UNIVERSIDADE ESTADUAL DE MARINGÁ

BRUNO MIGUEL NOGUEIRA DE SOUZA

UMA EXTENSÃO DO MÉTODO OOWS PARA LINHA DE PRODUTO DE SOFTWARE

MARINGÁ

2008

UNIVERSIDADE ESTADUAL DE MARINGÁ

BRUNO MIGUEL NOGUEIRA DE SOUZA

UMA EXTENSÃO DO MÉTODO OOWS PARA LINHA DE PRODUTO DE SOFTWARE

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Estadual de Maringá, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Orientadora: Profa. Dra. Itana Maria de Souza Gimenes.

MARINGÁ

2008

FICHA CATALOGRÁFICA

Souza, Bruno Miguel Nogueira de
S718u Uma extensão do método OOWS para a linha de produto de software / Bruno Miguel Nogueira de Souza. – Maringá, 2008.
108f : il. color., figs., tabs.

Orientadora : Profa. Iatana Maria de Souza Gimenes.
Dissertação (Mestrado) – UEM – Universidade Estadual de Maringá, 2008.

1. Engenharia de Software 2. Web Aplicações desenvolvimento. 3. Linhas e produtos. I. Título.

CDD 21.ed.005.1

UNIVERSIDADE ESTADUAL DE MARINGÁ

BRUNO MIGUEL NOGUEIRA DE SOUZA

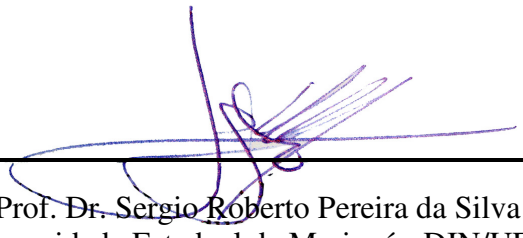
UMA EXTENSÃO DO MÉTODO OOWS PARA LINHA DE PRODUTO DE SOFTWARE

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação da Universidade Estadual de Maringá, como requisito parcial para a obtenção do grau de Mestre em Ciência da Computação.

Aprovado em 05/12/2008

BANCA EXAMINADORA

Profª Drª Itana Maria de Souza Gimenes
Universidade Estadual de Maringá - DIN/UEM



Prof. Dr. Sergio Roberto Pereira da Silva
Universidade Estadual de Maringá - DIN/UEM

Prof. Dr. Marcos Antonio Quináia
Universidade Estadual do Centro-Oeste – DECOMP/UNICENTRO

Dedicatória

A todos que me apoiaram nesta jornada para a conquista de mais um objetivo na minha vida.

Agradecimentos

Agradeço a Deus pela vida e pelas pessoas que fazem parte dela. A minha noiva Roberta Ekuni, futuramente de Souza, que me suportou e apoiou em todos os momentos desta trajetória.

Agradeço aos meus familiares pelo apoio e compreensão de minhas ausências: Marcus Vinicius Branco de Souza (Pai), Heidi Marta Nogueira (mãe), Elenil Gozzo de Souza ("stepmother"), Camila Gozzo de Souza (irmã), em especial aos meus tios Mauricio e Ligia Hironaka, meus "pais" de Maringá, que me aconselharam e me auxiliaram em toda minha trajetória nesta cidade. Aos meus primos de Maringá: Jú, Rafa e Flávia. A minha avó Therezinha, aos tios Ricardo e Viviane, Denise e aos primos de Parnaíba: Samuel, Talita, Bruna e Gabriel. A minha sogra Sueli Abe, a Batian de Londrina, Rodolfo e Mariana. Aos animais que ajudaram a "quebrar a rotina" propiciando momentos de lazer: Pompom (*in memorian*), Mel (*in memorian*), Diana (*in memorian*) Yuki, Juju, Bufy, Dita, Max, Mimi, Stich, Floquinha.

Agradeço a minha orientadora Itana Maria de Souza Gimenes pela paciência para com minhas limitações e confiança depositada em meu trabalho e pela oportunidade de ministrar aulas em conjunto.

Aos docentes da UEM: Elisa H. M. Huzita, Sergio Roberto Pereira, José Roberto, Franklin Cesar Flores, Cristina Ciferri, Ricardo Ciferri, Nardênio, Carniel, Ademir

Constantino, Madalena Dias, Thelma Colanzi, Ronaldo, João Ângelo, Tânia Tait e Marcelo Morandini, Flávio Braga, Flávio Schiavoni e Sandra Ferrari.

Aos colegas do mestrado: Cristiane, Mario Freitas, Daniela, Camila Leal, Ana Paula, Thiago Lopes, Francisco, Jesus, Roberto e Cesar.

A CAPES pelo apoio financeiro.

A prestatividade da Inês e eficiência do Robson, secretários do PCC-DIN, que me deram suporte para os processos burocráticos durante o desenvolvimento dos trabalhos de mestrado.

A todos meus amigos de Itapetininga: Leandro Franci (Nerdão), Diego Leite (Didi), Paulo Henrique (Bola), Carlos Alexandre, Renata, Cinthia, Elton Picoli. Aos amigos de Maringá: Diego, Sapulha, Adriano, Elaine, Garsa, Alessandra, Fukuda, Kevin, Juvenil, Josi, Herso (*in memorian*), Thiago Michel, Maria Fernanda, André (Giga), Laíssa, Andréia Muler, Alison, Caroline Cerri, Rafael Margonato e Tiago Cardoso.

Aos churrascos memoráveis na casa da Dona Teresa (mãe do Adriano), ao pessoal da Igreja Batista Vila Sete: Michel, Cyrano, Daivid, Camila Forti, Pastor Paulo, Marcia.

Ao pessoal da UNIFAMMA que compreendeu minha ausência: André, José Carlos, Rogério, Josiane, Rafael, Bruno Montelares, Evandro B. de Freitas, Evandro Junior, Mercia Cascone, Lupercio Cascone, Graziele, Denise, Fran, Eliete, Claudemir, Luciane, Débora, Amanda, Vilma e Lauro. Aos coordenadores: Marilsa, Claudia, Claudinei, Marcelo, Lidamar, Pedro, Arlete e Geandré. Ao professor Moacir da Silva.

Ao pessoal da Dental Press que me acolheu muito bem: Dona Teresa, Dr. Laurindo, Cleber, Helio, Carlos, Cadu, Soraia, Adriana, Roseli, Rosineide, Josi, Fernandinho, Fernando Batalha, Tati, Ronis e Jonathas.

E finalmente as bandas que embalaram meus trabalhos: Dreamtheater, Hillsong, Symphony X, Pink Floyd, Kiss, Metallica, Tears for fears, Faith no More, Apocalyptca, Pearl Jam, Red Hot Chili Peppers, Jack Johnson e Goo Goo dolls.

Resumo

Aplicações Web são produtos de software que funcionam através da Internet. O desenvolvimento de Aplicações Web tem aumentado significativamente nos últimos anos devido à sua grande portabilidade e visibilidade, assim muitas empresas estão migrando seus sistemas de informação para este tipo de aplicação. Porém, a qualidade dessas aplicações não vêm aumentando proporcionalmente. A maioria dos métodos de desenvolvimento de aplicações Web visa o desenvolvimento de aplicações completas, sem a preocupação com a reutilização de software. Uma das técnicas de reutilização mais bem sucedidas ultimamente é a de Linha de Produto de Software (LPS). Uma LPS contém um núcleo de artefatos que caracteriza uma família de aplicações de um mesmo domínio. Dentre os métodos de desenvolvimento de aplicações Web, optou-se por investigar e adaptar o método OOWS devido à sua compatibilidade com técnicas de LPS. Assim, esta dissertação apresenta uma extensão do método OOWS (*Object Oriented Web Solution*) com conceitos de LPS, denominada SPL-OOWS, para melhorar o desenvolvimento de aplicações Web. Foi desenvolvido um modelo de infra-estrutura básica a partir do qual aplicações podem ser geradas o qual é composto de: Gerenciador de Formulários, Gerenciador de Usuários e o Gerenciador de persistência. As etapas de especificação da solução foram reestruturadas para incluir definições e rastreamento de variabilidade. Foi também incluída uma etapa de configuração da aplicação que consiste em gerenciar os artefatos criados na fase de especificação a fim de criar os modelos de apresentação e navegação. O método proposto foi avaliado por meio do desenvolvimento de um exemplo de aplicação em que foram explorados as questões principais deste. Pode-se evidenciar que apesar do custo de adoção do método, com o amadurecimento dos domínios, o tempo de desenvolvimento tende a diminuir e a produtividade a aumentar, já que, não será necessário criar novos modelos, e sim configurar os modelos existentes para satisfazer as necessidades de uma aplicação diferente de uma mesma família.

Palavras-chave: Linha de produto de software, desenvolvimento de aplicações Web, Engenharia de software, OOWS.

Abstract

Web applications are software products that work in the Internet. The development of Web applications has significantly increased in the last years due to its portability and visibility. Thus, several companies are migrating their applications to information systems accessible through the Internet. However, the quality of these applications has not been increasing proportionally. Most of the Web application development methods aim at producing complete applications, without considering software reuse. One of the most successful techniques of reuse is Software Product Line (SPL). A SPL contains a core asset that characterizes a family of applications of a certain domain. Amongst the Web application development methods, we chose to investigate and adapt the OOWS (*Object Oriented Web Solution*) due to its compatibility with SPL techniques. Thus, this dissertation presents an extension to OOWS, named SPL-OOWS, to improve the development of Web application with reuse. A basic framework was developed from which Web applications can be generated. This framework contains: Form Management, User Management and Persistence Management. The specification stages were redefined to include variability definition and tracing. A configuration stage was included to manage the artifacts obtained in the specification stage to create the presentation and navigation models. SPL-OOWS was evaluated through the development of an application example in which important issues were explored. Evidences show that there is an adoption cost, but once a domain becomes mature, the development cost is reduced and the productivity increases as there is no need to create new models but just to configure the existent ones in order to satisfy the requirements of an application of the same domain.

Keywords: Software Product Line, Web applications development, Software Engineering, OOWS.

