobertas deste estudo sugerem que medidas agregadas podem ser efetivamente usadas para apoiar a avaliação da manutenibilidade dos modelos de *features* de LPSDs.

10 CONCLUSÕES E TRABALHOS FUTUROS

Este capítulo é dedicado às considerações finais e os trabalhos futuros que poderão ser derivados desta Tese de doutorado. A Seção 10.1 apresenta os resultados alcançados por esta tese de doutorado. A Seção 10.2 descreve as principais contribuições desta Tese, respondendo as questões de partida, relacionadas à hipótese definida no início deste trabalho. A Seção 10.3 apresenta as publicações alcançadas no decorrer do doutorado. Por fim, na Seção 10.4 são discutidas as direções para os trabalhos futuros.

10.1 Resultados Alcançados

Os principais resultados desta pesquisa de doutorado foram: (i) um catálogo de medidas de qualidade denominado COfFEE, o qual contém 40 medidas relacionadas à manutenibilidade do modelo de *features* (Capítulo 5); (ii) uma ferramenta, denominada DyMMer, para suportar a avaliação da qualidade do modelo de *features*, possibilitando a coleta automática das medidas pertencentes ao catálogo COfFEE e a modelagem das informações de contexto utilizadas nos modelos de *features* de LPSDs (Capítulo 6); (iii) a construção de três *datasets*, denominados AFFOgaTO, MAcchiATO, ESPREssO, que podem ser utilizados para auxiliar a avaliação da manutenibilidade de modelos de *features* (Capítulo 6); e (iv) a agregação de medidas utilizando lógica *fuzzy*, gerando novas 4 medidas agregadas, as quais estão relacionadas a diferentes aspectos do modelo de *features*: tamanho, estabilidade, flexibilidade e dinamicidade (Capítulo 9).

Para avaliação das medidas de manutenibilidade do modelo de *features* foram realizados dois estudos: (i) um estudo exploratório para investigar o impacto do processo de evolução dos modelos de *features* na manutenibilidade destes modelos (Capítulo 7); e (ii) um estudo de caso para investigar a aplicação das medidas voltadas para LPSs, por meio da análise da correlação entre essas medidas, agrupamento dessas medidas e da definição de *thresholds* para as medidas (Capítulo B).

10.2 Hipótese e Questões de Pesquisa

Nesta tese foi levantada a seguinte hipótese:

É possível identificar um conjunto de medidas de qualidade que possa ser utilizado

para suportar a avaliação da manutenibilidade do modelo de features.

Os resultados deste trabalho confirmaram que esta hipótese foi aceita. As questões de partida (QP) definidas nesta Tese são discutidas a seguir:

- **QP1:** Que características e subcaracterísticas são relevantes para a avaliação da qualidade do modelo de *features*?

  Para responder essa questão, esta Tese realizou um mapeamento sistemático para identificação de características, subcaracteristicas e medidas que são importantes para avaliação da qualidade do modelo de *features* em LPSs, detalhada na Seção 5.1. Também foi realizada uma revisão da literatura de forma não sistemática para extração de características, subcaracterísticas e medidas de qualidade importantes para avaliação do modelo de *features* em LPSDs, detalhada na Seção 5.2. A partir dessas revisões foi elaborado um catálogo contendo as medidas mais relevantes para avaliação da manutenibilidade do modelo de *features* em LPSs e LPSDs, o qual foi discutido no Capítulo 5. A partir deste catálogo, foi possível verificar que a característica de qualidade mais relevante, conforme os achados, é a manutenibilidade. Dessa forma, este Tese de doutorado teve como um de seus objetivos a identificação e utilização de medidas de manutenibilidade para avaliação da qualidade de modelos de *features* em LPSs e LPSDs.

- **QP2:** Quais as medidas de qualidade que podem ser utilizadas para avaliar a manutenibilidade dos modelos de *features*?

  A partir das revisões da literatura foram identificadas medidas para diversas características e subcaracterísticas de qualidade. Com este resultado foram selecionadas 40 medidas relacionadas à manutenibilidade do modelo de *features* para compor um catálogo de medidas denominado COfFEE, o qual foi discutido na Seção 5.4.

- **QP3:** É possível avaliar a manutenibilidade do modelo de *features* utilizando medidas de qualidade?

  Para responder essa questão de partida foram executados dois estudos exploratórios relacionados à avaliação da manutenibilidade do modelo de *features* utilizando medidas:

  - **Estudo Exploratório:** Esse estudo utilizou o *dataset* AFFOgaTO para investigar o impacto do processo de evolução dos modelos de *features* na manutenibilidade destes modelos (Capítulo 7). Como conclusões deste estudo, pode-se observar que o número de *features* tende a aumentar ao longo do processo de evolução dos modelos de *features*. Este crescimento impacta diretamente a analisabilidade

e modificabilidade do modelo de *features*. Foi possível observar também que a largura do modelo de *features* tende a aumentar ao longo do tempo. Modelos de *features* largos tendem a apresentar uma baixa modificabilidade. Adicionalmente, observou-se que a variabilidade, a complexidade estrutural e a flexibilidade dos modelos de *features* tendem a crescer com o tempo. A conclusão geral deste estudo é que o processo de evolução dos modelos de *features* de LPSs tende a diminuir a manutenibilidade do modelo de *features*. Por meio deste estudo de caso, pode-se concluir que é possível suportar a avaliação da qualidade por meio de medidas.

  - **Estudo de Caso Exploratório:** O estudo de caso realizado neste trabalho teve como objetivo investigar como as medidas de qualidade podem ser aplicadas para a avaliação da manutenibilidade dos modelos de *features* em LPSs (Capítulo B). Foram exploradas três técnicas de análise de dados diferentes, a fim de identificar relações entre as medidas, agrupar medidas de qualidade e definir *thresholds* para as medidas. Os resultados do estudo mostraram que existem correlações entre as medidas de qualidade e que técnica de PCA pode ser usada para agrupar medidas. Por fim, foi possível utilizar a técnica de intervalo de tolerância para definir *thresholds* para as medidas com base nos dados coletados. Os resultados do estudo sugerem que medidas podem ser efetivamente utilizadas para apoiar a avaliação da qualidade do modelos de *features* em LPSs.

- **QP4:** Como suportar a avaliação da manutenibilidade do modelo de *features* por meio de medidas de qualidade?

  Para responder esta questão de partida foram construídos a ferramenta DyMMer e os *datasets* AFFOgaTO, MAcchiATO e ESPREssO (Capítulo 6.1). A ferramenta denominada DyMMer foi desenvolvida com o objetivo de suportar a coleta automática das medidas do catálogo COfFEE e a modelagem dos modelos de *features* de LPSDs. Já os *datasets* foram utilizados em três estudos exploratórios. Pode-se concluir, pelos resultados dos estudos exploratórios realizados, com a utilização do catálogo de medidas COfFEE juntamente com a ferramenta DyMMer e os *datasets* construídos, é possível avaliar a qualidade do modelo de *features* de LPSs e LPSDs utilizando medidas.

- **QP5:** As medidas de qualidade existentes são suficientes para suportar a avaliação da manutenibilidade do modelo de *features*?

  Para responder essa questão de pesquisa, foi realizado um estudo com a finalidade de

explorar a utilização da lógica *fuzzy* na agregação de medidas (Capítulo 9). Neste estudo, foi utilizado o *dataset* ESPREssO, o qual contém os valores de 13 medidas do catálogo COfFEE para 30 modelos de *features* de LPSDs, extraídos da literatura. Como resultado deste estudo foram concebidas 4 novas medidas agregadas, as quais estão relacionadas a diferentes aspectos: tamanho, estabilidade, flexibilidade e dinamicidade. Adicionalmente, pode-se concluir que não existiam medidas para determinados aspectos, tais como tamanho e dinamicidade. Por esse motivo, concluiu-se que as medidas de qualidade existentes não eram suficientes para suportar a avaliação da manutenibilidade do modelo de *features*. Por este motivo, 4 novas medidas agregadas foram concebidas. As descobertas do estudo sugerem também que medidas agregadas podem ser efetivamente usadas para apoiar a avaliação da qualidade de modelos de *features* de LPSDs.

10.3 Publicações

Durante o desenvolvimento desta Tese de doutorado foram publicados trabalhos em conferências e periódicos importantes, conforme mencionado no Capítulo 1. A seguir são listados os principais artigos que resultaram desta pesquisa:

- Bezerra, C. I. M.; Andrade, R. M. C.; Monteiro, J. M. S. (2016) Exploring Quality Measures for the Evaluation of Feature Models: A Case Study. Journal of Systems and Software. (aceito para publicação)
- Bezerra, C. I. M., Barbosa, J., Freires, J. H., Andrade, R. M. C., e Monteiro, J. M. (2016) DyMMer: a measurement-based tool to support quality evaluation of DSPL feature models. In Proceedings of the 20th International Systems and Software Product Line Conference (pp. 314-317). ACM.
- Bezerra, C. I. M., Monteiro, J. M., Andrade, R., Rocha, L. S. (2016). Analyzing the Feature Models Maintainability over their Evolution Process: An Exploratory Study. In Proceedings of the Tenth International Workshop on Variability Modelling of Software-intensive Systems (pp. 17-24). ACM.
- Bezerra, C. I. M., Andrade, R. M., Monteiro, J. M. S. (2015). Measures for quality evaluation of feature models. In International Conference on Software Reuse (pp. 282-297). Springer International Publishing.
- Bezerra, C. I. M.; Andrade, R. M. C.; Monteiro, J. M. S. Um Método de Avaliação da Qualidade do Modelo de Features em Linhas de Produtos de Software Baseado em

Medidas. In: III Workshop de Teses e Dissertações do CBsoft. [S.l.]: IV Congresso Brasileiro de Software de 2013: Teoria e Prática, 2013.

- Bezerra, C. I. M., Andrade, R. M., Monteiro, J. M. S. (2013) Avaliação da qualidade do modelo de *features* em linhas de produto de software utilizando medidas. In: Simpósio Brasileiro de Qualidade de Software (SBQS), Salvador-BA.

Finalmente, são apresentadas outras publicações que não estão diretamente relacionadas à esta Tese de doutorado, mas foram contribuições importantes para o crescimento científico da pesquisadora:

- Carvalho, R. M., de Castro Andrade, R. M., de Oliveira, K. M., de Sousa Santos, I., Bezerra, C. I. M. (2016) Quality characteristics and measures for human–computer interaction evaluation in ubiquitous systems. Software Quality Journal, 1-53.
- Jorge, F. D. F., Bezerra, C. I., Coutinho, E. F., Monteiro, J. M., Andrade, R. M. (2015) A Evolução do Jogo iTestLearning para o Ensino das Atividades de Execução de Testes de Software. In XX Conferência Internacional sobre Informática na Educação (TISE2015), Santiago.
- Bezerra, C. I. M., Coutinho, E. F., Santos, I. S., Monteiro, J. M., Andrade, R. M. (2014). Evolução do Jogo ItestLearning para o Ensino de Testes de Software: Do Planejamento ao Projeto. In XIX Conferência Internacional sobre Informática na Educação (TISE2014), Fortaleza.
- Bezerra, C. I. M., Andrade, R., Santos, R. M., Abed, M., de Oliveira, K. M., Monteiro, J. M., Ezzedine, H. (2014, October). Challenges for usability testing in ubiquitous systems. In Proceedings of the 26th Conference on l'Interaction Homme-Machine (pp. 183-188). ACM.
- Santos, I. S.; Bezerra, C. I. M.; Monteiro, G. S.; Santos, R. M.; Araujo, I. L.; Oliveira, T. A.; Dantas, V. L. L.; Andrade, R. M. C. Uma avaliação de ferramentas para testes em sistemas de informação móveis baseada no método DMADV. In: IX Simpósio Brasileiro de Sistemas de Informação (SBSI), 2013, João Pessoa.
- Braga, M. R., Bezerra, C. I. M., Monteiro, J. M. S., Andrade, R. (2012, September). A pattern language for agile software estimation. In Proceedings of the 9th Latin-American Conference on Pattern Languages of Programming (p. 5). ACM.

10.4 Trabalhos Futuros

Após a realização desta pesquisa, identificou-se diversas lacunas no tema investigado, as quais podem resultar em trabalhos futuros. A seguir, discute-se as oportunidades identificadas para investigações futuras:

- **Definir uma abordagem de avaliação da qualidade do modelo de *features*:** Um dos trabalhos futuros é a elaboração de uma abordagem de avaliação da qualidade do modelo de *features* utilizando o catálogo de medidas e a ferramenta desenvolvidos neste trabalho. A abordagem pode guiar o engenheiro de domínio no processo de avaliação da qualidade de forma sistemática.
- **Estender o catálogo COfFEE:** O catálogo COfFEE é composto por medidas de manutenibilidade, devido as outras medidas identificadas estarem relacionadas à outras características de qualidade que não foram possíveis de coletar. Além disso, ainda existem poucas medidas específicas para o modelo de *features* de LPSDs. Um dos trabalhos futuros vislumbrados é a extensão desse catálogo com outras medidas para outras características de qualidade e medidas específicas para os modelos de *features* de LPSDs.
- **Construir um *dataset* com modelos de *features* de LPSs e LPSDs reais:** Neste trabalho foram utilizados modelos de *features* extraídos da ferramenta S.P.L.O.T. e da literatura. Muitos desses modelos de *features* são modelados apenas para representar um determinado domínio, mas não representam LPSs e LPSDs completas e reais. Como trabalho futuro, pode-se identificar LPSs e LPSDs reais e a partir delas criar um *dataset* com modelos de *features* dessas linhas. A partir dessas LPSs e LPSDs reais é possível acessar todas as versões do modelo de *features* da linha e construir uma base histórica de medições para analisar a evolução de cada linha.
- **Elaborar diretrizes para melhoria do modelo de *features*:** Outro trabalho futuro identificado no escopo da Tese é a criação de diretrizes de melhoria do modelo de *features* durante seu processo de evolução. A pesquisa desenvolvida neste trabalho apenas avalia o modelo de *features* e de acordo com as medidas avaliadas poderiam ser sugeridas diretrizes de melhoria para o engenheiro de domínio.
- **Agregar outras medidas de qualidade:** Neste trabalho são agregadas apenas algumas medidas para o modelo de *features* de LPSs e LPSDs. No entanto, existem outras características e subcaracterísticas de qualidade que podem ser calculadas a partir da agregação de outras medidas. Um possível trabalho futuro é a agregação de outras medidas para o

cálculo de outros fatores de qualidade do modelo de *features*.

- **Utilizar técnicas de aprendizagem de máquina para classificação dos modelos de *features*:** Um dos trabalhos futuros que podem também ser realizados é a utilização de técnicas de aprendizagem de máquina para a partir dos conjuntos de dados de todas as medidas, classificar o modelo como bom ou ruim. Neste trabalho de doutorado foram agrupadas e agregadas algumas medidas, e a partir de alguns fatores de qualidade pode-se calcular a qualidade do modelo de *features* para cada fator.

REFERÊNCIAS

ABDI, H.; WILLIAMS, L. J. Principal component analysis. **Wiley Interdisciplinary Reviews: Computational Statistics**, Wiley Online Library, v. 2, n. 4, p. 433–459, 2010.

ABILIO, R.; PADILHA, J.; FIGUEIREDO, E.; COSTA, H. Detecting code smells in software product lines–an exploratory study. In: IEEE. **Information Technology-New Generations (ITNG), 2015 12th International Conference on**. [S.l.], 2015. p. 433–438.

ACHER, M.; CLEVE, A.; COLLET, P.; MERLE, P.; DUCHIEN, L.; LAHIRE, P. Extraction and evolution of architectural variability models in plugin-based systems. **Software & Systems Modeling**, Springer, v. 13, n. 4, p. 1367–1394, 2014.

ACHER, M.; COLLET, P.; LAHIRE, P.; FRANCE, R. B. Familiar: A domain-specific language for large scale management of feature models. **Science of Computer Programming**, Elsevier, v. 78, n. 6, p. 657–681, 2013.

ALFEREZ, G. H.; PELECHANO, V. Context-aware autonomous web services in software product lines. In: IEEE. **Software Product Line Conference (SPLC), 2011 15th International**. [S.l.], 2011. p. 100–109.

ALFÉREZ, G. H.; PELECHANO, V.; MAZO, R.; SALINESI, C.; DIAZ, D. Dynamic adaptation of service compositions with variability models. **Journal of Systems and Software**, Elsevier, v. 91, p. 24–47, 2014.

ALFÉREZ, M.; BONIFÁCIO, R.; TEIXEIRA, L.; ACCIOLY, P.; KULESZA, U.; MOREIRA, A.; ARAUJO, J.; BORBA, P. Evaluating scenario-based spl requirements approaches: the case for modularity, stability and expressiveness. **Requirements Engineering**, Springer, v. 19, n. 4, p. 355–376, 2014.

ALMEIDA, E. S.; ALVARO, A.; GARCIA, V. C.; MASCENA, J. C. C. P.; BURÉGIO, V. A. A.; NASCIMENTO, L. M.; LUCRÉDIO, D.; L., M. S. **C.R.U.I.S.E: Component Reuse in Software Engineering**. [S.l.]: C.E.S.A.R e-book, 2007.

ALTMAN, D.; MACHIN, D.; BRYANT, T.; GARDNER, M. **Statistics with confidence: confidence intervals and statistical guidelines**. [S.l.]: John Wiley & Sons, 2013.

ANJORIN, A.; SALLER, K.; REIMUND, I.; OSTER, S.; ZORCIC, I.; SCHÜRR, A. Model-driven rapid prototyping with programmed graph transformations. **Journal of Visual Languages & Computing**, Elsevier, v. 24, n. 6, p. 441–462, 2013.

ARCEGA, L.; FONT, J.; CETINA, C. *et al.* Achieving knowledge evolution in dynamic software product lines. In: IEEE. **2016 IEEE 23rd International Conference on Software Analysis, Evolution, and Reengineering (SANER)**. [S.l.], 2016. v. 1, p. 505–516.

ASADI, M.; GRÖNER, G.; MOHABBATI, B.; GAŠEVIĆ, D. Goal-oriented modeling and verification of feature-oriented product lines. **Software & Systems Modeling**, Springer, v. 15, n. 1, p. 257–279, 2016.

BAGHERI, E.; ENSAN, F.; GASEVIC, D. Decision support for the software product line domain engineering lifecycle. **Automated Software Engineering**, Springer, v. 19, n. 3, p. 335–377, 2012.

BAGHERI E. E GASEVIC, D. Assessing the maintainability of software product line feature models using structural metrics. **Software Quality Control**, Kluwer Academic Publishers, Hingham, MA, USA, v. 19, n. 3, p. 579–612, set. 2011. ISSN 0963-9314.

BARCELLOS, M. P. **Uma estratégia para medição de software e avaliação de bases de medidas para controle estatístico de processos de software em organizações de alta maturidade**. Tese (Doutorado) — Tese de Doutorado. Universidade Federal do Rio de Janeiro, 2009.

BARESI, L.; GUINEA, S.; PASQUALE, L. Service-oriented dynamic software product lines. **Computer**, Institute of Electrical and Electronics Engineers, Inc., 3 Park Avenue, 17 th Fl New York NY 10016-5997 United States, v. 45, n. 10, p. 42–48, 2012.

BASS, L.; CLEMENTS, P. **Constructing Superior Software; Applying Proven Practices**. [S.l.]: New Riders Publishing, 1999.

BATORY, D. **Feature models, grammars, and propositional formulas**. [S.l.]: Springer, 2005.

BELATEGI, L.; SAGARDUI, G.; ETXEBERRIA, L. Model based analysis process for embedded software product lines. In: ACM. **Proceedings of the 2011 International Conference on Software and Systems Process**. [S.l.], 2011. p. 53–62.

BENAVIDES, D.; SEGURA, S.; RUIZ-CORTÉS, A. Automated analysis of feature models 20 years later: A literature review. **Information Systems**, Elsevier, v. 35, n. 6, p. 615–636, 2010.

BENAVIDES, D.; SEGURA, S.; TRINIDAD, P.; CORTÉS, A. R. Fama: Tooling a framework for the automated analysis of feature models. In: **Proc. Workshop Variability Modelling of Software-intensive Systems, VaMoS**. [S.l.: s.n.], 2007. p. 129–134.

BENAVIDES, D.; TRINIDAD, P.; RUIZ-CORTÉS, A. Automated reasoning on feature models. In: SPRINGER. **Advanced Information Systems Engineering**. [S.l.], 2005. p. 491–503.

BERGER T. E GUO, J. Towards system analysis with variability model metrics. In: ACM. **Proceedings of the Eighth International Workshop on Variability Modelling of Software-Intensive Systems**. [S.l.], 2014. p. 23.

BERKHIN, P. A survey of clustering data mining techniques. In: **Grouping multidimensional data**. [S.l.]: Springer, 2006. p. 25–71.

BEZERRA, C.; ANDRADE, R.; SANTOS, R. M.; ABED, M.; OLIVEIRA, K. M. de; MONTEIRO, J. M.; SANTOS, I.; EZZEDINE, H. Challenges for usability testing in ubiquitous systems. In: ACM. **Proceedings of the 26th Conference on l'Interaction Homme-Machine**. [S.l.], 2014. p. 183–188.

BEZERRA, C. I.; ANDRADE, R. M.; MONTEIRO, J. M. Exploring quality measures for the evaluation of feature models: A case study. **Journal of Systems and Software**, Elsevier, 2016.

BEZERRA, C. I.; COUTINHO, E. F.; SANTOS, I. S.; MONTEIRO, J. M.; ANDRADE, R. M. Evolução do jogo itestlearning para o ensino de testes de software: Do planejamento ao projeto. In: **XIX Conferência Internacional sobre Informática na Educação (TISE2014), Fortaleza**. [S.l.: s.n.], 2014.

BEZERRA, C. I.; MONTEIRO, J. M.; ANDRADE, R.; ROCHA, L. S. Analyzing the feature models maintainability over their evolution process: An exploratory study. In: ACM. **Proceedings of the Tenth International Workshop on Variability Modelling of Software-intensive Systems**. [S.l.], 2016. p. 17–24.

BEZERRA, C. I.; MONTEIRO, J. M. S.; ANDRADE, R. Avaliação da qualidade do modelo de *Features* em linhas de produto de software utilizando medidas. In: **Simpósio Brasileiro de Qualidade de Software**. [S.l.: s.n.], 2013. p. 15.

BEZERRA, C. I.; MONTEIRO, J. M. S.; ANDRADE, R. Um método de avaliação da qualidade do modelo de *Features* em linhas de produtos de software baseado em medidas. In: **III Workshop de Teses e Dissertações do CBsoft. [S.l.]: IV Congresso Brasileiro de Software de 2013: Teoria e Prática**. [S.l.: s.n.], 2013. p. 12.

BEZERRA, C. I. M.; ANDRADE, R. M. C.; MONTEIRO, J. M. Measures for quality evaluation of feature models. In: **Software Reuse for Dynamic Systems in the Cloud and Beyond - 14th International Conference on Software Reuse, ICSR 2015, Miami, FL, USA, January 4-6, 2015. Proceedings**. [s.n.], 2015. p. 282–297. Disponível em: <http://dx.doi.org/10.1007/978-3-319-14130-5_20>.

BOEHM, B.; BROWN, J.; KASPAR, H.; LIPOW, H.; LEOD, M. M.; MERRITT, G. Characteristics of software quality. **American Elsevier, New York TRW series of software technology**, North-Holland Pub. Co, Amsterdam (1978) ISBN 0444851054, v. 1, n. 1, 1978.

BOSCH, J. Software variability management. In: IEEE COMPUTER SOCIETY. **Proceedings of the 26th international Conference on Software Engineering**. [S.l.], 2004. p. 720–721.

BOSCH, J.; CAPILLA, R. Dynamic variability in software-intensive embedded system families. **Computer**, Institute of Electrical and Electronics Engineers, Inc., 3 Park Avenue, 17 th Fl New York NY 10016-5997 United States, v. 45, n. 10, p. 28–35, 2012.

BRAGA, M. R.; BEZERRA, C. I.; MONTEIRO, J. M. S.; ANDRADE, R. A pattern language for agile software estimation. In: ACM. **Proceedings of the 9th Latin-American Conference on Pattern Languages of Programming**. [S.l.], 2012. p. 5.

BRIAND, L. C.; MORASCA, S.; BASILI, V. R. Property-based software engineering measurement. **IEEE Transactions on Software Engineering**, IEEE, v. 22, n. 1, p. 68–86, 1996.

BRO, R.; SMILDE, A. K. Principal component analysis. **Analytical Methods**, Royal Society of Chemistry, v. 6, n. 9, p. 2812–2831, 2014.

BROWNE, M. W. Cross-validation methods. **Journal of Mathematical Psychology**, Elsevier, v. 44, n. 1, p. 108–132, 2000.

BRYANT, R. E. Graph-based algorithms for boolean function manipulation. **Computers, IEEE Transactions on**, IEEE, v. 100, n. 8, p. 677–691, 1986.

BUSH, M. E.; FENTON, N. E. Software measurement: a conceptual framework. **Journal of Systems and Software**, Elsevier, v. 12, n. 3, p. 223–231, 1990.

CAPILLA, R.; BOSCH, J. The promise and challenge of runtime variability. **Computer**, IEEE, v. 44, n. 12, p. 93–95, 2011.

CAPILLA, R.; BOSCH, J.; KANG, K.-C. **Systems and software variability management**. [S.l.]: Springer, 2013.

CAPILLA, R.; BOSCH, J.; TRINIDAD, P.; RUIZ-CORTÉS, A.; HINCHEY, M. An overview of dynamic software product line architectures and techniques: Observations from research and industry. **Journal of Systems and Software**, Elsevier, v. 91, p. 3–23, 2014.

CAPILLA, R.; ORTIZ, ; HINCHEY, M. Context variability for context-aware systems. **Computer**, v. 47, n. 2, p. 85–87, Feb 2014. ISSN 0018-9162.

CARVALHO, R. M.; ANDRADE, R. M. de C.; OLIVEIRA, K. M. de; SANTOS, I. de S.; BEZERRA, C. I. M. Quality characteristics and measures for human–computer interaction evaluation in ubiquitous systems. **Software Quality Journal**, Springer, p. 1–53, 2016.

CAVALCANTE, E.; ALMEIDA, A.; BATISTA, T.; CACHO, N.; LOPES, F.; DELICATO, F. C.; SENA, T.; PIRES, P. F. Exploiting software product lines to develop cloud computing applications. In: ACM. **Proceedings of the 16th International Software Product Line Conference-Volume 2**. [S.l.], 2012. p. 179–187.

CETINA, C.; GINER, P.; FONS, J.; PELECHANO, V. Autonomic computing through reuse of variability models at runtime: The case of smart homes. **Computer**, IEEE, v. 42, n. 10, p. 37–43, 2009.

CETINA, C.; GINER, P.; FONS, J.; PELECHANO, V. Using feature models for developing self-configuring smart homes. In: IEEE. **Autonomic and Autonomous Systems, 2009. ICAS'09. Fifth International Conference on**. [S.l.], 2009. p. 179–188.