Lista de Ilustrações

Figura 1: Método OOHDM 5

Figura 2: Método OOWS Adaptado de Fons (2001) 8

Figura 3: Modelo de Usuários. Adaptado de Fons (2002) 9

Figura 4: Contexto navegacional de produtos de uma aplicação de e-commerce..... 10

Figura 5: Exemplo de Modelo de Navegação – Fonte: Fons et al., 2003. 10

Figura 6: Árvore de Ação Hierárquica 11

Figura 7 : Esquema geral de uma linha de produto (SEI,2008) 18

Figura 8: Exemplo de Modelo de característica, modo edição (esquerda) e configuração (direita). Fonte: (CZARNECKI et al., 2005) 20

Figura 9: Visão geral do ESPLEP- Adaptado de (GOMAA, 2005) 22

Figura 10: Exemplo de caso de uso de um sistema de microondas- Fonte: (GOMAA, 2005) 23

Figura 11: Modelo de características com notação UML - Fonte: (GOMAA, 2005) 24

Figura 12: Diagrama estático com estereótipos de uma LPS - Fonte: (GOMAA, 2005) 26

Figura 13: Diagrama dinâmico interativo do caso de uso Cook Food - Fonte: (GOMAA, 2005) 27

Figura 14: Composição de componentes do sistema de controle de microondas - Fonte: (Gomaa, 2005) 29

Figura 15:Atividades do Processo de gerenciamento de variabilidade- Fonte:(OLIVEIRA JUNIOR, 2005) 32

Figura 16: Processo de Gerenciamento de Linha de Produto utilizando o gerenciamento de variabilidades – Fonte (OLIVEIRA JUNIOR, 2005) 33

Figura 17: Interação entre as atividades e o Gerenciamento de Variabilidades - Fonte: (OLIVEIRA JUNIOR, 2005) 34

Figura 18: SPL-OOWS - Visão geral 36

Figura 19: Infra Estrutura básica 37

Figura 20: Casos de Uso do modelo de Usuário 38

Figura 21: Diagrama de Estados do Caso de Uso Gerenciar Usuário 40

Figura 22: Diagrama de Estados do Caso de Uso Acessar Contextos navegacionais 41

Figura 23: Modelo de Usuário Proposto 42

Figura 24: Diagrama de Classes do controle de usuários - completo 43

Figura 25: Modelo de caso de uso do sistema de comercio eletrônico 47

Figura 26: Modelo de características do sistema de comércio eletrônico 49

Figura 27: Diagrama de Estados do Caso de Uso Controlar Entrega 50

Figura 28: Gerenciar Pagamento 51

Figura 29: Modelo Estático - e-commerce 52

Figura 30: Modelo dinâmico Interativo - Gerenciar entrega de um sistema de comércio eletrônico 53

Figura 31: Diagrama Arquitetural de Implementação 56

Figura 32: Diagrama de seqüência da fase de projeto do caso de uso Gerenciar Entrega do sistema de comércio eletrônico 57

Figura 33: Diagrama de Classes de um comércio eletrônico 59

Figura 34: Modelo Navegacional do comércio eletrônico 62

Figura 35: Estrutura Geral da Árvore de Ação Hierarquica 64

Figura 36 : Diagrama interativo de estados do gerenciamento do carrinho de compras do comércio eletrônico 65

Figura 37: Visão geral da estrutura de implementação do SPL-OOWS 67

Figura 38: Tela de configuração da Infra-estrutura básica do gerenciamento de usuários 68

Figura 39: Modelo de Caso de uso do Gerenciador de Referências 72

Figura 40: Modelo de características da aplicação do Gerenciador de Referências .. 73

Figura 41: Modelo Dinâmico de Estados do caso de uso Gerenciar Área de Concentração 74

Figura 42: Modelo dinâmico do caso de uso Gerenciar Publicação 75

Figura 43: Modelo Estático – Referência 75

Figura 44: Modelo dinâmico interativo do gerenciamento de área de concentração . 76

Figura 45: Modelo Dinâmico interativo do gerenciamento de publicações 77

Figura 46: Diagrama de sequencia de gerenciar area de concentração 78

Figura 47: Diagrama de Sequência de Gerenciar publicação 79

Figura 48: Diagrama de classe do sistema de gerenciamento de publicações 79

Figura 49: Modelo Navegacional do Usuário Administrador e GerenteGrupo 80

Figura 50: Modelo Navegacional do Usuário EstudanteGrupo 81

Figura 51: Modelo Navegacional do usuário Anônimo 81

Figura 52 : Estrutura de implementação da aplicação de Referencia..... 87

Figura 53: Configuração do modelo de características para a configuração da aplicação de referência 87

Figura 54: Visão Geral do Sistema com os Contextos navegacionais 88

Figura 55: Elementos do contexto navegacional - Autor. 89

Figura 56: Formulário de cadastro de autor 89

Figura 57: Página de lista de Autores..... 89

Figura 58: Página de Busca de Autores..... 90

Figura 59: Contexto navegacional Área de Concentração 90

Figura 60: Configuração do modelo de características para a configuração da aplicação de referência 91

Figura 61: Diagrama dinâmico de estados do caso de uso Gerenciar Área de Concentração 91

Figura 62: Diagrama dinâmico de estados do caso de uso Gerenciar Publicações 92

Figura 63: Novo diagrama de classes a partir da seleção do modelo de características 92

Figura 64: Visão Geral do Sistema com os Contextos navegacionais sem a utilização de “Áreas de Conhecimento” 93

Figura 60: Gerenciar Pedido..... 104

Figura 61: Gerenciar Produtos 105

Figura 62 : Controlar Estoque 105

Figura 63 :Montar Carrinho de Compras / Lista de desejos..... 106

Lista de Tabelas

Tabela 1: Cruzamento de informações: características X casos de uso – Fonte: (GOMAA, 2005) 25

Tabela 2: Dependência entre classe e características – Fonte: (GOMAA, 2005) 28

Tabela 3: Cruzamento de informações de características x casos de uso, de acordo com a configuração selecionada. 30

Tabela 4: Configuração da aplicação – modelo de dependência entre classes e características 31

Tabela 5: Descrição do caso de uso Gerenciar Usuário 39

Tabela 6 : Desrição do caso de uso Gerenciar acesos a contextos navegacionais 39

Tabela 7: Descrição do caso de uso Gerenciar grupos de usuários 40

Tabela 8: Modelo de rastreamento de variabilidades do comércio eletrônico 55

Tabela 9 : Mapeamento das Características e os contextos navegacionais de um comércio eletrônico 63

Tabela 10: Árvore de Ação Hierárquica do contexto Produto 66

Tabela 11: Modelo de Rastreamento das variabilidades para a aplicação do gerenciamento de Referências 77

Tabela 12: Mapeamento dos contextos navegacionais e as características da aplicação de gerenciamento de referências 82

Tabela 13: Árvore de ação hierárquica para o contexto navegacional Editora 82

Tabela 14: Árvore de Ação Hierárquica para o contexto navegacional Palavra-chave 83

Tabela 15: Árvore de Ação Hierárquica para o contexto navegacional Autor 83

Tabela 16: Árvore de Ação Hierárquica para o contexto navegacional Area de Concentração 84

Tabela 17: Árvore de Ação Hierárquica para o contexto navegacional Área de Conhecimento 85

Tabela 18: Árvore de Ação Hierárquica para o contexto navegacional Publicação .. 86

Lista de Abreviaturas e Siglas

CSS	Cascading Style Sheets
FAST	Family-Oriented Abstraction, Specification and Translation
FODA	Feature-Oriented Domain Analysis
FORM	Feature-Oriented Reuse Method
LPS	Linha de Produto de Software
OOHDM	Object Oriented Hypermidia design method
OOWS	Object Oriented Web Solution
PGV	Processo de Gerenciamento de Variabilidades
PHP	Hypertext Preprocessor
PLP	Product Line Practice
PLUS	Product Line UML-based Software-Engineering
PuLSE	Product Line Software Engineering
SHDM	Semantic Hypermidia design method
SPL-OOWS	Software Product Line –Object Oriented Web Solution

UML	Unified Modeling Language
UWE	UML Web Based Engineering
WebML	Web Modeling Language
XML	eXtend Markup Language

Sumário

1 Introdução 1

2 Desenvolvimento de Aplicações Web 4

2.1 OOHDM e o SHDM 4

2.2. UWE 6

2.3. WebML 6

2.4 OOWS 7

2.4.1 Modelagem conceitual 8

2.4.2 Modelagem Arquitetural 14

2.4.3 Implementação 15

3 Linha de Produto de Software 17

3.1 Atividades de desenvolvimento de uma LPS 17

3.2 Modelo de Características 19

3.3 Abordagens de linha de produto de software 21

3.3.1 O Método PLUS – (*Product Line UML-Based Software Engineering*) 22

3.3.2 Processo de Gerenciamento de variabilidade para linha de produto de software 31

3.4 Considerações Finais 34

4 SPL-OOWS – Uma extensão do método OOWS para linha de produto de software 35
4.1 Definição da Infra-Estrutura Básica 37
4.1.1 Gerenciamento de Usuário 38
4.1.2 Gerenciamento de Formulários 44
4.1.3 Gerenciamento de Banco de dados 45
4.2 Modelagem Conceitual 46
4.2.1 Requisitos 47
4.2.2 Análise 49
4.2.3 Projeto 55
4.3 Configurações da Aplicação 60
4.3.1 Criação e configuração do Modelo Navegacional 61
4.3.2 Criação e configuração do Modelo de Apresentação 63
4.4 Implementação – Criação de produtos de software 66
4.4.1 Infra Estrutura Básica 68
4.4.2 Camada de Apresentação 69
4.4.3 Camada de Aplicação 69
4.4.4 Camada de Persistência 70
4.5 Considerações finais 70
5 Um exemplo de aplicação do método SPL-OOWS 71
5.1. Domínio de aplicação – Gerenciador de Referências 71
5.2 Modelagem conceitual 72
5.2.1 Requisitos 72
5.2.2 Análise 73
5.2.3 Projeto 78
5.3 Configuração da Aplicação 80

5.4 Implementação - Criação de produtos de software 87
5.4.1 Exemplo de variação de Produtos 91
5.5 Avaliação do método 93
6 Conclusão 95
Referências Bibliográficas 97
Apêndice – A – Diagramas do estudo de caso ilustrativo do comércio eletrônico .. 102
Apêndice – B – CODIGO FONTE DO CONTROLADOR DE USUÁRIOS 106
Apêndice – C – código fonte do Gerenciador de Banco de dados para mysql 110

Capítulo

1

Introdução

Com o aumento do uso e a facilidade de acesso à Internet, a utilização de aplicações Web vem aumentando de forma significativa contribuindo para uma migração das aplicações existentes para a plataforma Web. Muitos métodos de desenvolvimento de aplicações Web têm surgido para melhorar a qualidade dessas aplicações. Exemplos desses métodos são: OOHDM – *Object Oriented Hipermídia Design Method* (SCHWABE e ROSSI, 1998; LIMA e SCHWABE,2003; MOURA e SCHWABE, 2004), OOWS – *Object Oriented Web Solution* (FONS et al., 2001; FONS et al.2003; PASTOR et al., 2001; PASTOR e INSFRAN, 1999; PELECHANO et al., 2003), WebML – *Web Modeling Language* (CERI et al., 2004; CERI et al., 2000) e UWE – *UML Web based Engeenering* (KOCH,2001; KNAPP et al., 2004). Esses métodos de uma maneira geral, possuem etapas similares, tais como: a modelagem conceitual, a modelagem navegacional e a modelagem de apresentação. A modelagem conceitual consiste na identificação de requisitos e de conceitos, em termos de casos de uso, diagramas de classes e de modelos comportamentais. Os modelos navegacionais expressam os elementos de ligação entre os contextos navegacionais, bem como entre os usuários que têm acesso a estes contextos. A modelagem de apresentação consiste na elaboração dos estilos gráficos e dos elementos que propiciam a interação da aplicação com o usuário. O objetivo desses métodos é o desenvolvimento de aplicações específicas, portanto têm pouca ênfase em reutilização de software. Um problema dos métodos existentes é a não utilização de padrões de desenvolvimento consolidados, como o processo unificado, que possui etapas bem definidas para a concepção de um projeto de software.