CETINA, C.; GINER, P.; FONS, J.; PELECHANO, V. Prototyping dynamic software product lines to evaluate run-time reconfigurations. **Science of Computer Programming**, Elsevier, v. 78, n. 12, p. 2399–2413, 2013.

CHEN, G.; LI, M.; KOTZ, D. Data-centric middleware for context-aware pervasive computing. **Pervasive and mobile computing**, Elsevier, v. 4, n. 2, p. 216–253, 2008.

CHEN, L.; BABAR, M. A. A systematic review of evaluation of variability management approaches in software product lines. **Information and Software Technology**, Elsevier, v. 53, n. 4, p. 344–362, 2011.

CINGOLANI, P.; ALCALA-FDEZ, J. jfuzzylogic: a robust and flexible fuzzy-logic inference system language implementation. In: CITESEER. **FUZZ-IEEE**. [S.l.], 2012. p. 1–8.

CLEMENTS P. E NORTHROP, L. **Software product lines: practices and patterns**. [S.l.]: Addison-Wesley, 2002.

COSTA, P. A. da S.; MARINHO, F. G.; ANDRADE, R. M. de C.; OLIVEIRA, T. Fixture-a tool for automatic inconsistencies detection in context-aware spl. In: **ICEIS - Proceedings of the 17th International Conference on Enterprise Information Systems, Volume 2, Barcelona, Spain**. [S.l.: s.n.], 2015. p. 114–125.

COURTNEY, R. E.; GUSTAFSON, D. A. Shotgun correlations in software measures. **Software Engineering Journal**, IET, v. 8, n. 1, p. 5–13, 1993.

CUEVAS, D. F. B.; RUEDA, S. S.; ARROYO, P. T. M.; CORTÉS, A. R. Fama: Tooling a framework for the automated analysis of feature models. 2007.

CZARNECKI, K.; HELSEN, S.; EISENECKER, U. Staged configuration using feature models. In: **Software Product Lines**. [S.l.]: Springer, 2004. p. 266–283.

CZARNECKI, K.; WASOWSKI, A. Feature diagrams and logics: There and back again. In: IEEE. **Software Product Line Conference, 2007. SPLC 2007. 11th International**. [S.l.], 2007. p. 23–34.

DEELSTRA, S.; SINNEMA, M.; BOSCH, J. Variability assessment in software product families. **Information and Software Technology**, Elsevier, v. 51, n. 1, p. 195–218, 2009.

DEY, A. K.; ABOWD, G. D.; SALBER, D. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. **Human-computer interaction**, L. Erlbaum Associates Inc., v. 16, n. 2, p. 97–166, 2001.

DINTZNER, N.; DEURSEN, A. V.; PINZGER, M. Extracting feature model changes from the linux kernel using fmdiff. In: **Proceedings of the Eighth International Workshop on Variability Modelling of Software-Intensive Systems**. New York, NY, USA: ACM, 2013. (VaMoS '14), p. 22:1–22:8. ISBN 978-1-4503-2556-1. Disponível em: <http://doi.acm.org/10.1145/2556624.2556631>.

DINTZNER, N.; DEURSEN, A. van; PINZGER, M. Analysing the linux kernel feature model changes using fmdiff. **Software & Systems Modeling**, Springer, p. 1–22, 2015.

DROMEY, R. G. Cornering the chimera [software quality]. **IEEE Software**, v. 13, n. 1, p. 33–43, Jan 1996. ISSN 0740-7459.

DUAN, Y.; KATTEPURY, A.; GETAHUN, F.; ELFAKIZ, A.; DU, W. Releasing the power of variability: Towards constraint driven quality assurance. In: IEEE. **Advanced Applied Informatics (IIAIAAI), 2013 IIAI International Conference on**. [S.l.], 2013. p. 15–20.

ETXEBERRIA, L.; SAGARDUI, G. Product-line architecture: New issues for evaluation. In: **Software Product Lines**. [S.l.]: Springer, 2005. p. 174–185.

ETXEBERRIA, L.; SAGARDUI, G. Quality assessment in software product lines. In: **High Confidence Software Reuse in Large Systems**. [S.l.]: Springer, 2008. p. 178–181.

ETXEBERRIA, L.; SAGARDUI, G. Variability driven quality evaluation in software product lines. In: IEEE. **Software Product Line Conference, 2008. SPLC'08. 12th International**. [S.l.], 2008. p. 243–252.

ETXEBERRIA, L.; SAGARDUI, G.; BELATEGI, L. Quality aware software product line engineering. **Journal of the Brazilian Computer Society**, SciELO Brasil, v. 14, n. 1, p. 57–69, 2008.

FENTON, N.; BIEMAN, J. **Software metrics: a rigorous and practical approach**. [S.l.]: CRC Press, 2014.

FENTON, N.; MELTON, A. Deriving structurally based software measures. **Journal of Systems and Software**, Elsevier, v. 12, n. 3, p. 177–187, 1990.

FERNANDES, P.; WERNER, C.; TEIXEIRA, E. An approach for feature modeling of context-aware software product line. **J. UCS**, v. 17, n. 5, p. 807–829, 2011.

FERNANDES, P. G. d. M. **Linha de produtos de software dinâmica direcionada por qualidade: o caso de redes de monitoração do corpo humano**. Dissertação (Mestrado) — Dissertação de Mestrado. Universidade de Brasília, Instituto de Ciências Exatas, Departamento de Ciência da Computação, 2013.

FIGUEIREDO, E.; CACHO, N.; SANT'ANNA, C.; MONTEIRO, M.; KULESZA, U.; GARCIA, A.; SOARES, S.; FERRARI, F.; KHAN, S.; FILHO, F. C. *et al.* Evolving software product lines with aspects: an empirical study on design stability. In: ACM. **Proceedings of the 30th international conference on Software engineering**. [S.l.], 2008. p. 261–270.

FLEUREY, F.; HAUGEN, Ø.; MØLLER-PEDERSEN, B.; SVENDSEN, A.; ZHANG, X. Standardizing variability–challenges and solutions. In: **SDL 2011: Integrating System and Software Modeling**. [S.l.]: Springer, 2011. p. 233–246.

FLORAC, W. A.; CARLETON, A. D. **Measuring the software process: statistical process control for software process improvement**. [S.l.]: Addison-Wesley Professional, 1999.

FRAKES, W.; KANG, K. *et al.* Software reuse research: Status and future. the IEEE Computer Society., 2005.

GALSTER, M.; WEYNS, D.; TOFAN, D.; MICHALIK, B.; AVGERIOU, P. Variability in software systems—a systematic literature review. **Software Engineering, IEEE Transactions on**, IEEE, v. 40, n. 3, p. 282–306, 2014.

GAMEZ, N.; FUENTES, L. Software product line evolution with cardinality-based feature models. In: **Top Productivity through Software Reuse**. [S.l.]: Springer, 2011. p. 102–118.

GAMEZ, N.; FUENTES, L.; ARAGÜEZ, M. A. Autonomic computing driven by feature models and architecture in famiware. In: **Software Architecture**. [S.l.]: Springer, 2011. p. 164–179.

GAMEZ, N.; FUENTES, L.; TROYA, J. M. Creating self-adapting mobile systems with dynamic software product lines. **IEEE Software**, IEEE, n. 2, p. 105–112, 2015.

GAMEZ, N.; ROMERO, D.; FUENTES, L.; ROUVOY, R.; DUCHIEN, L. Constraint-based self-adaptation of wireless sensor networks. In: ACM. **Proceedings of the 2nd International Workshop on Adaptive Services for the Future Internet and 6th International Workshop on Web APIs and Service Mashups**. [S.l.], 2012. p. 20–27.

GARCÍA, F.; BERTOA, M. F.; CALERO, C.; VALLECILLO, A.; RUÍZ, F.; PIATTINI, M.; GENERO, M. Towards a consistent terminology for software measurement. **Information and Software Technology**, Elsevier, v. 48, n. 8, p. 631–644, 2006.

GONZALEZ-HUERTA, J.; INSFRAN, E.; ABRAHAO, S. A multimodel for integrating quality assessment in model-driven engineering. In: IEEE. **Quality of Information and Communications Technology (QUATIC), 2012 Eighth International Conference on the**. [S.l.], 2012. p. 251–254.

GOUSIOS, G. The ghtorent dataset and tool suite. In: IEEE PRESS. **Proceedings of the 10th Working Conference on Mining Software Repositories**. [S.l.], 2013. p. 233–236.

GOUSIOS, G.; ZAIDMAN, A. A dataset for pull-based development research. In: ACM. **Proceedings of the 11th Working Conference on Mining Software Repositories**. [S.l.], 2014. p. 368–371.

GREENWOOD, P.; CHITCHYAN, R.; AYED, D.; GIRARD-REYDET, V.; FLEUREY, F.; DEHLEN, V.; SOLBERG, A. **Modelling service requirements variability: The DiVA way**. [S.l.]: Springer, 2011.

GRISS, M. L.; FAVARO, J.; ALESSANDRO, M. D. Integrating feature modeling with the rseb. In: IEEE. **Software Reuse, 1998. Proceedings. Fifth International Conference on**. [S.l.], 1998. p. 76–85.

GUO, J.; WANG, Y.; TRINIDAD, P.; BENAVIDES, D. Consistency maintenance for evolving feature models. **Expert Systems with Applications**, Elsevier, v. 39, n. 5, p. 4987–4998, 2012.

GURP, J. V.; BOSCH, J.; SVAHNBERG, M. On the notion of variability in software product lines. In: IEEE. **Software Architecture, 2001. Proceedings. Working IEEE/IFIP Conference on**. [S.l.], 2001. p. 45–54.

HALLSTEINSEN, S.; HINCHEY, M.; PARK, S.; SCHMID, K. Dynamic software product lines. **Computer**, IEEE, v. 41, n. 4, p. 93–95, 2008.

HALLSTEINSEN, S.; STAV, E.; SOLBERG, A.; FLOCH, J. Using product line techniques to build adaptive systems. In: IEEE. **Software Product Line Conference, 2006 10th International**. [S.l.], 2006. p. 10–pp.

HARTMANN, H.; TREW, T. Using feature diagrams with context variability to model multiple product lines for software supply chains. In: IEEE. **Software Product Line Conference, 2008. SPLC'08. 12th International**. [S.l.], 2008. p. 12–21.

HETTIARACHCHI, C.; DO, H.; CHOI, B. Risk-based test case prioritization using a fuzzy expert system. **Information and Software Technology**, Elsevier, v. 69, p. 1–15, 2016.

HINCHEY, M.; PARK, S.; SCHMID, K. Building dynamic software product lines. **Computer**, IEEE, n. 10, p. 22–26, 2012.

HONG, J.-y.; SUH, E.-h.; KIM, S.-J. Context-aware systems: A literature review and classification. **Expert Systems with Applications**, Elsevier, v. 36, n. 4, p. 8509–8522, 2009.

ISO/IEC. **ISO/IEC 9126-1. Software engineering–Product quality–Part 1: Quality model**. [S.l.], 2001.

ISO/IEC. **ISO/IEC 9126-1:2001. Software engineering – Product quality – Part 1: Quality model**. [S.l.], 2001.

ISO/IEC. **ISO/IEC 12207:2008. Systems and software engineering – Software life cycle processes**. [S.l.], 2008.

ISO/IEC. **ISO/IEC 25010 - Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models**. [S.l.], 2011.

ISO/IEC. **ISO/IEC 25000 - Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE**. [S.l.], 2014.

ISO/IEC. **ISO 9000:2015 - covers the basic concepts and language**. [S.l.], 2015.

JANAKIRAM, D.; RAJASREE, M. Request: Requirements-driven quality estimator. **ACM SIGSOFT Software engineering notes**, ACM, v. 30, n. 1, p. 4, 2005.

JEDLITSCHKA, A.; PFAHL, D. Reporting guidelines for controlled experiments in software engineering. In: IEEE. **Empirical Software Engineering, 2005. 2005 International Symposium on**. [S.l.], 2005. p. 10–pp.

JORGE, F. d. F.; BEZERRA, C. I.; COUTINHO, E. F.; MONTEIRO, J. M.; ANDRADE, R. M. A evolução do jogo itestlearning para o ensino das atividades de execução de testes de software. In: **XX Conferência Internacional sobre Informática na Educação (TISE2015), Santiago**. [S.l.: s.n.], 2015.

JUNIOR, E. A. O.; GIMENES, I. M.; MALDONADO, J. C.; MASIERO, P. C.; BARROCA, L. Systematic evaluation of software product line architectures. **Journal of Universal Computer Science**, v. 19, n. 1, p. 25–52, 2013.

KAN, S. H. **Metrics and models in software quality engineering**. [S.l.]: Addison-Wesley Longman Publishing Co., Inc., 2002.

KANG, K.; COHEN, S.; HESS, J.; NOVAK, W.; PETERSON, A. **Feature-Oriented Domain Analysis (FODA) Feasibility Study**. Pittsburgh, PA, 1990.

KANG, K. C.; KIM, S.; LEE, J.; KIM, K.; SHIN, E.; HUH, M. Form: A feature-; oriented reuse method with domain-; specific reference architectures. **Annals of Software Engineering**, Springer, v. 5, n. 1, p. 143–168, 1998.

KÄSTNER, C.; RHEIN, A. V.; ERDWEG, S.; PUSCH, J.; APEL, S.; RENDEL, T.; OSTERMANN, K. Toward variability-aware testing. In: ACM. **Proceedings of the 4th International Workshop on Feature-Oriented Software Development**. [S.l.], 2012. p. 1–8.

KIM, C. H. P.; KHURSHID, S.; BATORY, D. Shared execution for efficiently testing product lines. In: IEEE. **2012 IEEE 23rd International Symposium on Software Reliability Engineering**. [S.l.], 2012. p. 221–230.

KIM, M.; ZIMMERMANN, T.; DELINE, R.; BEGEL, A. The emerging role of data scientists on software development teams. In: ACM. **Proceedings of the 38th International Conference on Software Engineering**. [S.l.], 2016. p. 96–107.

KIM, T.; KO, I. Y.; KANG, S. W.; LEE, D. H. Extending atam to assess product line architecture. In: IEEE. **Computer and Information Technology. CIT 2008. 8th IEEE International Conference on**. [S.l.], 2008. p. 790–797.

KITCHENHAM, B.; AL-KHILIDAR, H.; BABAR, M. A.; BERRY, M.; COX, K.; KEUNG, J.; KURNIAWATI, F.; STAPLES, M.; ZHANG, H.; ZHU, L. Evaluating guidelines for reporting empirical software engineering studies. **Empirical Software Engineering**, Springer, v. 13, n. 1, p. 97–121, 2008.

KITCHENHAM, B.; PFLEEGER, S. L.; FENTON, N. Towards a framework for software measurement validation. **Software Engineering, IEEE Transactions on**, IEEE, v. 21, n. 12, p. 929–944, 1995.

KITCHENHAM, B. A.; CHARTERS, S. **Guidelines for performing systematic literature reviews in software engineering**. [S.l.], 2007.

KITCHENHAM, B. A.; WALKER, J. G. A quantitative approach to monitoring software development. **Software Engineering Journal**, IET, v. 4, n. 1, p. 2–13, 1989.

KRAEMER, H. C. Kappa coefficient. **Wiley StatsRef: Statistics Reference Online**, Wiley Online Library, 1982.

KUNCHEVA, L. I.; KRISHNAPURAM, R. A fuzzy consensus aggregation operator. **Fuzzy Sets and Systems**, Elsevier, v. 79, n. 3, p. 347–356, 1996.

LEE, J.; KOTONYA, G.; ROBINSON, D. Engineering service-based dynamic software product lines. **Computer**, IEEE Computer Society, v. 45, n. 10, p. 0049–55, 2012.

LEE, J.; MUTHIG, D. Feature-oriented variability management in product line engineering. **Communications of the ACM**, ACM, v. 49, n. 12, p. 55–59, 2006.

LEE, K.; KANG, K. C. Usage context as key driver for feature selection. In: **Software Product Lines: Going Beyond**. [S.l.]: Springer, 2010. p. 32–46.

LIMA, F. F.; ROCHA, L. S.; MAIA, P. H.; ANDRADE, R. M. A decoupled and interoperable architecture for coordination in ubiquitous systems. In: IEEE. **Software Components, Architectures and Reuse (SBCARS), 2011 Fifth Brazilian Symposium on**. [S.l.], 2011. p. 31–40.

LINDEN, F. J. Van der; SCHMID, K.; ROMMES, E. **Software product lines in action: the best industrial practice in product line engineering**. [S.l.]: Springer Science & Business Media, 2007.

LOCHAU, M.; BÜRDEK, J.; HÖLZLE, S.; SCHÜRR, A. Specification and automated validation of staged reconfiguration processes for dynamic software product lines. **Software & Systems Modeling**, Springer, p. 1–28, 2015.

LOTUFO, R.; SHE, S.; BERGER, T.; CZARNECKI, K.; WĄSOWSKI, A. Evolution of the linux kernel variability model. In: **Software Product Lines: Going Beyond**. [S.l.]: Springer, 2010. p. 136–150.

MAIA, M. E.; FONTELES, A.; NETO, B.; GADELHA, R.; VIANA, W.; ANDRADE, R. Loccam-loosely coupled context acquisition middleware. In: ACM. **Proceedings of the 28th Annual ACM Symposium on Applied Computing**. [S.l.], 2013. p. 534–541.

MAIA, M. E.; ROCHA, L. S.; ANDRADE, R. Requirements and challenges for building service-oriented pervasive middleware. In: ACM. **Proceedings of the 2009 international conference on Pervasive services**. [S.l.], 2009. p. 93–102.

MARINHO, F. G.; ANDRADE, R. M. *et al.* A verification mechanism of feature models for mobile and context-aware software product lines. In: IEEE. **Software Components, Architectures and Reuse (SBCARS), 2011 Fifth Brazilian Symposium on**. [S.l.], 2011. p. 1–10.

MARINHO, F. G.; ANDRADE, R. M.; WERNER, C.; VIANA, W.; MAIA, M. E.; ROCHA, L. S.; TEIXEIRA, E.; FILHO, J. B. F.; DANTAS, V. L.; LIMA, F. *et al.* Mobiline: A nested software product line for the domain of mobile and context-aware applications. **Science of Computer Programming**, Elsevier, v. 78, n. 12, p. 2381–2398, 2013.

MARINHO, F. G.; MAIA, P. H.; ANDRADE, R.; VIDAL, V. M.; COSTA, P. A.; WERNER, C. Safe adaptation in context-aware feature models. In: ACM. **Proceedings of the 4th International Workshop on Feature-Oriented Software Development**. [S.l.], 2012. p. 54–61.

MASSEN, T. von der; LICHTER, H. Deficiencies in feature models. In: **Workshop on Software Variability Management for Product Derivation-Towards Tool Support**. [S.l.: s.n.], 2004. p. 46.

MATINLASSI, M.; NIEMELÄ, E.; DOBRICA, L. Quality-driven architecture design and quality analysis method. **VTT PUBLICATIONS**, v. 4, n. 5, p. 6, 2002.

MAZO, R.; MUÑOZ-FERNÁNDEZ, J. C.; RINCÓN, L.; SALINESI, C.; TAMURA, G. Variamos: an extensible tool for engineering (dynamic) product lines. In: ACM. **Proceedings of the 19th International Conference on Software Product Line**. [S.l.], 2015. p. 374–379.

MCCALL, J. A.; RICHARDS, P. K.; WALTERS, G. F. **Factors in software quality. volume i. concepts and definitions of software quality**. [S.l.], 1977.

MCCRUM-GARDNER, E. Which is the correct statistical test to use? **British Journal of Oral and Maxillofacial Surgery**, Elsevier, v. 46, n. 1, p. 38–41, 2008.

MENDONCA, M.; BRANCO, M.; COWAN, D. S.p.l.o.t.: Software product lines online tools. In: **Proceedings of the 24th ACM SIGPLAN Conference Companion on Object Oriented Programming Systems Languages and Applications**. New York, NY, USA: ACM, 2009. (OOPSLA '09), p. 761–762. ISBN 978-1-60558-768-4.

MENDONCA, M.; WĄSOWSKI, A.; CZARNECKI, K. Sat-based analysis of feature models is easy. In: CARNEGIE MELLON UNIVERSITY. **Proceedings of the 13th International Software Product Line Conference**. [S.l.], 2009. p. 231–240.

MENDONCA, M.; WASOWSKI, A.; CZARNECKI, K.; COWAN, D. Efficient compilation techniques for large scale feature models. In: ACM. **Proceedings of the 7th international conference on Generative programming and component engineering**. [S.l.], 2008. p. 13–22.

MENS, K.; CAPILLA, R.; CARDOZO, N.; DUMAS, B. A taxonomy of context-aware software variability approaches. In: ACM. **Companion Proceedings of the 15th International Conference on Modularity**. [S.l.], 2016. p. 119–124.

MENZIES, T.; ZIMMERMANN, T. Software analytics: so what? **IEEE Software**, IEEE, v. 30, n. 4, p. 31–37, 2013.

MIZOUNI, R.; MATAR, M. A.; MAHMOUD, Z. A.; ALZAHMI, S.; SALAH, A. A framework for context-aware self-adaptive mobile applications spl. **Expert Systems with applications**, Elsevier, v. 41, n. 16, p. 7549–7564, 2014.

MOENS, H.; TURCK, F. D. Feature-based application development and management of multi-tenant applications in clouds. In: ACM. **Proceedings of the 18th International Software Product Line Conference-Volume 1**. [S.l.], 2014. p. 72–81.

MONTAGUD, S.; ABRAHÃO, S.; INSFRAN, E. A systematic review of quality attributes and measures for software product lines. **Software Quality Journal**, Springer, v. 20, n. 3-4, p. 425–486, 2012.

MONTAGUD, S.; ABRAHãO, S. Gathering current knowledge about quality evaluation in software product lines. In: **Proceedings of the 13th International Software Product Line Conference**. Pittsburgh, PA, USA: Carnegie Mellon University, 2009. (SPLC '09), p. 91–100.

MURGUZUR, A.; CAPILLA, R.; TRUJILLO, S.; ORTIZ, Ó.; LOPEZ-HERREJON, R. E. Context variability modeling for runtime configuration of service-based dynamic software product lines. In: ACM. **Proceedings of the 18th International Software Product Line Conference: Companion Volume for Workshops, Demonstrations and Tools-Volume 2**. [S.l.], 2014. p. 2–9.

NASCIMENTO, A. S.; RUBIRA, C. M. F.; LEE, J. An spl approach for adaptive fault tolerance in soa. In: ACM. **Proceedings of the 15th International Software Product Line Conference, Volume 2**. [S.l.], 2011. p. 15.

NGUYEN, H. V.; KÄSTNER, C.; NGUYEN, T. N. Exploring variability-aware execution for testing plugin-based web applications. In: ACM. **Proceedings of the 36th International Conference on Software Engineering**. [S.l.], 2014. p. 907–918.

OLIINYK, O.; PETERSEN, K.; SCHOELZKE, M.; BECKER, M.; SCHNEICKERT, S. Metrics for the evaluation of feature models in an industrial context: A case study at opel. In: SPRINGER. **International Working Conference on Requirements Engineering: Foundation for Software Quality**. [S.l.], 2015. p. 33–48.

OLIVEIRA, R. P. de; SANTOS, A. R.; ALMEIDA, E. S. de; GOMES, G. S. da S. Evaluating lehman's laws of software evolution within software product lines industrial projects. **Journal of Systems and Software**, Elsevier, v. 131, p. 347–365, 2017.

OLUMOFIN, F. G.; MIŠIĆ, V. B. A holistic architecture assessment method for software product lines. **Information and Software Technology**, Elsevier, v. 49, n. 4, p. 309–323, 2007.

ORTU, M.; DESTEFANIS, G.; ADAMS, B.; MURGIA, A.; MARCHESI, M.; TONELLI, R. The jira repository dataset: Understanding social aspects of software development. In: ACM. **Proceedings of the 11th International Conference on Predictive Models and Data Analytics in Software Engineering**. [S.l.], 2015. p. 1.

PARRA, C.; BLANC, X.; DUCHIEN, L. Context awareness for dynamic service-oriented product lines. In: CARNEGIE MELLON UNIVERSITY. **Proceedings of the 13th International Software Product Line Conference**. [S.l.], 2009. p. 131–140.

PARRA, C.; ROMERO, D.; MOSSER, S.; ROUVOY, R.; DUCHIEN, L.; SEINTURIER, L. Using constraint-based optimization and variability to support continuous self-adaptation. In: ACM. **Proceedings of the 27th Annual ACM Symposium on Applied Computing**. [S.l.], 2012. p. 486–491.

PASCUAL, G. G.; LOPEZ-HERREJON, R. E.; PINTO, M.; FUENTES, L.; EGYED, A. Applying multiobjective evolutionary algorithms to dynamic software product lines for reconfiguring mobile applications. **Journal of Systems and Software**, Elsevier, v. 103, p. 392–411, 2015.

PASSOS, L.; CZARNECKI, K. A dataset of feature additions and feature removals from the linux kernel. In: ACM. **Proceedings of the 11th Working Conference on Mining Software Repositories**. [S.l.], 2014. p. 376–379.

PASSOS, L.; TEIXEIRA, L.; DINTZNER, N.; APEL, S.; WĄSOWSKI, A.; CZARNECKI, K.; BORBA, P.; GUO, J. Coevolution of variability models and related software artifacts. **Empirical Software Engineering**, Springer, p. 1–50, 2015.

PATZKE, T.; BECKER, M.; STEFFENS, M.; SIERSZECKI, K.; SAVOLAINEN, J. E.; FOGDAL, T. Identifying improvement potential in evolving product line infrastructures: 3 case studies. In: ACM. **Proceedings of the 16th International Software Product Line Conference-Volume 1**. [S.l.], 2012. p. 239–248.

PETERSEN, K.; FELDT, R.; MUJTABA, S.; MATTSSON, M. Systematic mapping studies in software engineering. In: SN. **12th international conference on evaluation and assessment in software engineering**. [S.l.], 2008. v. 17, n. 1, p. 1–10.

PETTERSEN, E. F.; GODDARD, T. D.; HUANG, C. C.; COUCH, G. S.; GREENBLATT, D. M.; MENG, E. C.; FERRIN, T. E. Ucsf chimera—a visualization system for exploratory research and analysis. **Journal of computational chemistry**, Wiley Online Library, v. 25, n. 13, p. 1605–1612, 2004.

PIZZI, N. J. A fuzzy classifier approach to estimating software quality. **Information Sciences**, Elsevier, v. 241, p. 1–11, 2013.

POELS, G.; DEDENE, G. Distance-based software measurement: necessary and sufficient properties for software measures. **Information and Software Technology**, Elsevier, v. 42, n. 1, p. 35–46, 2000.

POHL, K.; BÖCKLE, G.; LINDEN, F. J. van D. **Software product line engineering: foundations, principles and techniques**. [S.l.]: Springer Science & Business Media, 2005.