A reutilização têm sido considerada uma das técnicas mais importantes da engenharia de software para se atingir software de melhor qualidade e com maior produtividade. Dentre as técnicas de reutilização, uma que tem ganhado relevância é Linha de Produto de Software - LPS. Uma LPS é descrita por Clements e Northorp (2002) como sendo um conjunto de sistemas de software que compartilham um conjunto de características comuns e gerenciáveis, visando atender às necessidades de um segmento particular de mercado ou missão. Este conjunto de sistemas é desenvolvido a partir de um núcleo comum de artefatos de uma forma sistemática. A aplicação de uma técnica de LPS ao desenvolvimento de aplicações Web visa permitir a reutilização deste núcleo comun de artefatos desde infra-estrutura para produzir aplicações para domínios específicos.

Um dos métodos recentes de desenvolvimento de aplicações Web é o OOWS (Fons et al., 2001). Este método adota a notação UML e representa uma evolução do OO-Method (Pastor et al., 2001). A compatibilização do OOWS com técnicas de LPS mostra-se interessante, pois este método possui um processo de desenvolvimento similar ao de métodos de desenvolvimento de LPS como o PLUS (GOMAA, 2005) e o Processo de Gerenciamento de Variabilidades - PGV (OLIVEIRA JUNIOR, 2005). Esses métodos também são compatíveis com o processo unificado (JACOBSON et al., 1999).

Assim, este trabalho propõe o método SPL-OOWS, uma extensão do OOWS que inclui conceitos de LPS. O objetivo deste método é permitir a construção de famílias de aplicações Web, utilizando desta forma conceitos de reutilização de software para melhorar a qualidade e a produtividade de aplicações.

Esta dissertação está dividida em 6 capítulos. O segundo capítulo é dedicado aos conceitos teóricos dessa pesquisa abrangendo especialmente a literatura recente sobre métodos de desenvolvimento Web; neste capítulo, portanto, o método escolhido como referencial principal para o desenvolvimento deste trabalho, o OOWS, é descrito. No terceiro capítulo é apresentado o conceito de LPS, incluindo o modelo de características e métodos de desenvolvimento de LPS existentes. Destacam-se o método PLUS (GOMAA, 2005) e o PGV (OLIVEIRA JUNIOR, 2005) utilizados como referência para extensão do OOWS. No quarto capítulo é apresentado o SPL-OOWS, uma extensão do método OOWS para LPS. Para mostrar a aplicação do método proposto foi utilizado um estudo de caso ilustrativo no âmbito

do comércio eletrônico. O quinto capítulo consiste na apresentação de um exemplo de aplicação em que o método proposto é explorado em outro domínio de aplicação. O sexto capítulo apresenta as conclusões e trabalhos futuros.

Capítulo

2

Desenvolvimento de Aplicações Web

Conforme Bianchini (2008), os métodos de desenvolvimento de aplicações Web são similares aos métodos de desenvolvimento convencionais, porém, incluem novos conceitos que devem ser tratados de forma diferenciada como: a navegação em ambiente imprevisível, a usabilidade elaborada e a possibilidade de inclusão de materiais multimidiáticos (ex. imagens, vídeos, animações e sons). Esses métodos, conforme tem sido abordado pela literatura especializada apresentam aspectos similares como a modelagem conceitual, a construção de modelos navegacionais, bem como de modelos de apresentação. Existem vários métodos de desenvolvimento de aplicações Web, este capítulo apresenta uma seleção deles por serem considerados completos e historicamente relevante. Apresenta, principalmente, o OOWS que é utilizado com base para o método proposto nesta dissertação.

2.1 OOHDM e o SHDM

OOHDM (SCHWABE, 1998) é um método baseado em modelos e que utiliza técnicas de orientação a objetos para o projeto de aplicações hipermídia. Este método enfatiza o aspecto navegacional da aplicação. O método OOHDM apresenta as seguintes etapas de desenvolvimento: modelagem conceitual, modelagem de navegação, modelagem de interface e implementação, em que se tem a combinação de estilos de desenvolvimento iterativo e incremental, de modo que para cada etapa do processo um modelo é construído ou incrementado. A Figura 1 ilustra a visão geral deste método.

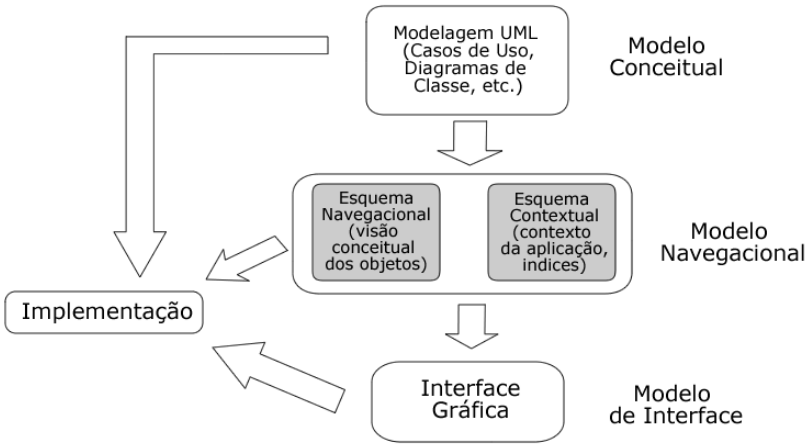


Figura 1: Método OOHDM

No desenvolvimento do modelo conceitual, observa-se a representação do domínio e contexto da aplicação. A representação do domínio é realizada por meio de modelos UML (JACOBSON, 1999) com base nas perspectivas e nos atributos dos subsistemas. O modelo navegacional é definido por meio da elaboração de contextos navegacionais que determinarão o que os usuários da aplicação acessarão. O modelo de interface define os objetos de navegação para a interação do usuário com a aplicação. Na fase de implementação, tem-se o mapeamento dos modelos conceitual, navegacional e de interface.

Existe um ambiente de desenvolvimento de apoio ao OOHDM chamado HyperDE (NUNES, 2005). Este ambiente oferece a possibilidade de construção de uma aplicação Web baseada no método SHDM (NUNES, 2005) que é uma extensão do método OOHDM voltado para a elaboração de Web semântica. O método SHDM, segundo Nunes (2005), é constituído das seguintes etapas:

I. Identificação de atores e tarefas, especificação de cenários e casos de uso, especificação dos diagramas de interação do usuário (UIDs) e validação dos casos de uso e UIDs.

II. Criação da ontologia conceitual, caso não seja empregada uma ontologia já existente.

III. Uma vez definida a ontologia conceitual as instâncias conceituais podem ser geradas.

IV. Especificação do mapeamento navegacional.

V. Especificação do projeto da interface abstrata.

VI. Implementação da aplicação utilizando os artefatos.

2.2. UWE

O UWE (KOCH, 2001;KOCH e KRAUS, 2002) é um método de desenvolvimento de aplicações Web que enfatiza a modelagem sistemática, personalizada e a geração semi-automática de código. O UWE utiliza como notação UML e mecanismos de extensão.

UWE possui quatro modelos: conceitual, navegação, apresentação e tarefas. A elaboração do modelo conceitual é realizada utilizando os diagramas de classe, de pacotes e casos de uso. O modelo de navegação inclui a especificação de cada objeto que pode ser visitado pela navegação através da aplicação Web e como esses objetos podem ser alcançados pelas estruturas de acesso. O modelo de apresentação descreve onde e como os objetos de navegação e acesso primitivo serão apresentados ao usuário. A constituição do modelo de tarefas serve para apoiar a geração automática do código da aplicação Web.

O processo UWE conta com o apoio da ferramenta CASE ArgoUWE (ARGOUWE, 2007) que auxilia o desenvolvedor na tarefa de elaboração e concepção de aplicações Web.

2.3. WebML

O método de desenvolvimento WebML (CERI, 2004) permite aos desenvolvedores expressarem as características centrais de uma aplicação de alto nível, sem a necessidade de especificar uma arquitetura detalhada. O princípio de WebML está associado à uma representação gráfica intuitiva que pode ter o apoio de ferramentas CASE e assim, pode ser utilizada por desenvolvedores que não detenham conhecimento tecnológico sobre o desenvolvimento (por exemplo: produtores de interface e os produtores de conteúdo). A

especificação de uma aplicação em WebML consiste de quatro perspectivas: o modelo estrutural, o modelo de hipertexto, o modelo de apresentação e um modelo de adaptação.

No modelo estrutural do WebML, observa-se o desenvolvimento de diagramas de entidade relacionamento e diagramas de classes que representam a construção da base de dados e da estrutura da aplicação. O modelo de hipertexto é dividido em dois sub-modelos, o modelo de composição que especifica quais páginas que compõem um hipertexto e quais os conteúdos que constituem uma página; e o modelo de navegação que expressa o relacionamento entre os conteúdos. No modelo de apresentação é definido e desenvolvido como será o projeto gráfico da aplicação. No modelo de adaptação, observa-se a atribuição de níveis de acesso aos usuários do sistema, podendo assim restringir ou autorizar o acesso a determinados conteúdos.

WebML é apoiado por uma ferramenta comercial consistente chamada *WebRatio* (WEBRATIO, 2007). Esta ferramenta possui como característica a automação dos processos descritos pelo WebML e a possibilidade de geração da aplicação a partir dos modelos desenvolvidos na ferramenta. A partir da WebRatio desenvolve-se um diagrama de entidade relacionamento e os mapas navegacionais, onde se indicam elementos de apresentação, fazendo assim a geração da aplicação de forma automática.

2.4 OOWS

OOWS é uma extensão de um método orientado a objetos baseado no OO-Method (PASTOR e INSFRAN, 2001). Conforme ilustra a Figura 2, o processo OOWS engloba duas fases: (i) especificação do sistema e (ii) desenvolvimento da solução. A especificação do sistema, por sua vez, envolve as seguintes etapas: levantamento de requisitos funcionais e modelagem conceitual. Os requisitos funcionais são especificados em três modelos: dinâmico, semântico e funcional. Na fase de desenvolvimento da solução, busca-se a geração de elementos de software que representem uma solução tecnológica para o sistema. Nesta fase, tem-se a definição da arquitetura do sistema e de sua implementação. A arquitetura de sistema adotada consiste em três camadas: apresentação, aplicação e persistência.