PRATAP, A.; CHAUDHARY, R.; YADAV, K. Estimation of software maintainability using fuzzy logic technique. In: IEEE. **Issues and Challenges in Intelligent Computing Techniques (ICICT), 2014 International Conference on**. [S.l.], 2014. p. 486–492.

PUSSINEN, M. **A survey on software product-line evolution**. [S.l.]: Tampere University of Technology, 2002.

QUINTON, C.; RABISER, R.; VIERHAUSER, M.; GRÜNBACHER, P.; BARESI, L. Evolution in dynamic software product lines: challenges and perspectives. In: ACM. **Proceedings of the 19th International Conference on Software Product Line**. [S.l.], 2015. p. 126–130.

REINHARTZ-BERGER, I. When aspect-orientation meets software product line engineering. In: **Domain Engineering**. [S.l.]: Springer, 2013. p. 83–111.

ROBSON, C. Real world research. 2nd. **Edition. Blackwell Publishing. Malden**, 2002.

ROBSON, C.; MCCARTAN, K. **Real world research**. [S.l.]: Wiley, 2016.

ROCHA, A. d.; SOUZA, G. d. S.; BARCELLOS, M. Medição de software e controle estatístico de processos. **Ministério da Ciência, Tecnologia e Inovação; Secretaria de Política de Informática**, p. 21, 2012.

ROMERO, D.; URLI, S.; QUINTON, C.; BLAY-FORNARINO, M.; COLLET, P.; DUCHIEN, L.; MOSSER, S. Splemma: a generic framework for controlled-evolution of software product lines. In: ACM. **Proceedings of the 17th International Software Product Line Conference co-located workshops**. [S.l.], 2013. p. 59–66.

ROSS, T. J. **Fuzzy logic with engineering applications**. [S.l.]: John Wiley & Sons, 2009.

RUNESON P. E HÖST, M. Guidelines for conducting and reporting case study research in software engineering. **Empirical software engineering**, Springer, v. 14, n. 2, p. 131–164, 2009.

SALKIND, N. J.; RAINWATER, T. **Exploring research**. [S.l.]: Prentice Hall Upper Saddle River, NJ, 2003.

SALLER, K.; LOCHAU, M.; REIMUND, I. Context-aware dspls: model-based runtime adaptation for resource-constrained systems. In: ACM. **Proceedings of the 17th International Software Product Line Conference co-located workshops**. [S.l.], 2013. p. 106–113.

SANCHEZ, L. E.; DIAZ-PACE, J. A.; ZUNINO, A.; MOISAN, S.; RIGAULT, J.-P. An approach based on feature models and quality criteria for adapting component-based systems. **Journal of Software Engineering Research and Development**, Springer, v. 3, n. 1, p. 1–30, 2015.

SANTOS, I. d. S. **Um ambiente para geração de cenários de testes para linhas de produtos de software sensíveis ao contexto**. Dissertação (Mestrado) — Dissertação de Mestrado. Universidade Federal do Ceará, 2013.

SANTOS, I. S.; ANDRADE, R. M.; NETO, P. A. S. Templates for textual use cases of software product lines: results from a systematic mapping study and a controlled experiment. **Journal of Software Engineering Research and Development**, Springer Berlin Heidelberg, v. 3, n. 1, p. 1, 2015.

SCHILIT, B.; ADAMS, N.; WANT, R. Context-aware computing applications. In: IEEE. **Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on**. [S.l.], 1994. p. 85–90.

SCHRÖTER, R.; KRIETER, S.; THÜM, T.; BENDUHN, F.; SAAKE, G. Feature-model interfaces: the highway to compositional analyses of highly-configurable systems. In: IEEE. **Software Engineering (ICSE), 2016 IEEE/ACM 38th International Conference on**. [S.l.], 2016. p. 667–678.

SHLENS, J. A tutorial on principal component analysis. **arXiv preprint arXiv:1404.1100**, 2014.

SINNEMA, M.; DEELSTRA, S.; NIJHUIS, J.; BOSCH, J. Covamof: A framework for modeling variability in software product families. In: SPRINGER. **International Conference on Software Product Lines**. [S.l.], 2004. p. 197–213.

SOLINGEN, V. R.; BASILI, V.; CALDIERA, G.; ROMBACH, H. D. Goal question metric (gqm) approach. **Encyclopedia of software engineering**, Wiley Online Library, 2002.

SPINELLIS, D. A repository with 44 years of unix evolution. In: IEEE PRESS. **Proceedings of the 12th Working Conference on Mining Software Repositories**. [S.l.], 2015. p. 462–465.

ŠTUIKYS, V.; DAMAŠEVICIUS, R. Measuring complexity of domain models represented by feature diagrams. **Information Technology and Control**, v. 38, n. 3, p. 179–187, 2009.

SVAHNBERG, M.; GURP, J. V.; BOSCH, J. A taxonomy of variability realization techniques. **Software: Practice and Experience**, Wiley Online Library, v. 35, n. 8, p. 705–754, 2005.

THOMMAZO, A. D.; RIBEIRO, T.; OLIVATTO, G.; WERNECK, V.; FABBRI, S. An automatic approach to detect traceability links using fuzzy logic. In: **Software Engineering (SBES), 2013 27th Brazilian Symposium on**. [S.l.: s.n.], 2013. p. 21–30.

THÖRN, C. A quality model for evaluating feature models. In: **Software Product Lines, 11th International Conference, SPLC 2007, Kyoto, Japan, September 10-14, 2007, Proceedings. Second Volume (Workshops)**. [S.l.: s.n.], 2007. p. 184–190.

THÖRN, C. On the quality of feature models. Linköping University Electronic Press, 2010.

THÜM, T.; APEL, S.; KÄSTNER, C.; SCHAEFER, I.; SAAKE, G. A classification and survey of analysis strategies for software product lines. **ACM Computing Surveys (CSUR)**, ACM, v. 47, n. 1, p. 6, 2014.

THÜM, T.; BATORY, D.; KÄSTNER, C. Reasoning about edits to feature models. In: IEEE. **Software Engineering, 2009. ICSE 2009. IEEE 31st International Conference on**. [S.l.], 2009. p. 254–264.

THÜM, T.; KÄSTNER, C.; BENDUHN, F.; MEINICKE, J.; SAAKE, G.; LEICH, T. Featureide: An extensible framework for feature-oriented software development. **Science of Computer Programming**, Elsevier, v. 79, p. 70–85, 2014.

TRINIDAD, P.; BENAVIDES, D.; DURÁN, A.; RUIZ-CORTÉS, A.; TORO, M. Automated error analysis for the agilization of feature modeling. **Journal of Systems and Software**, Elsevier, v. 81, n. 6, p. 883–896, 2008.

TSANG, E. **Foundations of Constraint Satisfaction: The Classic Text**. [S.l.]: BoD–Books on Demand, 2014.

VALE, G.; ALBUQUERQUE, D.; FIGUEIREDO, E.; GARCIA, A. Defining metric thresholds for software product lines: a comparative study. In: ACM. **Proceedings of the 19th International Conference on Software Product Line**. [S.l.], 2015. p. 176–185.

VALE G. E FIGUEIREDO, E. A method to derive metric thresholds for software product lines. In: IEEE. **Software Engineering (SBES), 2015 29th Brazilian Symposium on**. [S.l.], 2015. p. 110–119.

WAGNER, S.; LOCHMANN, K.; WINTER, S.; GOEB, A.; KLAES, M. Quality models in practice: A preliminary analysis. In: **Empirical Software Engineering and Measurement, 2009. ESEM 2009. 3rd International Symposium on**. [S.l.: s.n.], 2009. p. 464–467. ISSN 1938-6451.

WANG, L. **Support vector machines: theory and applications**. [S.l.]: Springer Science & Business Media, 2005. v. 177.

WHITE, J.; BENAVIDES, D.; SCHMIDT, D. C.; TRINIDAD, P.; DOUGHERTY, B.; RUIZ-CORTES, A. Automated diagnosis of feature model configurations. **Journal of Systems and Software**, Elsevier, v. 83, n. 7, p. 1094–1107, 2010.

WHITE, J.; GALINDO, J. A.; SAXENA, T.; DOUGHERTY, B.; BENAVIDES, D.; SCHMIDT, D. C. Evolving feature model configurations in software product lines. **Journal of Systems and Software**, Elsevier, v. 87, p. 119–136, 2014.

WIERINGA, R.; MAIDEN, N.; MEAD, N.; ROLLAND, C. Requirements engineering paper classification and evaluation criteria: a proposal and a discussion. **Requirements Engineering**, Springer, v. 11, n. 1, p. 102–107, 2006.

WILLIAMS, L. G.; SMITH, C. U. Pasa sm: a method for the performance assessment of software architectures. In: ACM. **Proceedings of the 3rd international workshop on Software and performance**. [S.l.], 2002. p. 179–189.

WOHLIN, C.; RUNESON, P.; HÖST, M.; OHLSSON, M. C.; REGNELL, B.; WESSLÉN, A. **Experimentation in software engineering**. [S.l.]: Springer Science & Business Media, 2012.

XU, Z.; GAO, K.; KHOSHGOFTAAR, T. M.; SELIYA, N. System regression test planning with a fuzzy expert system. **Information Sciences**, Elsevier, v. 259, p. 532–543, 2014.

YADAV, H. B.; YADAV, D. K. A fuzzy logic based approach for phase-wise software defects prediction using software metrics. **Information and Software Technology**, Elsevier, v. 63, p. 44–57, 2015.

ZHANG, B.; BECKER, M. Mining complex feature correlations from software product line configurations. In: ACM. **Proceedings of the Seventh International Workshop on Variability Modelling of Software-intensive Systems**. [S.l.], 2013. p. 19.

ZHANG, B.; BECKER, M. Recovar: A solution framework towards reverse engineering variability. In: IEEE. **Product Line Approaches in Software Engineering (PLEASE), 2013 4th International Workshop on**. [S.l.], 2013. p. 45–48.

ZHANG, G.; YE, H.; LIN, Y. Quality attributes assessment for feature-based product configuration in software product line. In: IEEE. **Software Engineering Conference (APSEC), 2010 17th Asia Pacific**. [S.l.], 2010. p. 137–146.

ZHANG, G.; YE, H.; LIN, Y. Quality attribute modeling and quality aware product configuration in software product lines. **Software Quality Journal**, Springer, v. 22, n. 3, p. 365–401, 2014.

ZHANG, L.; MADIGAN, C. F.; MOSKEWICZ, M. H.; MALIK, S. Efficient conflict driven learning in a boolean satisfiability solver. In: IEEE PRESS. **Proceedings of the 2001 IEEE/ACM international conference on Computer-aided design**. [S.l.], 2001. p. 279–285.

APÊNDICE A – PROCEDIMENTOS OPERACIONAIS DAS MEDIDAS DO CATÁLOGO COFFEE

Neste apêndice são detalhados os procedimentos operacionais para cada uma das medidas do catálogo COfFEE. A definição dos procedimentos operacionais é importante para auxiliar a avaliação da qualidade dos modelos de *features* pelo Engenheiro de Domínio, tanto em LPSs quanto em LPSDs.

Tabela 35 – Definição operacional da medida número de *features* (NF).

Definição Operacional da Medida	**Descrição**
Nome	Número de *Features*
Definição da Medida	Quantidade total de *features* de um modelo de *features*
Mneumônico	NF
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	Tamanho
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números reais positivos
Intervalo esperado dos dados	$10 \geq NF \leq 88$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#*Features* do modelo de *feature*)
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	Um número de *features* adequado para o modelo de *features* deve estar dentro do intervalo entre 10 *features* e 88 *features*. Um valor de NF abaixo de 10 significa que é um número insuficiente para representar o domínio. Um valor de NF acima de 88 significa que o modelo é muito complexo.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 36 – Definição operacional da medida número de *features* opcionais (NO).

Definição Operacional da Medida	**Descrição**
Nome	Número de *Features* Opcionais
Definição da Medida	Quantidade total de *features* opcionais de um modelo de *features*
Mneumônico	NO
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	Tamanho
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	$NO \leq 40$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#*Features* opcionais do modelo de *features*)
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	Um número de *features* opcionais adequado deve ser menor do que 40. Um valor de NO acima de 40 significa que a variabilidade do modelo é alta.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 37 – Definição operacional da medida número de *features* mandatórias (NM).

Definição Operacional da Medida	**Descrição**
Nome	Número de *Features* Mandatórias
Definição da Medida	Quantidade total de *features* mandatórias (obrigatórias) de um modelo de *features*
Mneumônico	NM
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	Tamanho
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	$NM \leq 41$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#*Features* mandatórias do modelo de *features*)
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	Um número de *features* mandatórias adequado para o modelo de *features* deve ser menor do que 41. Um valor de NM acima de 41 significa que a complexidade do modelo é alta.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 38 – Definição operacional da medida número de *features top* (NTop).

Definição Operacional da Medida	**Descrição**
Nome	Número de *Features Top*
Definição da Medida	Quantidade total de *features* descendentes da raiz da árvore *feature*
Mneumônico	NTop
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	Tamanho
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	$2 \geq NTop \leq 13$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#*Features* descendentes da raiz da árvore do modelo de *features*)
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O modelo de *features* adequado deve ter um valor do NTop entre 2 e 13. Um valor do NTop abaixo de 2 significa que a largura do modelo é muito pequena, e que caso o modelo tenha um alto número de features a complexidade aumenta. Um valor de NTop acima de 13 significa que o modelo é muito largo.
Momento da Análise de Medição	A cada versão do modelo de *feature*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 39 – Definição operacional da medida número de *features* folhas (NLeaf).

Definição Operacional da Medida	**Descrição**
Nome	Número de *Features* Folhas
Definição da Medida	Quantidade total de *features* sem filhos da árvore de um modelo de *features*
Mneumônico	NLeaf
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	Tamanho
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	$7 \geq NLeaf \leq 65$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#*Features* sem filhos do modelo de *features*)
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor de NLeaf deve ser entre 7 e 65, para ter uma largura adequada do modelo de *features*, e consequentemente uma melhor analisabilidade. Caso o valor de Nleaf seja menor que 7 a largura do modelo é baixa. O NLeaf acima de 65 indica que o modelo é muito largo.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 40 – Definição operacional da medida profundidade máxima (DT Max).

Definição Operacional da Medida	**Descrição**
Nome	Profundidade Máxima
Definição da Medida	Número de *features* do caminho mais longo a partir da raiz do modelo de *features*
Mneumônico	DT Max
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	Tamanho
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	$2 \geq DTMax \leq 8$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#*Features* do caminho mais longo a partir da raiz do modelo de *features*)
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor do DTMax deve estar entre 2 e 30 para ter uma profundidade máxima adequada para o modelo de *features*. Um valor abaixo de 2 significa que a profundidade máxima do modelo é muito baixa. O DTMax acima de 30 significa que a profundidade máxima do modelo é muito alta, e consequentemente aumenta sua complexidade.
Momento da Análise de Medição	A cada versão do modelo de *feature*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 41 – Definição operacional da medida profundidade média (DT Mean).

Definição Operacional da Medida	**Descrição**
Nome	Profundidade Média
Definição da Medida	Média de profundidade dos caminhos da árvore de um modelo de *features*
Mneumônico	DT Mean
Tipo da Medida	Medida derivada
Entidade da Medida	Artefato
Propriedade da Medida	Tamanho
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números reais
Intervalo esperado dos dados	$DTMean \leq 4$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#Profundidade dos caminhos da árvore de um modelo de *features*)/Número de caminhos
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor do DTMean deve estar abaixo de 4 para ter uma profundidade média adequada para o modelo de *features*. O DTMean acima de 4 significa que a profundidade média do modelo é muito alta, e consequentemente aumenta sua complexidade.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 42 – Definição operacional da medida profundidade mediana (DT Median).

Definição Operacional da Medida	**Descrição**
Nome	Profundidade Mediana
Definição da Medida	Profundidade do caminho mediano de um modelo de *features*
Mneumônico	DT Median
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	Tamanho
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números reais
Intervalo esperado dos dados	$DTMedian \leq 4$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#*Features* do caminho mediano a partir da raiz do modelo de *feature*
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor do DTMedian deve estar abaixo de 4 para ter uma profundidade mediana adequada para o modelo de *features*. O DTMedian acima de 4 significa que a profundidade mediana do modelo é muito alta, e consequentemente aumenta sua complexidade.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 43 – Definição operacional da medida complexidade cognitiva (CogC).

Definição Operacional da Medida	**Descrição**
Nome	Complexidade Cognitiva
Definição da Medida	Quantidade total de pontos de variação de um modelo de *features*
Mneumônico	CogC
Tipo da Medida	Medida derivada
Entidade da Medida	Artefato
Propriedade da Medida	Tamanho
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	$CogC \leq 16$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#Pontos de variação Or) + $\sum$(#Pontos de variação XOr)
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	OO valor da complexidade cognitiva deve ser menor do que 16 para ter uma boa entendabilidade do modelo de *features*. Um valor do CogC acima de 16 signfica que a entendibilidade do modelo é baixa.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 44 – Definição operacional da medida extensibilidade da *feature* (FEX).

Definição Operacional da Medida	**Descrição**
Nome	Extensibilidade da *Feature*
Definição da Medida	Quantidade total de *features* folhas e de *features* filhas de pontos de variação de um modelo de *feature*
Mneumônico	FEX
Tipo da Medida	Medida derivada
Entidade da Medida	Artefato
Propriedade da Medida	Complexidade
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	$7 \geq FEX \leq 67$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	NLeaf + SCDF + MCDF
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor do FEX deve ficar entre 7 e 67 para ter uma extensibilidade adequada. O FEX acima de 67 significa que a extensibilidade do modelo é baixa.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 45 – Definição operacional da medida flexibilidade da configuração (FoC).

Definição Operacional da Medida	**Descrição**
Nome	Flexibilidade da Configuração
Definição da Medida	Taxa de flexibilidade do modelo dada pelas *features* opcionais em relação a quantidade total de *features* de um modelo de *features*
Mneumônico	FoC
Tipo da Medida	Medida derivada
Entidade da Medida	Artefato
Propriedade da Medida	Complexidade
Unidade da Medida	Numeral
Tipo de Escala	Escala taxa
Valores da Escala	Números reais
Intervalo esperado dos dados	$FoC \leq 0.7$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	NO/NF
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor de FoC deve ser abaixo de 0.7 para possuir uma flexibilidade adequada. Um valor de FoC acima de 0.7 sigfica que o modelo é pouco flexível.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 46 – Definição operacional da medida *features* dependentes cíclicas únicas (SCDF).

Definição Operacional da Medida	**Descrição**
Nome	*Features* Dependentes Cíclicas Únicas
Definição da Medida	Quantidade de *features* participantes de restrições e que são filhas de pontos de variação com cardinalidade [1..1]
Mneumônico	SCDF
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	Complexidade
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	$SCDF \leq 8$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#Número de *features* filhas de pontos de variação com cardinalidade [1..1])
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor adequado da medida SCDF é abaixo de 8. O valor do SCDF acima de 8 significa que a modularidade do modelo é baixa.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 47 – Definição operacional da medida *features* dependentes cíclicas múltiplas (MCDF).

Definição Operacional da Medida	**Descrição**
Nome	*Features* Dependentes Cíclicas Múltiplas
Definição da Medida	Quantidade de *features* participantes de restrições e que são filhas de pontos de variação com cardinalidade [1..*] *features*
Mneumônico	MCDF
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	Complexidade
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	$MCDF \leq 10$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#Número de *features* filhas de pontos de variação com cardinalidade [1..*])
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor adequado da medida MCDF é abaixo de 10. O valor do MCDF acima de 10 significa que a modularidade do modelo é baixa.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 48 – Definição operacional da medida complexidade ciclomática (CyC).

Definição Operacional da Medida	**Descrição**
Nome	Complexidade Ciclomática
Definição da Medida	Quantidade de restrições de integridade
Mneumônico	CyC
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	Complexidade
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	$CyC \leq 20$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(# Restrições de integridade do modelo de *features*)
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor adequado da medida CyC é abaixo de 20. O valor do CyC acima de 20 significa que a complexidade do modelo é alta.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 49 – Definição operacional da medida complexidade composta (ComC).

Definição Operacional da Medida	**Descrição**
Nome	Complexidade Composta
Definição da Medida	Complexidade do modelo de *features* de acordo com as relações entre as *features*
Mneumônico	ComC
Tipo da Medida	Medida derivada
Entidade da Medida	Artefato
Propriedade da Medida	Complexidade
Unidade da Medida	Numeral
Tipo de Escala	Escala taxa
Valores da Escala	Números reais
Intervalo esperado dos dados	$113.444 \geq ComC \leq 8972.667$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum(NF^2 + (NM^2 + 2 * NOr^2 + 3 * NXOr^2 + 3 * NGF^2 + 3 * R^2)/9$ $R = NGF + CyC)$
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O Valor do ComC deve ser entre 113.444 e 8972.667 para ter uma complexidade composta adequada. O valor do ComC acima de 8972.667 significa que a complexidade do modelo é muito alta.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 50 – Definição operacional da medida número de agrupamento de *features* (NGF).

Definição Operacional da Medida	**Descrição**
Nome	Número de Agrupamento de *Features*
Definição da Medida	Número de *features* que possuem algum filho.
Mneumônico	NGF
Tipo da Medida	Medida derivada
Entidade da Medida	Artefato
Propriedade da Medida	Tamanho
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	$2 \geq NGF \leq 28$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	NF - NLeaf
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O Valor do NGF deve ser entre 2 e 28 para ter um número de agrupamentos adequado. O valor do NGF acima de 28 significa que a complexidade do modelo é muito alta.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 51 – Definição operacional da medida restrições variáveis de *cross-tree* (CTCV).

Definição Operacional da Medida	**Descrição**
Nome	Restrições Variáveis de *Cross-Tree*
Definição da Medida	Número de variáveis distintas em restrições *cross-tree*
Mneumônico	CTCV
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	Complexidade
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	$CTCV \leq 28$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#Restrições *cross-tree* do modelo de *features*)
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor adequado do CTCV deve ser abaixo de 28 para ter número de restrições variáveis adequado. O valor do CTCV acima de 28 significa que a complexidade do modelo é alta.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 52 – Definição operacional da medida taxa de restrições *cross-tree* (CTCR).

Definição Operacional da Medida	**Descrição**
Nome	Taxa de Restrições *Cross-Tree*
Definição da Medida	Taxa do número de *features* envolvidas em restrições de integridade em relação ao número de *features* total do modelo de *feature*
Mneumônico	CTCR
Tipo da Medida	Medida derivada
Entidade da Medida	Artefato
Propriedade da Medida	Complexidade
Unidade da Medida	Numeral
Tipo de Escala	Escala taxa
Valores da Escala	Números reais
Intervalo esperado dos dados	$CTC \leq 0.66$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	NFRI/NF NFRI = Número de *features* envolvidas em restrições de integridade
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor da medida CTC deve ser abaixo de 0.66 para possuir uma complexidade adequada do modelo de *features*. Um valor de CTC acima de 0.66 significa que o modelo possui complexidade alta.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 53 – Definição operacional da medida taxa de conectividade do grafo (RCon).

Definição Operacional da Medida	**Descrição**
Nome	Taxa de Conectividade do Grafo
Definição da Medida	Taxa de *features* que referenciam outras *features* (exceto pais) em suas restrições
Mneumônico	RCon
Tipo da Medida	Medida derivada
Entidade da Medida	Artefato
Propriedade da Medida	Complexidade
Unidade da Medida	Numeral
Tipo de Escala	Escala taxa
Valores da Escala	Números reais
Intervalo esperado dos dados	$RCon \leq 0.39$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#*Features* que referenciam outras *features* em suas restrições)/NF
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor da medida RCOn deve ser abaixo de 0.39 para possuir uma taxa de conectividade do grafo adequada do modelo de features. Um valor de RCon acima de 0.39 significa que o modelo possui complexidade alta.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 54 – Definição operacional da medida densidade do grafo (RDen).

Definição Operacional da Medida	**Descrição**
Nome	Densidade do grafo
Definição da Medida	Média do número de *features* não-pais que são referenciadas em restrições
Mneumônico	RDen
Tipo da Medida	Medida derivada
Entidade da Medida	Artefato
Propriedade da Medida	Complexidade
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números reais
Intervalo esperado dos dados	$RDen \leq 2$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#*Features* que referenciam outras *features* em suas restrições)/CyC)
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor da medida RDen deve ser abaixo de 2 para possuir uma densidade do grafo adequada para o modelo de features. Um valor de RDen acima de 2 significa que o modelo possui complexidade alta.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 55 – Definição operacional da medida coeficiente de densidade de conectividade (CoC).

Definição Operacional da Medida	**Descrição**
Nome	Coeficiente de Densidade de Conectividade
Definição da Medida	Taxa de conectividade em relação ao número de *features*
Mneumônico	CoC
Tipo da Medida	Medida derivada
Entidade da Medida	Artefato
Propriedade da Medida	Complexidade
Unidade da Medida	Numeral
Tipo de Escala	Escala taxa
Valores da Escala	Números reais
Intervalo esperado dos dados	$0.9 \geq CoC \leq 0.99$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#Arestas do modelo de *features*)/NF
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor da medida CoC deve estar entre 0.9 e 0.99 para ter uma taxa de conectividade adequada para o modelo de *features*. Um valor abaixo de 0.9 significa que a complexidade do modelo é muito baixa. O CoC acima de 0.99 significa que a complexidade do modelo é muito alta.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 56 – Definição operacional da medida *features* variáveis (NVF).

Definição Operacional da Medida	**Descrição**
Nome	*Features* Variáveis
Definição da Medida	Quantidade total de *features* que não possuem estado fixo
Mneumônico	NVF
Tipo da Medida	Medida derivada
Entidade da Medida	Artefato
Propriedade da Medida	Tamanho
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	$1 \geq NVF \leq 76$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	SHoF + MHoF + NO
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor da medida NVF deve ser entre 1 e 76, para ter um número de *features* variáveis adequada do modelo de *features*. O NVF acima de 76 indica que o modelo possui uma alta variabilidade.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 57 – Definição operacional da medida *single hotspot feature* (SHoF).

Definição Operacional da Medida	**Descrição**
Nome	*Single Hotspot Feature*
Definição da Medida	Número de *features* filhas de pontos de variação com cardinalidade [1..1]
Mneumônico	SHoF
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	Tamanho
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	$SHoF \leq 29$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#*Features* filhas de pontos de variação com cardinalidade [1..1] do modelo de *features*)
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor da medida SHoF deve ser menor do que 29 para ter uma variabilidade estática adequada do modelo de *features*. Um valor do SHoF acima de 29 significa que a variabilidade estática do modelo é muito alta.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 58 – Definição operacional da medida *multiple hotspot feature* (MHoF).

Definição Operacional da Medida	**Descrição**
Nome	*Multiple Hotspot Feature*
Definição da Medida	Número de *features* filhas de pontos de variação com cardinalidade [1..*]
Mneumônico	MHoF
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	Tamanho
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	MHoF <= 37
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#*Features* filhas de pontos de variação com cardinalidade [1..*] do modelo de *features*)
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor da medida MHoF deve ser menor do que 37 para ter uma variabilidade estática adequada do modelo de *features*. Um valor do MHoF acima de 37 significa que a variabilidade estática do modelo é muito alta.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 59 – Definição operacional da medida *rigid nohotspot feature* (RNoF).