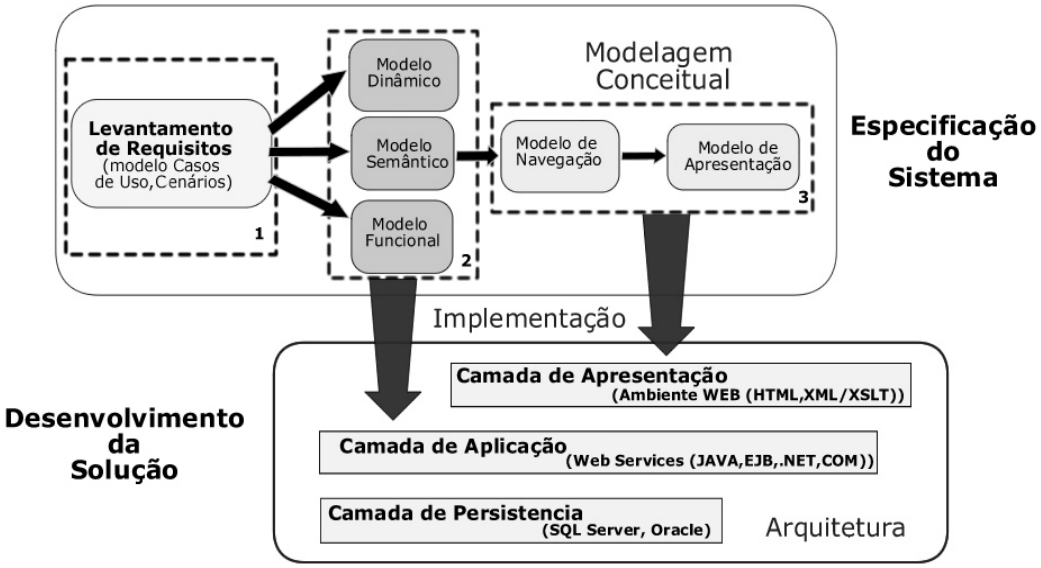


Figura 2: Método OOWS Adaptado de Fons (2001)

2.4.1 Modelagem conceitual

Esta etapa consiste na captura de requisitos para aplicações Web, gerando assim especificações do sistema. As especificações podem ser classificadas pela: funcionalidade, navegabilidade e especificidade das aplicações Web. Os diagramas que descrevem o modelo conceitual da aplicação são: diagramas de casos de uso, diagrama de classes, diagramas de estados e uma especificação textual definindo a semântica dos estados. Na modelagem da apresentação e navegação, temos que a partir do diagrama de classes e dos requisitos de navegação é possível criar o modelo navegacional. A especificação de requisitos de apresentação é feita utilizando um modelo de apresentação baseado no modelo navegacional.

O método OOWS utiliza três modelos oriundos do OO-Method conforme descritos a seguir.

a) Modelo de Usuário

O modelo de usuário expressa que tipo de usuário pode interagir com o sistema e qual sua visão do sistema. Isto é representado por meio de uma hierarquia de atores. Existem três tipos de usuários: anônimo, registrado e genérico, conforme mostra a Figura 3. Um

usuário registrado pode ser qualificado de acordo com as atividades que ele desenvolverá no sistema. Os tipos de usuários registrados são desenvolvidos no modelo de usuário e suas atividades dentro do sistema são desenvolvidas no modelo de navegação. Um usuário registrado pode ser qualificado de acordo com as atividades que ele desenvolverá no sistema. Os tipos de usuários registrados são desenvolvidos no modelo de usuário e suas atividades dentro do sistema são desenvolvidas no modelo de navegação.

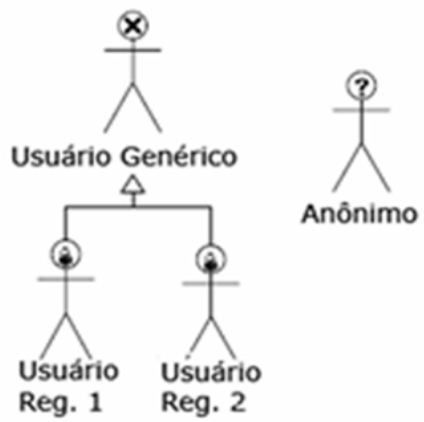


Figura 3: Modelo de Usuários. Adaptado de Fons (2002)

b) Modelo de Navegação

O modelo de navegação depende do modelo de usuário pois nele são definidos os tipos de usuários do sistema. Após definidos os usuários, define-se também a semântica de navegação do sistema baseado no diagrama de classes criado no modelo semântico que define as atividades do usuário. Um mapa de navegação representa a acessibilidade e a visibilidade associadas a cada tipo de usuário. Este mapa é representado por um grafo direcionado em que cada nó pode ser um contexto navegável ou um subsistema navegável. Um subsistema navegável é representado como um sub-grafo. Um contexto navegável é representado como um pacote UML que contém um conjunto de informações abstratas (AIU – Abstract Information Unit). Uma AIU representa uma unidade para recuperar alguma informação específica. A Figura 4 mostra um exemplo de um contexto navegacional onde é possível instanciar várias páginas como: Formulário do Produto (cadastro), Visualização de produtos e a Busca por produtos.

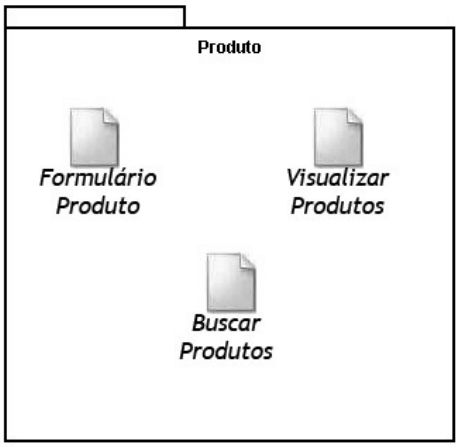


Figura 4: Contexto navegacional de produtos de uma aplicação de e-commerce

A Figura 5 mostra um exemplo de um mapa navegacional, onde o membro (`Member`) tem acesso aos contextos navegacionais `Activities`, `Groups`, `ResearchLines`, `Publications`, `Members`, `Resources`, `Projects` e `Guests` de forma indireta.

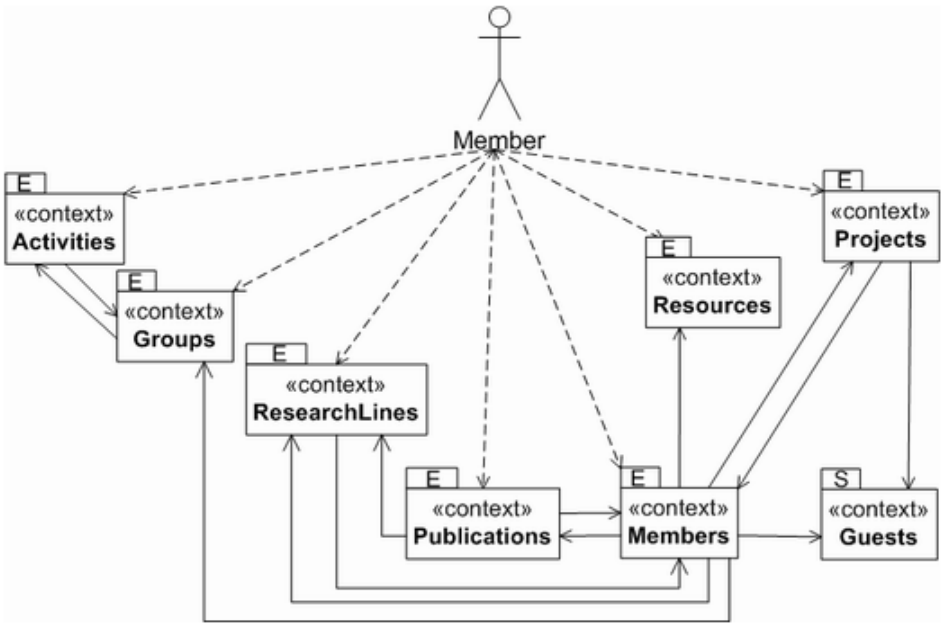


Figura 5: Exemplo de Modelo de Navegação – Fonte: Fons et al., 2003.

c) Modelo de Apresentação

O modelo de apresentação pode ser visto como uma árvore de ação hierárquica composta de três níveis que mostram o acesso à aplicação (PASTOR e INSFRÁN, 1999). O modelo de apresentação é elaborado com base no modelo navegacional. A apresentação do sistema é especificada por meio de padrões que são associados às premissas do contexto navegacional chamadas de Unidades de Interação (UI). Os padrões incluem paginação da informação e organização das informações (ex. registro, tabelas, árvore). O modelo básico da apresentação não inclui o estilo da página (ex. cores, fontes, figuras e barras). A Figura 6 ilustra os níveis da árvore de ação hierárquica (MOLINA, MELIA e PASTOR, 2002) os quais são descritos a seguir.

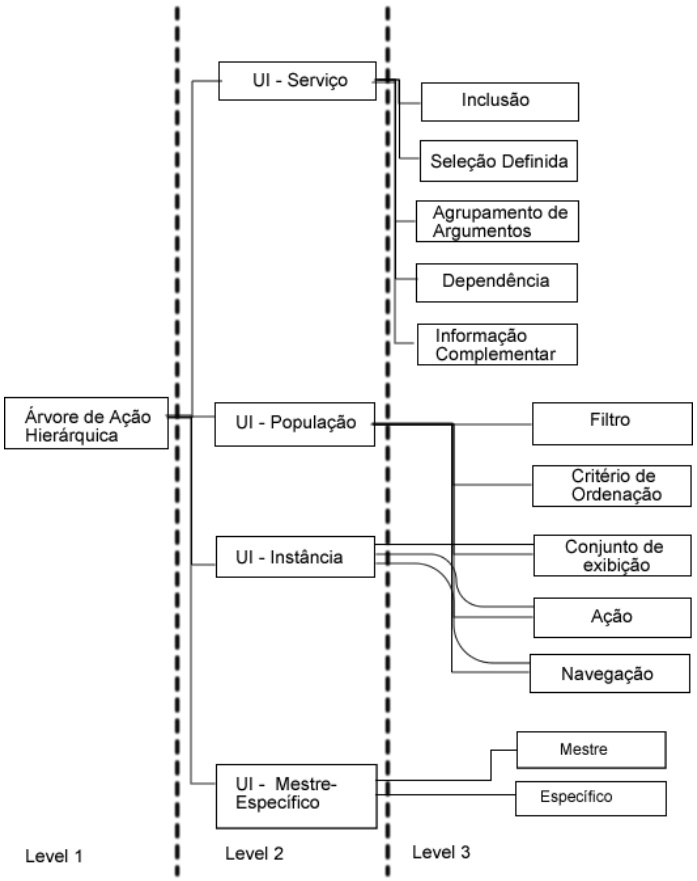


Figura 6: Árvore de Ação Hierárquica

Unidades de Interação de Nível 2

As especificações das UI de Nível 2 são: Serviço, Instância, População e Mestre-específico, conforme descrito a seguir:

UI – Serviço: o desenvolvedor identifica o serviço associado a uma classe. Em termos de especificação o serviço de apresentação encapsula a UI para prover o serviço na interface. Asim, a especificação do serviço pode ser completada perguntando-se ao usuário: Quais os dados necessários para este serviço? Quantos dados de entrada serão agrupados? Os campos de entrada estão relacionados? Qual tipo de resposta é necessária para cada objeto selecionado? Muitas respostas para estas indagações podem ser encontradas usando outros padrões: **inclusão** (limita-se a introdução dos valores), **seleção definida** (define enumerando os valores válidos), **seleção de população** (expressa como selecionar objetos), **dependência** (expressa a interdependência entre objetos), **informação complementar** (provê resposta extra para objetos identificados), **recuperação de estado** (recuperação dos valores de argumentos de um atributo) e **agrupamento** (agrupamento lógico dos argumentos).

UI – Instância: modelagem de dados de apresentação de uma instância. Utilizado convencionalmente para a manipulação de objetos. Em termos de usuário, a apresentação de um contexto vai além da necessidade de observar objetos únicos, pois o usuário pode querer modificar o estado do objeto ao qual está navegando e navegar entre objetos relacionados na instância. A análise então identifica os objetos relacionados para que então o usuário possa definir as ações, relacionamentos e atributos que poderão ser visualizados a partir do objeto em questão. A especificação da instância é definida utilizando elementos do terceiro nível da árvore de ação hierárquica: conjuntos de exibição, ações e navegação.

UI – População: este padrão visa mostrar um conjunto de instâncias de uma classe e trabalha com coleções de objetos. Ele é composto de 5 (cinco) elementos da árvore de ação hierárquica, à saber: filtros, critério de ordenação, conjunto de exibição, ações e navegação.

UI – Mestre-Específico - os componentes são divididos em dois tipos lógicos: componentes específicos e mestre. Esses são especificados por meio do significado de um relacionamento de agregação do mestre para o específico. Quando um componente do tipo mestre muda, o componente específico também é afetado. Em aplicações de negócios, este

padrão é bastante comum, por exemplo: uma fatura e a possibilidade de inserir linhas nesta fatura. Com o intuito de especificá-lo, o desenvolvedor deve detectar e criar as classes, componentes elementares e finalmente a unidade de mestre-específico. O componente mestre pode ser um padrão da instância que ilustrará um objeto, ou um padrão de população que mostrará um conjunto de objetos. O padrão específico pode ser de dois tipos: (i) instância, se a regra do caminho for uma só; é (ii) população da classe, se for multivalorado ou um padrão Mestre/específico, o qual pode apresentar recursivamente elementos de mais baixo nível. A especificação da apresentação do padrão Mestre/específico é expressa através dos seguintes conceitos: classe Mestre, Padrão de apresentação atuando como classe mestre, Caminho de regras (relacionamento de agregação) visível pela classe mestre que atua como específica e o padrão de apresentação como detalhe.

Unidades de Interação de Nível 3

As especificações das UIs de Nível 3 são: filtros, critérios de ordenação, conjunto de exibição, navegação e ações, a seguir é detalhada cada uma destas unidades.

Filtros - um filtro é utilizado quando se necessita buscar objetos a partir de uma condição. O filtro é concebido pelo desenvolvedor para satisfazer uma condição de busca para o usuário. Exemplo: buscar os produtos que contenham a palavra "linha de produto".

Critérios de Ordenação - os critérios de ordenação ditarão em qual ordem os objetos serão exibidos (ex. ascendente ou descendente), assim como quais serão os atributos que integrarão o critério de ordenação. Exemplo: ordenar os produtos por nome, de forma Ascendente.

Conjunto de exibição - após a aplicação do filtro e do critério de ordenação, o conjunto de exibição definirá os atributos que serão visíveis. Exemplo: nome do produto, valor do produto e peso do produto.

Navegação - quando há a necessidade de relacionar informações em determinado conjunto de exibição, o elemento de navegação é utilizado para criar este relacionamento. Exemplo: na descrição do produto existe uma ligação para verificar as avaliações do produto. Esta ligação é representada por um elemento de navegação.

Ações - executam métodos para modificar estado dos objetos, assim as ações definem os processos de alteração das págins de exibição.

Além dos modelos oriundos do método OO (OO-Method) a modelagem conceitual envolve a definição de mais três modelos (PELECHANO et al., 2003), conforme segue:

Modelo semântico: inclui as classes e seus relacionamentos representados por um diagrama de classes;

Modelo dinâmico: representa a forma de interação das classes representada por diagramas de transição de estados (*Statechart*);

Modelo funcional: este modelo tem o papel de especificar as etapas das transições de estados propostas pelo modelo dinâmico utilizando uma descrição textual padronizada.

2.4.2 Modelagem Arquitetural

A arquitetura de uma aplicação desenvolvida com OOWS possui várias camadas, levam em consideração a natureza das aplicações Web. Na etapa da modelagem arquitetural, como se pode ver na Figura 2, as camadas e suas especificações são definidas (FONS, 2003) conforme descrita a seguir.

Camada de apresentação: inclui componentes de interface gráfica como páginas Web e objetos visuais. Nesta camada, define-se a interação do usuário com a aplicação, deste modo, ela deve ser desenvolvida para diferentes tipos de usuários.

Camada de Aplicação: define a lógica do negócio de modo a permitir a implementação das estruturas e funcionalidades das classes obtidas na modelagem conceitual. Esta camada é dividida em duas subcamadas: componente de interface de negócio[1] que fornece o apoio para a implementação de aplicações Web. Ela atua como um elo de ligação entre a camada de persistência e a lógica do negócio[2] que implementa as estruturas e as funcionalidades das classes da modelagem conceitual.

[1] Do inglês Business Facade

[2] Do inglês Business Logic

Camada de Persistência: é a camada responsável por armazenar e controlar os dados omitindo maiores detalhes do repositório de dados para as camadas superiores.

2.4.3 Implementação

A fase de implementação envolve as transformações das abstrações conceituais para os elementos de software que implementam cada camada da arquitetura especificada. Segundo Pelechano et al. (2003), a estratégia de desenvolvimento para cada camada segue os parâmetros descritos a seguir conforme as camadas da arquitetura.

Camada de Apresentação

A partir do modelo de navegação e apresentação, um grupo relacionado de páginas Web pode ser obtido para cada tipo de usuário de uma maneira sistemática. Essas páginas definem: a interface de navegação, a visualização de dados e o acesso às funcionalidades oferecidas ao usuário. Um diagrama de usuários é especificado para expressar quais tipos de usuários podem interagir com o sistema. A partir do diagrama de usuários e do diagrama de classes, é possível criar um mapa navegacional que é responsável por estruturar o acesso ao sistema. Este mapa consiste de um grupo de nós, chamados de contextos navegacionais que visam estruturar as informações da aplicação. Uma página Web é criada para cada contexto navegacional. Esta página é responsável por responder as informações especificadas no contexto navegacional.

Camada de Aplicação

A partir das classes definidas no modelo conceitual, é definido um mapeamento entre as classes de domínio e de modelagem que implementam a aplicação Web com a camada de aplicação. Esta camada pode ser dividida em duas subcamadas: Interface do Negócio e a lógica do negócio. Na interface de negócio é criada uma aplicação Web para cada domínio de uma classe. Este serviço é implementado por uma classe que combina um conjunto de padrões de projeto (*facade*, *singleton* e *factory*)(GAMMA et. al.,1995). Na subcamada da lógica do negócio são implementadas as classes do modelo conceitual, seguindo o padrão de modelo de domínio e suas variações. Nessas classes são implementadas as funcionalidades especificadas no modelo conceitual. Elas são responsáveis por: salvar, responder e atualizar os dados dos objetos na camada de persistência.

Camada de Persistência

A partir do modelo conceitual podem ser definidos os elementos da camada de persistência que encapsulam o acesso ao banco de dados e sua tecnologia e consiste de um elemento controlador para realizar a persistência dos dados.

Atualmente, o conjunto de ferramentas OlivaNOVA (CARE,2007) apoiam a fase de implementação, pois a partir da modelagem conceitual é possível a geração de código de forma automática.

Capítulo

3

Linha de Produto de Software

Nos últimos anos tem se percebido uma crescente adoção da abordagem de LPS devido aos seus muitos benefícios. Segundo Clements e Northrop (2002), os benefícios da adoção de uma abordagem de LPS podem ser classificados em: **organizacionais**: melhor compreensão do domínio, alta qualidade dos produtos e confiança do cliente; **engenharia de software**: melhor análise e reutilização dos requisitos e dos artefatos, controle da qualidade dos produtos, estabelecimento de padrões e documentação reutilizável; e, **negócio**: redução dos gastos com teste e manutenção. Para alcançar tais benefícios, são necessários alguns procedimentos e mudanças.

Neste trabalho, os conceitos de LPS são utilizados para estender um método de desenvolvimento de aplicações Web com o objetivo de promover reutilização e assim melhorar a qualidade das aplicações desenvolvidas. Este capítulo apresenta os principais conceitos de LPS e uma visão das abordagens de desenvolvimento, destacando a abordagem PLUS (GOMAA, 2005) e o PGV (OLIVEIRA JUNIOR, 2005) que são utilizados como referência para extensão do método OOWS.

3.1 Atividades de desenvolvimento de uma LPS

O SEI (2008), por meio da iniciativa PLP (Product Line Practice), estabeleceu algumas atividades essenciais de uma abordagem de LP. Essas atividades são:

- **Desenvolvimento do Núcleo de Artefatos**: responsável por estabelecer uma infra-estrutura central que é reutilizada pelos produtos gerados a partir da LP.

- **Desenvolvimento do Produto:** responsável pelo desenvolvimento de produtos, membros da família de produtos representados pela LP.

- **Gerenciamento da Linha de Produto:** responsável pelo gerenciamento da LP, garantindo que todas as atividades sejam realizadas de acordo com um planejamento coordenado. O gerenciamento pode ser dividido em gerenciamento técnico, que coordena as atividades de desenvolvimento e gerenciamento organizacional, que deve garantir que as unidades organizacionais recebam os recursos corretos em quantidades suficientes.

A Figura 7 ilustra as interações entre as atividades de uma LPS em três círculos, ilustrando engrenagens. Esses círculos indicam que as atividades de uma LPS são altamente interligadas e iterativas. As flechas rotativas indicam que além dos artefatos do núcleo para desenvolver os produtos, também são utilizadas as revisões desses artefatos ou até mesmo novos artefatos no desenvolvimento de LPS.

Figura 7 : Esquema geral de uma linha de produto (SEI,2008)

3.2 Modelo de Características

O modelo de características[3], é usado para modelar aplicações de um domínio, resumindo as capacidades comuns e diferentes de uma aplicação. Segundo Kang (1990) e Simons et al. (1996), características podem ser definidas como elementos de um sistema relevantes e visíveis ao usuário final. O conceito das características vem da engenharia de domínio (KANG, 1990) e tem sido melhorado constantemente para suprir às demandas da LPS (SIMONS et al., 1996; VAN GURP et al., 2001; SOCHOS et al., 2004; CZARNECKI et al., 2005).

Segundo Czarnecki et al. (2005), a modelagem de características é uma importante técnica para captura e gerenciamento de atributos e variabilidades dentro da LPS por todo estágio de projeto. Os atributos geralmente representados em um modelo de característica são os relacionamentos entre características e pontos de variação, os relacionamentos entre características, e a característica de tempo de resolução.

Segundo Van Gurp e Bosch (2001), as características podem ser classificadas em:

- **obrigatórias**: são as características que identificam um produto e são essenciais para o seu funcionamento. Por exemplo, enviar e receber e-mail em um sistema de correio eletrônico;

- **opcionais**: são as características, que quando habilitadas, podem adicionar algum valor às características obrigatórias de um produto. Por exemplo, a possibilidade de se adicionar assinatura ao e-mail;

- **variáveis**: são as características que podem ser selecionadas para estar presente em um produto, a partir de um grupo de características. Por exemplo, o sistema de correio eletrônico pode oferecer um editor de mensagens;

[3] Do inglês features.