Definição Operacional da Medida	**Descrição**
Nome	*Rigid Nohotspot Feature*
Definição da Medida	Número de *features* não filhas de pontos de variação
Mneumônico	RNoF
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	Tamanho
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	$5 \geq RNoF \leq 58$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#*Features* não filhas de pontos de variação do modelo de *features*)
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor da medida RNoF deve estar entre 5 e 58 para ter variabilidade estática adequada para o modelo de *features*. Um valor abaixo de 5 significa que a variabilidade estática do modelo é muito baixa. O RNoF acima de 58 significa que a variabilidade estática do modelo é muito alta.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 60 – Definição operacional da medida taxa de variabilidade (RoV).

Definição Operacional da Medida	**Descrição**
Nome	Taxa de Variabilidade
Definição da Medida	Média do número de filhos das *features*
Mneumônico	RoV
Tipo da Medida	Medida derivada
Entidade da Medida	Artefato
Propriedade da Medida	Tamanho
Unidade da Medida	Numeral
Tipo de Escala	Escala taxa
Valores da Escala	Números reais
Intervalo esperado dos dados	$1.666667 \geq RoV \leq 4.857143$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#*Features* filhas de todos os nós do modelo de *features*)/(NF-NLeaf)
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor da medida RoV deve estar entre 1.666667 e 4.857143 para ter variabilidade estática adequada para o modelo de *features*. Um valor abaixo de 1.666667 significa que a variabilidade estática do modelo é muito baixa. O RoV acima de 4.857143 significa que a variabilidade estática do modelo é muito alta.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 61 – Definição operacional da medida número de configurações válidas (NVC).

Definição Operacional da Medida	**Descrição**
Nome	Número de Configurações Válidas
Definição da Medida	Número de possíveis configurações válidas do modelo de *features*
Mneumônico	NVC
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	tamanho
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	$2 \geq NVC \leq 1.65542e+13$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#Configurações válidas do modelo de *features*)
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor da medida NVC deve estar entre 2 e 1.65542e+13 para ter variabilidade estática adequada para o modelo de *features*. Um valor abaixo de 2 significa que a variabilidade estática do modelo é muito baixa. O NVC acima de 1.65542e+13 significa que a variabilidade estática do modelo é muito alta.
Momento da Análise de Medição (se aplicável)	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 62 – Definição operacional da medida fator de ramificação máximo (BF Max).

Definição Operacional da Medida	**Descrição**
Nome	Fator de Ramificação Máximo
Definição da Medida	Número máximo de filhos por *feature*
Mneumônico	BF Max
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	Tamanho
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	$3 \geq BFMax \leq 19$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	Número máximo de filhos por *feature*
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor da medida BFMax deve estar entre 3 e 19 para ter uma complexidade adequada para o modelo de *features*. Um valor abaixo de 3 significa que a complexidade do modelo é muito baixa. O BFMax acima de 19 significa que a complexidade do modelo é muito alta.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 63 – Definição operacional da medida número de grupos Or (NGOr).

Definição Operacional da Medida	**Descrição**
Nome	Número de Grupos Or
Definição da Medida	Número de pontos de variação com relacionamentos Or
Mneumônico	NGOr
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	Tamanho
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	$NGOr \leq 11$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#Pontos de variação Or)
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor da medida NGOr deve ser menor do que 11 para ter uma variabilidade estática adequada do modelo de *features*. Um valor do NGOr acima de 11 significa que a variabilidade estática do modelo é muito alta.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 64 – Definição operacional da medida número de grupos XOr (NGXOr).

Definição Operacional da Medida	**Descrição**
Nome	Número de Grupos XOr
Definição da Medida	Número de pontos de variação com relacionamentos XOr
Mneumônico	NGXOr
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	Tamanho
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	$NGXOr \leq 11$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#Pontos de variação XOr)
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor da medida NGXOr deve ser menor do que 11 para ter uma variabilidade estática adequada do modelo de *features*. Um valor do NGXOr acima de 11 significa que a variabilidade estática do modelo é muito alta.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 65 – Definição operacional da medida taxa de *features* Or (ROr).

Definição Operacional da Medida	**Descrição**
Nome	Taxa de *Features* Or
Definição da Medida	Taxa de *features* filhas de um ponto de variação Or
Mneumônico	ROr
Tipo da Medida	Medida derivada
Entidade da Medida	Artefato
Propriedade da Medida	Variabilidade
Unidade da Medida	Numeral
Tipo de Escala	Escala taxa
Valores da Escala	Números reais
Intervalo esperado dos dados	$ROr \leq 0.66$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	SHoF/NF
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor da medida ROr deve ser menor do que 0.66 para ter uma variabilidade estática adequada do modelo de *features*. Um valor do ROr acima de 0.66 significa que a variabilidade estática do modelo é muito alta.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 66 – Definição operacional da medida taxa de *features* XOr (RXOr).

Definição Operacional da Medida	**Descrição**
Nome	Taxa de *Features* XOr
Definição da Medida	Taxa de *features* filhas de um ponto de variação XOr
Mneumônico	RXOr
Tipo da Medida	Medida derivada
Entidade da Medida	Artefato
Propriedade da Medida	Variabilidade
Unidade da Medida	Numeral
Tipo de Escala	Escala taxa
Valores da Escala	Números reais
Intervalo esperado dos dados	$RXOr \leq 0.6$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	MHoF/NF
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor da medida RXOr deve ser menor do que 0.6 para ter uma variabilidade estática adequada do modelo de *features*. Um valor do RXOr acima de 0.6 significa que a variabilidade estática do modelo é muito alta.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 67 – Definição operacional da medida número de contextos (NC).

Definição Operacional da Medida	**Descrição**
Nome	Número de Contextos
Definição da Medida	Número de contextos do modelo de *features*
Mneumônico	NC
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	Contexto
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	$CF \leq 3$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#Número de contextos do modelo de *features*)
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor da medida CF adequado para o modelo de *features* de LPSDs é até 3 contextos. O Valor de CF acima de 3 aumenta a variabilidade dinâmica. No entanto, isso depende do tamanho do modelo de *features*.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 68 – Definição operacional da medida número de *features* ativadas (NAF).

Definição Operacional da Medida	**Descrição**
Nome	Número de *Features* Ativadas
Definição da Medida	Número de *features* ativadas de um contexto
Mneumônico	NAF
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	Tamanho
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	$5 \geq NAF \leq 36$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#*Features* ativadas em cada contexto do modelo de *feature*)
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor da medida NAF adequado para o modelo de *features* de LPSDs é entre 5 e 36. No entanto, isso depende do tamanho do modelo de *features* e do número de contextos.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 69 – Definição operacional da medida número de *features* desativadas (NDF).

Definição Operacional da Medida	**Descrição**
Nome	Número de *Features* Desativadas
Definição da Medida	Número de *features* desativadas de um contexto
Mneumônico	NDF
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	Tamanho
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	$1 \geq NFD \leq 22$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#*Features* desativadas em cada contexto do modelo de *feature*)
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor da medida NDF adequado para o modelo de *features* de LPSDs é entre 1 e 22. No entanto, isso depende do tamanho do modelo de *features* e do número de contextos.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 70 – Definição operacional da medida número de restrições de contexto (NCC).

Definição Operacional da Medida	**Descrição**
Nome	Número de Restrições de Contexo
Definição da Medida	Número de restrições de um contexto
Mneumônico	NCC
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	Contexto
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	-
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#Restrições de um contexto)
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	-
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 71 – Definição operacional da medida número de *features* de Contexto (CF).

Definição Operacional da Medida	**Descrição**
Nome	Número de *Features* de Contexto
Definição da Medida	Número de *features* que estão sempre presentes nos contextos ativos do modelo de *features*
Mneumônico	CF
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	Contexto
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	$1 \geq CF \leq 28$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#*Features* que estão sempre presentes nos contextos ativos do modelo de *feature*)
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor da medida CF adequado para o modelo de *features* de LPSDs é entre 1 e 28.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 72 – Definição operacional da medida *features* de contextos em restrições (CFC).

Definição Operacional da Medida	**Descrição**
Nome	*Features* de Contextos em Restrições
Definição da Medida	Número de *features* que estão presentes nos contextos e nas restrições do modelo de *feature*
Mneumônico	CFC
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	Contexto
Unidade da Medida	Numeral
Tipo de Escala	Escala absoluta
Valores da Escala	Números naturais
Intervalo esperado dos dados	$CFC \leq 12$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	$\sum$(#*Features* que estão presentes nos contextos e nas restrições do modelo de *feature*)
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor da medida CFC adequado para o modelo de *features* de LPSDs é até 12.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

Tabela 73 – Definição operacional da medida número de *features* ativadas por contexto (AFCA).

Definição Operacional da Medida	**Descrição**
Nome	Medida Número de *Features* Ativadas por Contexto
Definição da Medida	Taxa do número de *features* ativadas em cada contexto em ralação ao número de contextos
Mneumônico	AFCA
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	Tamanho
Unidade da Medida	Numeral
Tipo de Escala	Escala taxa
Valores da Escala	Números reais
Intervalo esperado dos dados	$4 \geq AFCA \leq 35$
Procedimento de Medição	O modelo de *features* a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	NAF/NC
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor da medida AFCA adequado para o modelo de *features* de LPSDs é entre 4 e 35.
Momento da Análise de Medição	A cada versão do modelo de *features*.
Responsável pela Análise	Engenheiro de Domínio

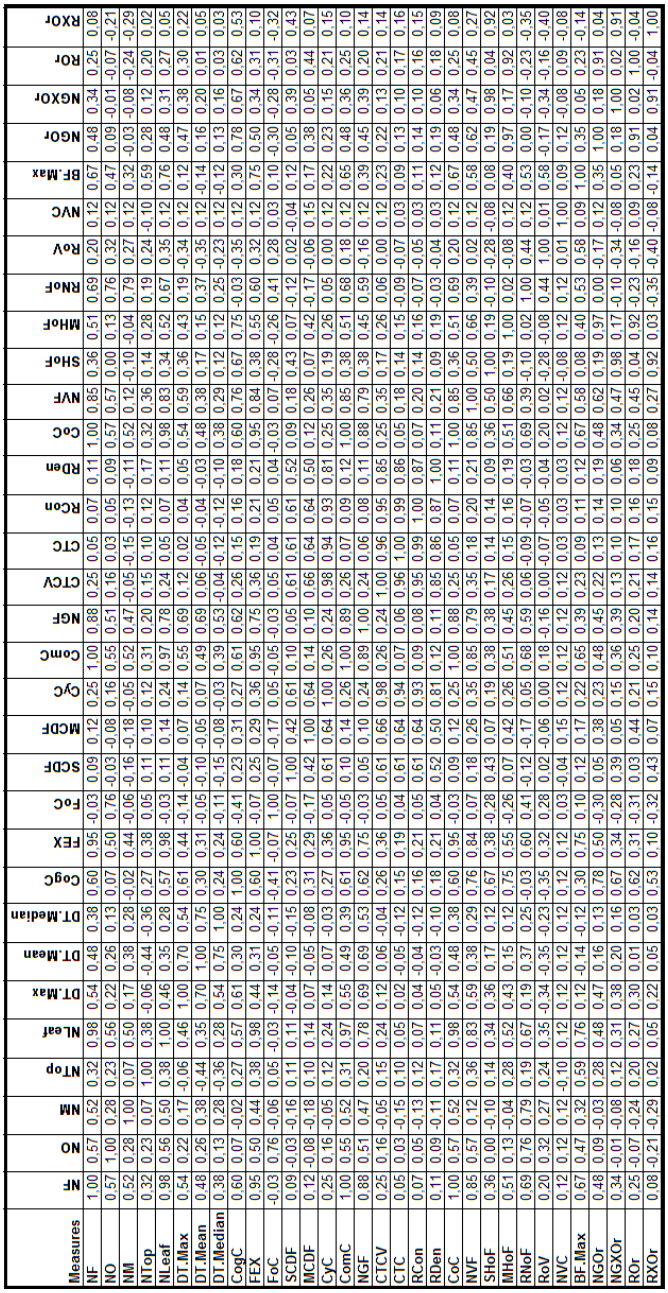
Tabela 74 – Definição operacional da medida número de *features* desativadas por contexto (DFCA).

Definição Operacional da Medida	**Descrição**
Nome	Medida Número de *Features* desativadas por Contexto
Definição da Medida	Taxa do número de *features* desativadas em cada contexto em ralação ao número de contextos
Mneumônico	DFCA
Tipo da Medida	Medida base
Entidade da Medida	Artefato
Propriedade da Medida	Contexto
Unidade da Medida	Numeral
Tipo de Escala	Escala taxa
Valores da Escala	Números reais
Intervalo esperado dos dados	$2 \geq DFCA \leq 17$
Procedimento de Medição	O modelo de feature a ser coletado deve ser importado para ferramenta DyMMer. O engenheiro de domínio pode coletar a medida no visualizador da ferramenta ou exportar para uma planilha Excel.
Fórmula de Cálculo da Medida	NDF/NC
Responsável pela Medição	Engenheiro de Domínio
Momento da Medição	Na elaboração ou na avaliação do modelo de *features*
Periodicidade da Medição	A cada versão do modelo de *features*
Procedimento de Análise	O valor da medida DFCA adequado para o modelo de *features* de LPSDs é entre 2 e 7.
Momento da Análise de Medição	A cada versão do modelo de *feature*.
Responsável pela Análise	Engenheiro de Domínio

APÊNDICE B – MATRIZ DE ANÁLISE DE CORRELAÇÃO

Neste apêndice, são apresentados os resultados da análise de correlação entre as 32 medidas do catálogo COfFEE que são analisadas no estudo de caso exploratório (Capítulo 7).