- **externas**: são as características oferecidas pela plataforma-alvo do sistema. Por exemplo, um sistema de correio eletrônico com capacidade de fazer conexões TCP (*Transmission Control Protocol*). O modelo de características é geralmente representado por meio de diagramas em forma de árvores que contêm as características identificadas para uma família de produtos.

Um exemplo de um modelo de características pode ser visualizado na Figura 8. Este foi desenvolvido utilizando uma ferramenta denominada *Feature-plugin* (ANTKIEWICZ e CZARNECKI, 2004) desenvolvida para o ambiente de desenvolvimento Eclipse (ECLIPSE, 2008) e ilustra um perfil de segurança para acessos a sistemas, onde é possível configurar o tempo de validade de uma senha (`expiration`), a forma dos caracteres (`chars`), o conjunto de permissões (`permissionSet`), a forma de apresentação da caixa de diálogo de arquivo(`fileDialog`), as variáveis de ambiente (`environmentVariables`) os tipos de permissão (`permission`).

Figura 8: Exemplo de Modelo de característica, modo edição (esquerda) e configuração (direita). Fonte: (CZARNECKI et al., 2005)

3.3 Abordagens de linha de produto de software

A literatura existente apresenta abordagens de LPS, dentre elas, pode-se citar algumas (GIMENES e TRAVASSOS, 2002): Feature-Oriented Domain Analysis (FODA) (KANG, 1990), Synthesis (SPC, 1993), Family-Oriented Abstraction, Specification and Translation (FAST) (WEISS e CHI TAU, 1999), Product Line Software Engineering (PuLSE) (BAYER et al., 1999), a abordagem proposta por Bosch (BOSCH, 2000), a iniciativa Product Line Practice (PLP) (CLEMENTS e NORTHROP, 2002), o método KobrA (ATKINSON et al., 2001), Product Line UML-Based Software Engineering (PLUS) (GOMAA, 2005) e o PGV (OLIVEIRA JUNIOR, 2005).

O FODA é um dos precursores da abordagem de LPS. Ele foi desenvolvido no SEI como um método para análise de domínio. Descata-se pela introdução do modelo de features, amplamente utilizado nas abordagens de LPS. Em seguida, foi desenvolvida uma extensão dessa abordagem, chamada FORM (KANG et al., 1998) que inclui questões arquiteturais e de componentes.

As abordagens Synthesis e FAST tratam de questões abrangentes de LPS. São abordagens precursoras que serviram de base para possibilitar a definição de um contexto mais geral para LP, como o definido na iniciativa PLP, porém não tiveram influência direta neste trabalho.

O método KobrA seguiu a abordagem Pulse caracterizando-se como uma abordagem de LP baseada em componentes. O KobrA engloba várias tecnologias da engenharia de software como desenvolvimento baseado em componentes, frameworks, modelagem de processos e arquiteturas. Ele é composto por duas etapas: engenharia de framework, que é a representação estática de um conjunto de componentes, e engenharia de aplicação, que usa o framework para construir aplicações de domínio específico.

A abordagem Bosch considera uma LPS em três dimensões, sendo elas: arquitetura, componente e sistema; negócios, organização, processo e tecnologia; e desenvolvimento, aplicação e evolução. Esta abordagem enfatiza a especificação de uma arquitetura de software formada por componentes reutilizáveis.

O Método PLUS (GOMAA, 2005) e o Processo de Gerenciamento de Variabilidade (OLIVEIRA JUNIOR, 2005) para LPS foram utilizados como base para estender o método OOWS com conceitos de LPS, assim essas abordagens são destacadas nas próximas seções.

3.3.1 O Método PLUS – (*Product Line UML-Based Software Engineering*)

O método PLUS (GOMAA, 2005) é baseado em UML e composto pelo processo ESPLEP[4] que apresenta uma perspectiva para o desenvolvimento de LPS (veja Figura 9). O objetivo é deixar explícito as variabilidades e os pontos comuns de uma LPS. PLUS provê um conjunto de conceitos e técnicas que complementam UML para representar variabilidades. O método PLUS é composto das seguintes etapas de desenvolvimento: Modelagem de requisitos para LPS; Análise para LPS; Projeto para LPS; e, Engenharia da aplicação de software. Essas etapas são descritas a seguir.

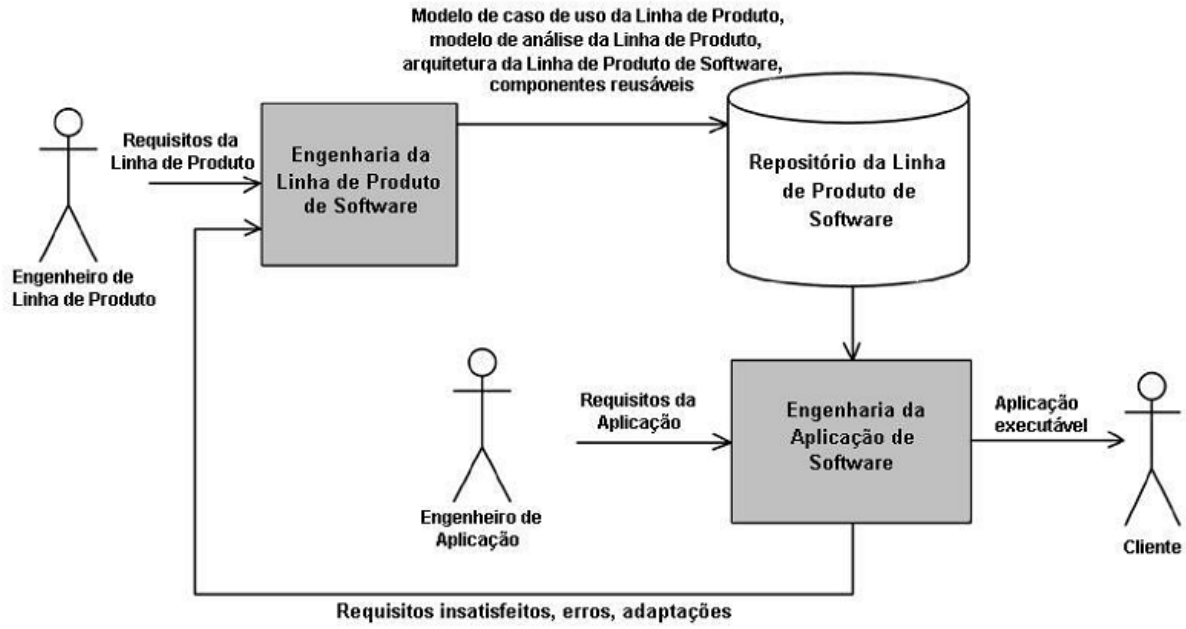


Figura 9: Visão geral do ESPLEP- Adaptado de (GOMAA, 2005)

Modelagem de requisitos para LPS

A modelagem de requisitos para LPS consiste na concepção dos seguintes artefatos: modelo de casos de uso, modelo de variabilidades e a relação entre os modelos de casos de

[4] Do inglês Evolutionary Software Product Line Engineering Process

uso e o modelo de variabilidades. O método PLUS indica com estereótipos qual o tipo de cada caso de uso, os estereótipos são:

- `kernel` (núcleo) - indica que o caso de uso em questão é essencial para o sistema;
- `optional` (opcional) - indica que o caso de uso é opcional e pode ser adicionado à lógica do sistema; e,
- `alternative` (alternativo) - sugere a existência de alternativas e que pelo menos um caso de uso alternativo deva ser selecionado.

A Figura 10 ilustra um exemplo de caso de uso de um sistema de controle de um aparelho de microondas.

Figura 10: Exemplo de caso de uso de um sistema de microondas- Fonte: (GOMAA, 2005)

Para complementar a modelagem de requisitos é elaborado o modelo de características que é chave para a modelagem de LPS. O método PLUS permite a modelagem de características comuns, opcionais e alternativas, assim como uma forma de representação das características em UML. A Figura 11 ilustra um exemplo de modelo de características proposta por Gomaa (2005).

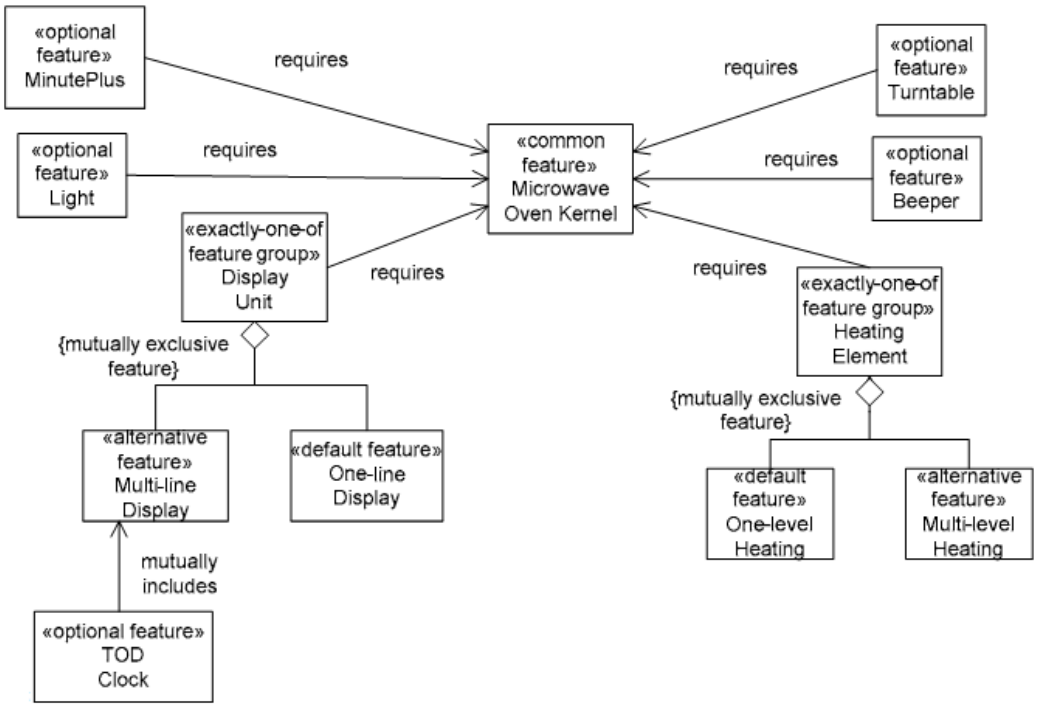


Figura 11: Modelo de características com notação UML - Fonte: (GOMAA, 2005)

Com o modelo de características e o modelo de casos de uso modelados, então é realizado o relacionamento entre eles, a representatividade deste relacionamento é realizado como ilustra a Tabela 1.