Tabela 75 – Matriz de análise de correlação.



Measures	NF	NO	NM	NTop	NLeaf	DT.Max	DT.Mean	DT.Median	CogC	FEX	FoC	SCDF	MCDF	CyC	ComC	NGF	CTCV	CTC	RCon	RDen	CoC	NVF	SHoF	MHoF	RNoF	RoV	NVC	BF.Max	NGOr	NGXOr	ROr	RXOr
NF	1.00	0.57	0.52	0.32	0.98	0.54	0.48	0.38	0.60	0.95	-0.03	0.09	0.12	0.25	1.00	0.88	0.25	0.05	0.07	0.11	1.00	0.85	0.36	0.51	0.69	0.20	0.12	0.67	0.48	0.34	0.25	0.08
NO	0.57	1.00	0.28	0.23	0.56	0.22	0.26	0.13	0.07	0.50	0.76	-0.03	-0.08	0.16	0.55	0.51	0.16	0.03	0.05	0.09	0.57	0.57	0.00	0.13	0.76	0.32	0.12	0.47	0.09	-0.01	-0.07	-0.21
NM	0.52	0.28	1.00	0.07	0.50	0.17	0.38	0.28	-0.02	0.44	-0.06	-0.16	-0.05	-0.18	0.52	0.47	-0.05	-0.15	-0.13	-0.11	0.52	0.12	-0.10	-0.04	0.79	0.27	0.12	0.32	-0.03	-0.08	-0.24	-0.29
NTop	0.32	0.23	0.07	1.00	0.38	-0.06	-0.44	-0.36	0.27	0.38	0.05	0.11	0.10	0.12	0.31	0.20	0.15	0.10	0.12	0.17	0.32	0.36	0.14	0.28	0.19	0.24	-0.10	0.59	0.28	0.12	0.20	0.02
NLeaf	0.98	0.56	0.50	0.38	1.00	0.46	0.35	0.28	0.57	0.98	-0.03	0.11	0.14	0.24	0.97	0.78	0.24	0.05	0.07	0.11	0.98	0.83	0.34	0.52	0.67	0.35	0.12	0.76	0.48	0.31	0.27	0.05
DT.Max	0.54	0.22	0.17	-0.06	0.46	1.00	0.70	0.54	0.61	0.44	-0.14	-0.04	0.07	0.14	0.55	0.69	0.12	0.02	0.04	0.05	0.54	0.59	0.36	0.43	0.19	-0.34	0.12	0.12	0.47	0.38	0.30	0.22
DT.Mean	0.48	0.26	0.38	-0.44	0.35	0.70	1.00	0.75	0.30	0.31	-0.05	-0.10	-0.05	0.07	0.49	0.69	0.06	-0.05	-0.04	-0.03	0.48	0.38	0.17	0.15	0.37	-0.35	0.12	-0.14	0.16	0.20	0.01	0.05
DT.Median	0.38	0.13	0.28	-0.36	0.28	0.54	0.75	1.00	0.24	0.24	-0.11	-0.15	-0.08	-0.03	0.39	0.53	-0.04	-0.12	-0.12	-0.10	0.38	0.29	0.12	0.12	0.25	-0.23	0.12	-0.12	0.13	0.16	0.03	0.03
CogC	0.60	0.07	-0.02	0.27	0.57	0.61	0.30	0.24	1.00	0.60	-0.41	0.23	0.31	0.27	0.61	0.62	0.26	0.15	0.16	0.18	0.60	0.76	0.67	0.75	-0.03	-0.35	0.12	0.30	0.78	0.67	0.62	0.53
FEX	0.95	0.50	0.44	0.38	0.98	0.44	0.31	0.24	0.60	1.00	-0.07	0.25	0.29	0.36	0.95	0.75	0.36	0.19	0.21	0.21	0.95	0.84	0.38	0.55	0.60	0.32	0.12	0.75	0.50	0.34	0.31	0.10
FoC	-0.03	0.76	-0.06	0.05	-0.03	-0.14	-0.05	-0.11	-0.41	-0.07	1.00	-0.07	-0.17	0.05	-0.05	-0.03	0.05	0.04	0.05	0.04	-0.03	0.07	-0.28	-0.26	0.41	0.28	0.03	0.10	-0.30	-0.28	-0.31	-0.32
SCDF	0.09	-0.03	-0.16	0.11	0.11	-0.04	-0.10	-0.15	0.23	0.25	-0.07	1.00	0.42	0.61	0.10	0.05	0.61	0.61	0.61	0.52	0.09	0.18	0.43	0.07	-0.12	0.02	-0.04	0.12	0.05	0.39	0.03	0.43
MCDF	0.12	-0.08	-0.18	0.10	0.14	0.07	-0.05	-0.08	0.31	0.29	-0.17	0.42	1.00	0.64	0.14	0.10	0.66	0.64	0.64	0.50	0.12	0.26	0.07	0.42	-0.17	-0.06	0.15	0.17	0.38	0.05	0.44	0.07
CyC	0.25	0.16	-0.05	0.12	0.24	0.14	0.07	-0.03	0.27	0.36	0.05	0.61	0.64	1.00	0.26	0.24	0.98	0.94	0.93	0.81	0.25	0.35	0.19	0.26	0.05	0.00	0.12	0.22	0.23	0.15	0.21	0.15
ComC	1.00	0.55	0.52	0.31	0.97	0.55	0.49	0.39	0.61	0.95	-0.05	0.10	0.14	0.26	1.00	0.89	0.24	0.06	0.09	0.12	1.00	0.85	0.38	0.51	0.68	0.18	0.12	0.65	0.48	0.36	0.25	0.10
NGF	0.88	0.51	0.47	0.20	0.78	0.69	0.69	0.53	0.62	0.75	-0.03	0.05	0.10	0.24	0.89	1.00	0.24	0.06	0.08	0.11	0.88	0.79	0.38	0.45	0.59	-0.16	0.12	0.39	0.45	0.39	0.20	0.14
CTCV	0.25	0.16	-0.05	0.15	0.24	0.12	0.06	-0.04	0.26	0.36	0.05	0.61	0.66	0.98	0.26	0.24	1.00	0.96	0.95	0.85	0.25	0.35	0.17	0.26	0.06	0.00	0.12	0.23	0.22	0.13	0.21	0.14
CTC	0.05	0.03	-0.15	0.10	0.05	0.02	-0.05	-0.12	0.15	0.19	0.04	0.61	0.64	0.94	0.07	0.06	0.96	1.00	0.99	0.86	0.05	0.18	0.14	0.15	-0.09	-0.07	0.03	0.09	0.13	0.10	0.17	0.16
RCon	0.07	0.05	-0.13	0.12	0.07	0.04	-0.04	-0.12	0.16	0.21	0.05	0.61	0.64	0.93	0.09	0.08	0.95	0.99	1.00	0.87	0.07	0.20	0.14	0.16	-0.07	-0.05	0.03	0.11	0.14	0.10	0.16	0.15
RDen	0.11	0.09	-0.11	0.17	0.11	0.05	-0.03	-0.10	0.18	0.21	0.04	0.52	0.50	0.81	0.12	0.11	0.85	0.86	0.87	1.00	0.11	0.21	0.09	0.19	-0.03	-0.04	0.03	0.12	0.19	0.06	0.18	0.09
CoC	1.00	0.57	0.52	0.32	0.98	0.54	0.48	0.38	0.60	0.95	-0.03	0.09	0.12	0.25	1.00	0.88	0.25	0.05	0.07	0.11	1.00	0.85	0.36	0.51	0.69	0.20	0.12	0.67	0.48	0.34	0.25	0.08
NVF	0.85	0.57	0.12	0.36	0.83	0.59	0.38	0.29	0.76	0.84	0.07	0.18	0.26	0.35	0.85	0.79	0.35	0.18	0.20	0.21	0.85	1.00	0.50	0.66	0.39	0.02	0.12	0.58	0.62	0.47	0.45	0.27
SHoF	0.36	0.00	-0.10	0.14	0.34	0.36	0.17	0.12	0.67	0.38	-0.28	0.43	0.07	0.19	0.38	0.38	0.17	0.14	0.14	0.09	0.36	0.50	1.00	0.19	-0.10	-0.28	-0.08	0.08	0.19	0.98	0.04	0.92
MHoF	0.51	0.13	-0.04	0.28	0.52	0.43	0.15	0.12	0.75	0.55	-0.26	0.07	0.42	0.26	0.51	0.45	0.26	0.15	0.16	0.19	0.51	0.66	0.19	1.00	0.02	-0.08	0.12	0.40	0.97	0.17	0.92	0.03
RNoF	0.69	0.76	0.79	0.19	0.67	0.19	0.37	0.25	-0.03	0.60	0.41	-0.12	-0.17	0.05	0.68	0.59	0.06	-0.09	-0.07	-0.03	0.69	0.39	-0.10	0.02	1.00	0.44	0.12	0.53	0.00	-0.10	-0.23	-0.35
RoV	0.20	0.32	0.27	0.24	0.35	-0.34	-0.35	-0.23	-0.35	0.32	0.28	0.02	-0.06	0.00	0.18	-0.16	0.00	-0.07	-0.05	-0.04	0.20	0.02	-0.28	-0.08	0.44	1.00	0.01	0.58	-0.17	-0.34	-0.16	-0.40
NVC	0.12	0.12	0.12	-0.10	0.12	0.12	0.12	0.12	0.12	0.12	0.03	-0.04	0.15	0.12	0.12	0.12	0.12	0.03	0.03	0.03	0.12	0.12	-0.08	0.12	0.12	0.01	1.00	0.09	0.12	-0.08	0.09	-0.08
BF.Max	0.67	0.47	0.32	0.59	0.76	0.12	-0.14	-0.12	0.30	0.75	0.10	0.12	0.17	0.22	0.65	0.39	0.23	0.09	0.11	0.12	0.67	0.58	0.08	0.40	0.53	0.58	0.09	1.00	0.35	0.05	0.23	-0.14
NGOr	0.48	0.09	-0.03	0.28	0.48	0.47	0.16	0.13	0.78	0.50	-0.30	0.05	0.38	0.23	0.48	0.45	0.22	0.13	0.14	0.19	0.48	0.62	0.19	0.97	0.00	-0.17	0.12	0.35	1.00	0.18	0.91	0.04
NGXOr	0.34	-0.01	-0.08	0.12	0.31	0.38	0.20	0.16	0.67	0.34	-0.28	0.39	0.05	0.15	0.36	0.39	0.13	0.10	0.10	0.06	0.34	0.47	0.98	0.17	-0.10	-0.34	-0.08	0.05	0.18	1.00	0.02	0.91
ROr	0.25	-0.07	-0.24	0.20	0.27	0.30	0.01	0.03	0.62	0.31	-0.31	0.03	0.44	0.21	0.25	0.20	0.21	0.17	0.16	0.18	0.25	0.45	0.04	0.92	-0.23	-0.16	0.09	0.23	0.91	0.02	1.00	-0.04
RXOr	0.08	-0.21	-0.29	0.02	0.05	0.22	0.05	0.03	0.53	0.10	-0.32	0.43	0.07	0.15	0.10	0.14	0.14	0.16	0.15	0.09	0.08	0.27	0.92	0.03	-0.35	-0.40	-0.08	-0.14	0.04	0.91	-0.04	1.00

APÊNDICE C – FORMULÁRIO DE PERFIL DOS ESPECIALISTAS

Neste apêndice, é detalhado o formulário do perfil dos especialistas que auxiliaram na revisão por pares do catálogo COfFEE (Capítulo 5).

Perfil do Especialista:

Nome do Especialista:

Instituição:

Estado:

País:

Questões:

1. *Qual é sua posição na instituição?*
   a) Engenheiro de Domínio
   b) Pesquisador
   c) Professor Doutor
   d) Estudante de Doutorado
   Outros:
2. *Qual é o seu nível de formação acadêmica?*
   a) Doutorado
   b) Mestrado
   c) Especialização
   d) Graduação
3. *Quanto tempo de experiência possui em Engenharia de Domínio de Linhas de Produtos de Software?*
   a) Acima de 5 anos
   b) Mais do que 2 e menos de 5 anos
   c) Menos de 2 anos
   d) Nenhum
4. *Quanto tempo de experiência possui em Engenharia de Domínio de Linhas de Produtos de Software Dinâmica?*
   a) Acima de 5 anos
   b) Mais do que 2 e menos de 5 anos
   c) Menos de 2 anos
   d) Nenhum

5. *Como você classifica seu nível de conhecimento em Linha de Produtos de Software?*
   a) Excelente
   b) Alto
   c) Bom
   d) Médio
   e) Baixo
   f) Nenhum
6. *Como você classifica seu nível de conhecimento em Linha de Produtos de Software Dinâmica?*
   a) Excelente
   b) Alto
   c) Bom
   d) Médio
   e) Baixo
   f) Nenhum
7. *Quantos projetos (pesquisa e/ou indústria) envolvendo Linhas de Produtos de Software você participou?*
   a) Acima de 5 anos
   b) Mais do que 2 e menos de 5 anos
   c) Menos de 2 anos
   d) Nenhum
8. *Quantos projetos (pesquisa e/ou indústria) envolvendo Linhas de Produtos de Software Dinâmica você participou?*
   a) Acima de 5 anos
   b) Mais do que 2 e menos de 5 anos
   c) Menos de 2 anos
   d) Nenhum