Tabela 1: Cruzamento de informações: características X casos de uso – Fonte: (GOMAA, 2005)

Característica	Categoria	Caso de uso	Categoria do caso de uso/ ponto de variação	Nome do Ponto de variação
Microwave Oven Kernel	common	Cook Food	Kernel	
Light	optional	Cook Food	vp	Light
Turntable	optional	Cook Food	vp	Turntable
Beeper	optional	Cook Food	vp	Beeper
Minute Plus	optional	Cook Food	vp	Minute Plus
One-Line Display	default	Cook Food	vp	Display Unit
Multi-Line Display	alternative	Cook Food	vp	Display Unit
English	alternative	Cook Food	vp	Display Language
French	alternative	Cook Food	vp	Display Language
Spanish	alternative	Cook Food	vp	Display Language
German	alternative	Cook Food	vp	Display Language
Italian	alternative	Cook Food	vp	Display Language
Boolean Weight	default	Cook Food	vp	Weight Sensor
Analog Weight	alternative	Cook Food	vp	Weight Sensor
One-level Heating	default	Cook Food	vp	Heating Element
Multi-level Heating	alternative	Cook Food	vp	Heating Element
Power Level	optional	Cook Food	vp	Power Level
TOD Clock	optional	Set Time of Day Display Time Fo Day	optional	
12/24 Hour Clock	parametrized	Set Time of Day Display Time Fo Day	vp	12/24 Hour Clock
Recipe	optional	Cook food with Recipe	optional	

Etapa de análise para LPS

A etapa de análise para LPS é composta de quatro modelos, sendo eles: modelo estático, dinâmico interativo, dinâmico da máquina de estados e o modelo de dependência de características. Cada modelo é descrito a seguir:

- **Modelo estático** – Definição das entidades estáticas e seus relacionamentos da LPS, sendo que estas futuramente se transformarão em classes na etapa de projeto. Neste modelo, há a determinação dos tipos das entidades que podem ser: *kernel*, *optional* e *variant*. Uma entidade do tipo *kernel* faz parte do núcleo da aplicação, sendo que esta é obrigatória para os produtos criados

pela LPS, já a entidade *optional* pode não fazer parte da aplicação criada pela LPS e por fim as entidades do tipo *variant* são as que irão definir os pontos de variação da aplicação, ou seja, são entidades mutáveis de acordo com a aplicabilidade da aplicação a ser criada pela LPS. A Figura 12 mostra o diagrama estático para a aplicação.

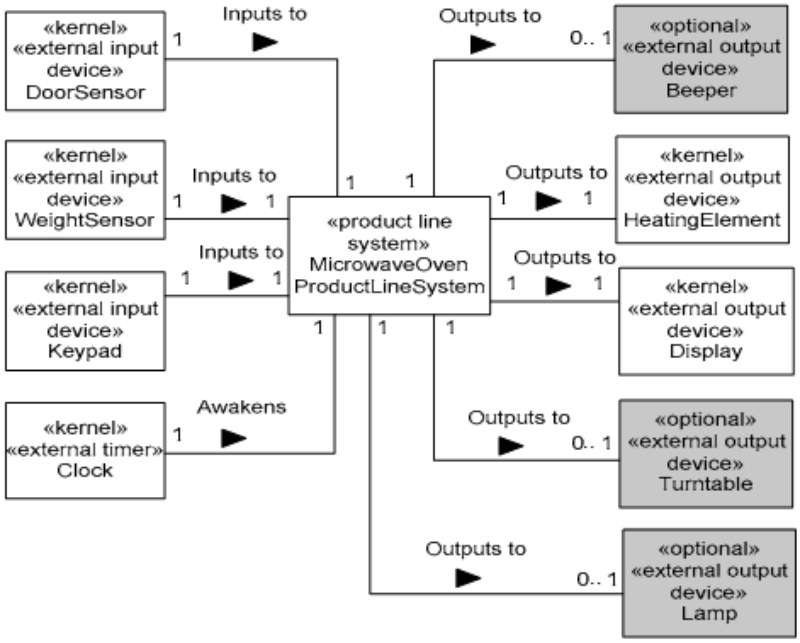


Figura 12: Diagrama estático com estereótipos de uma LPS - Fonte: (GOMAA, 2005)

- **Modelo dinâmico interativo –** Cada caso de uso do tipo *kernel*, *optional* ou *alternative* terá sua representatividade em um modelo dinâmico interativo, que é criado a partir de diagramas de colaboração ou de sequência oriundos do UML para representar a interatividade e as mensagens entre os objetos do sistema (Figura 13).

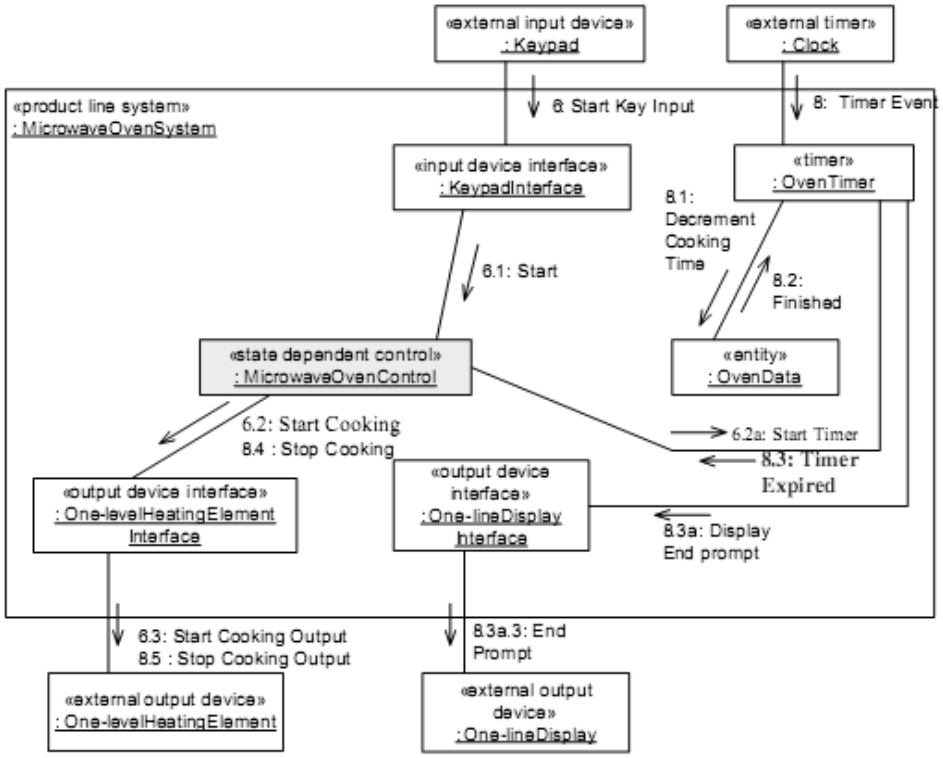


Figura 13: Diagrama dinâmico interativo do caso de uso Cook Food - Fonte: (GOMAA, 2005)

- **Modelo dinâmico da máquina de estados** – Consiste na criação de diagramas de estados finitos para as entidades dos tipos: *kernel*, *optional* e *variant*. Estes diagramas visam determinar os estados de cada entidade, ilustrando assim o ciclo de vida de cada uma delas e provendo um meio de interpretação para elas.

- **Modelo de dependência entre características e classes** – Consiste de um modelo que determina as dependências de cada classe com as características dos sistemas. Este modelo serve para auxiliar na identificação das classes que serão afetadas a partir da escolha de uma característica. A Tabela 2 ilustra esta dependência entre as classes e as características do sistema.

Tabela 2: Dependência entre classe e características – Fonte: (GOMAA, 2005)

Característica	Categoria da característica	Classe	Categoria de reuso da classe	Parametros da classe
Microwave Oven Kernel	Common	Door Sensor Interface	kernel	
		Weight Sensor Interface	Kernel-abstract-vp	
		Keypad Interface	Kernel-param-vp	
		Heating Element Interface	kernel-abstract-vp	
		Display Interface	kernel-abstract-vp	
		Microwave Oven Control	kernel-param-vp	
		Oven Timer	kernel-param-vp	
		Oven Data	kernel-param-vp	
		Display prompts	kernel-abstract-vp	
Light	Optional	Lamp Interface	optional	
		Microwave Oven Control	kernel-param-vp	light:Boolean
Turntable	optional	Turntable Interface	optional	
		Microwave Oven Control	kernel-param-vp	turntable:Boolean
Beeper	Optional	Beeper Interface	optional	
		Microwave Oven Control	kernel-param-vp	beeper:boolean
Minute Plus	Optional	Keypad Interface	kernel-param-vp	minuteplus:boolean
		Mirowave Oven Control	kernel-param-vp	minuteplus:boolean
		Oven Time	kernel-param-vp	minuteplus:boolean
One-line Display	default	One-line Display Interface	default	
Multi-line Display	alternative	Multi-line Display Interface	variant	
English	Default	English Display Prompts	default	
French	alternative	French Display Prompts	variant	
Spanish	alternative	Spanish Display Prompts	variant	
German	alternative	German Display Prompts	variant	
Italian	alternative	Italian Display Prompts	variant	

Etapa de Projeto para LPS

A etapa de projeto é dividida em Arquitetura de software e o projeto de software baseado em componentes.

- **Arquitetura de software** – Na arquitetura é determinado a estrutura arquitetural e os padrões de projeto[5] para o desenvolvimento da LPS. O diagrama para a representação arquitetural na notação UML é o diagrama de pacotes.

- **Projeto de software baseado em componentes** – Nesta fase a LPS é dividida em componentes e são definidas as interfaces de comunicação entre eles. Estes componentes são desenvolvidos para serem configuráveis. No âmbito geral de desenvolvimento esta fase auxilia na aplicação das definições dos padrões que foram definidos na arquitetura de software. A Figura 14 ilustra a composição de componentes para o sistema de controle de microondas.

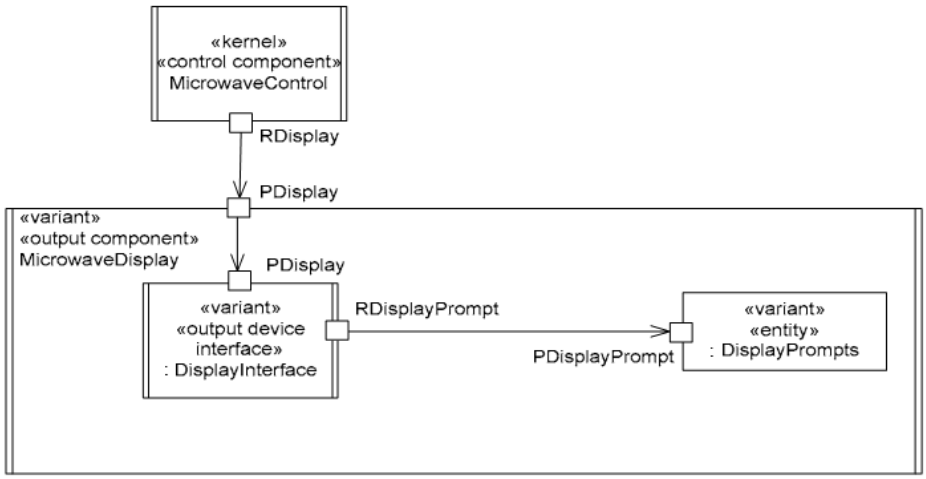


Figura 14: Composição de componentes do sistema de controle de microondas - Fonte: (Gomaa, 2005)

Engenharia da aplicação de software

Consiste no desenvolvimento das aplicações de uma LPS, utilizando os modelos desenvolvidos nas fases de: Requisitos, Análise e Projeto a fim de gerenciar estes modelos para então criar a família de produtos de uma LPS.

[5] Do Inglês: Design Patterns

Tomando como base a Figura 10, onde existem casos de uso que são opcionais, pode-se eliminar um dos casos de uso de estereótipo opcional como o `Cook Food with Recipe` , se este for eliminado as características que estavam ligadas a ele também serão, além disso, manipulando o modelo de características da Figura 11 e removendo as características opcionais: `Turntable`, `Minute Plus` e selecionando as alternativas: `English`, `Boolean Weight`, `Multi-line Display` e `One-level Heating`, temos como resultado um produto com as características/casos de uso descritos na Tabela 3.

Tabela 3: Cruzamento de informações de características x casos de uso, de acordo com a configuração selecionada.

Característica	**Categoria**	**Caso de uso**	**Categoria do caso de uso/ ponto de variação**	**Nome do Ponto de variação**
Microwave Oven Kernel	common	Cook Food	Kernel	
Light	optional	Cook Food	vp	Light
Beeper	optional	Cook Food	vp	Beeper
Multi-Line Display	alternative	Cook Food	vp	Display Unit
English	alternative	Cook Food	vp	Display Language
Boolean Weight	default	Cook Food	vp	Weight Sensor
One-level Heating	default	Cook Food	vp	Heating Element
Power Level	optional	Cook Food	vp	Power Level
TOD Clock	optional	Set Time of Day Display Time Fo Day	optional	
12/24 Hour Clock	parametrized	Set Time of Day Display Time Fo Day	vp	12/24 Hour Clock

Com as configurações do modelo de características, haverá mudanças no modelo de dependência entre classes e características, a Tabela 4, ilustra o impacto destas configurações.

Tabela 4: Configuração da aplicação – modelo de dependência entre classes e características

Característica	Categoria da característica	Classe	Categoria de reuso da classe	Parametros da classe
Microwave Oven Kernel	Common	Door Sensor Interface	kernel	
		Weight Sensor Interface	Kernel-abstract-vp	
		Keypad Interface	Kernel-param-vp	
		Heating Element Interface	kernel-abstract-vp	
		Display Interface	kernel-abstract-vp	
		Microwave Oven Control	kernel-param-vp	
		Oven Timer	kernel-param-vp	
		Oven Data	kernel-param-vp	
		Display prompts	kernel-abstract-vp	
Light	Optional	Lamp Interface	optional	
		Microwave Oven Control	kernel-param-vp	light:Boolean
		Microwave Oven Control	kernel-param-vp	turntable:Boolean
Beeper	Optional	Beeper Interface	optional	
		Microwave Oven Control	kernel-param-vp	beeper:boolean
Multi-line Display	alternative	Multi-line Display Interface	variant	
English	Default	English Display Prompts	default	

Com estas configurações, teremos a criação de um produto de software com as características selecionadas e ilustradas na Tabela 4. Então, para a criação de novos produtos de software, o gerente da LPS deverá configurar as características e casos de uso que serão considerados para a criação da família de produtos.

3.3.2 Processo de Gerenciamento de variabilidade para linha de produto de software

O PGV (OLIVEIRA JUNIOR, 2005) utiliza como base as atividades propostas por Van Gurp e Bosch (2001), a construção de modelos de características, proposta por Griss, Favaro e D'Alessandro (1998) e a construção de um modelo genérico de variabilidades, proposta por Becker (2003). O PGV segue as atividades que são ilustradas na Figura 15. Nesta figura, as elipses representam as atividades e os retângulos são os artefatos de entrada para estas atividades.

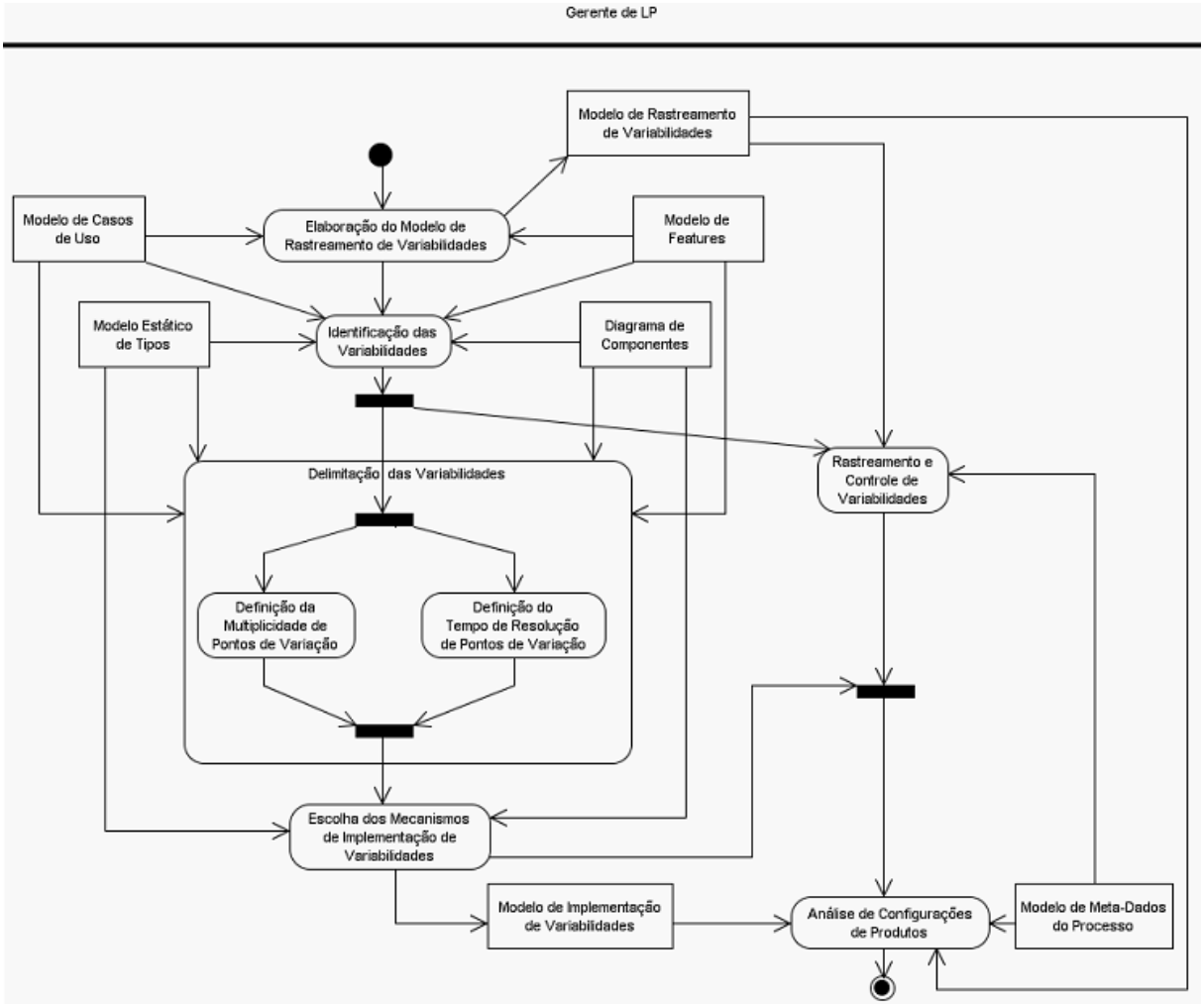


Figura 15:Atividades do Processo de gerenciamento de variabilidade- Fonte:(OLIVEIRA JUNIOR, 2005)

O processo é visto como um elemento que consome informações provenientes da atividade de desenvolvimento da LPS e também produz informações para tal atividade como, por exemplo, o modelo de casos de uso e o modelo estático de tipos com as variabilidades identificadas e delimitadas. O processo também produz informações para a atividade de geração de produtos específicos como, por exemplo, o modelo de rastreamento de variabilidades e o modelo de implementação de variabilidades. A Figura 16 mostra o relacionamento entre o gerenciamento de variabilidade e as atividades de desenvolvimento de LPS e geração de produtos. Nesta figura, o PGV está representado pelo retângulo hachurado em cinza e corresponde a um dos elementos da atividade de gerenciamento de LPS propostas pelo SEI (ver Figura 7) (OLIVEIRA JUNIOR, 2005).

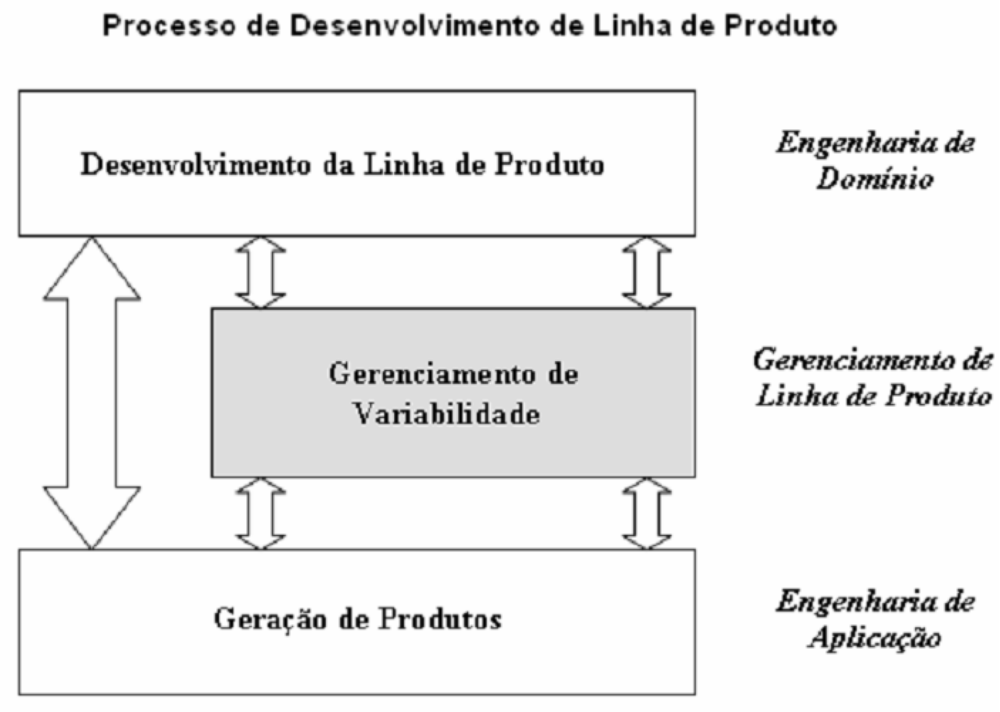


Figura 16: Processo de Gerenciamento de Linha de Produto utilizando o gerenciamento de variabilidades – Fonte (OLIVEIRA JUNIOR, 2005)

A Figura 17 mostra a interação entre as atividades de desenvolvimento da LPS, realizadas pelo engenheiro de LPS, e as do gerenciamento de variabilidade, realizadas pelo gerente de LPS. A figura mostra que ao final de cada atividade do desenvolvimento da LPS é realizada uma iteração do gerenciamento de variabilidade. Comparando esta figura com a abordagem PLP ilustrada na Figura 7, as atividades alinhadas verticalmente à esquerda correspondem ao ciclo superior esquerdo das atividades propostas pelo SEI (Figura 7), enquanto as demais atividades correspondem ao ciclo inferior. (OLIVEIRA JUNIOR, 2005).

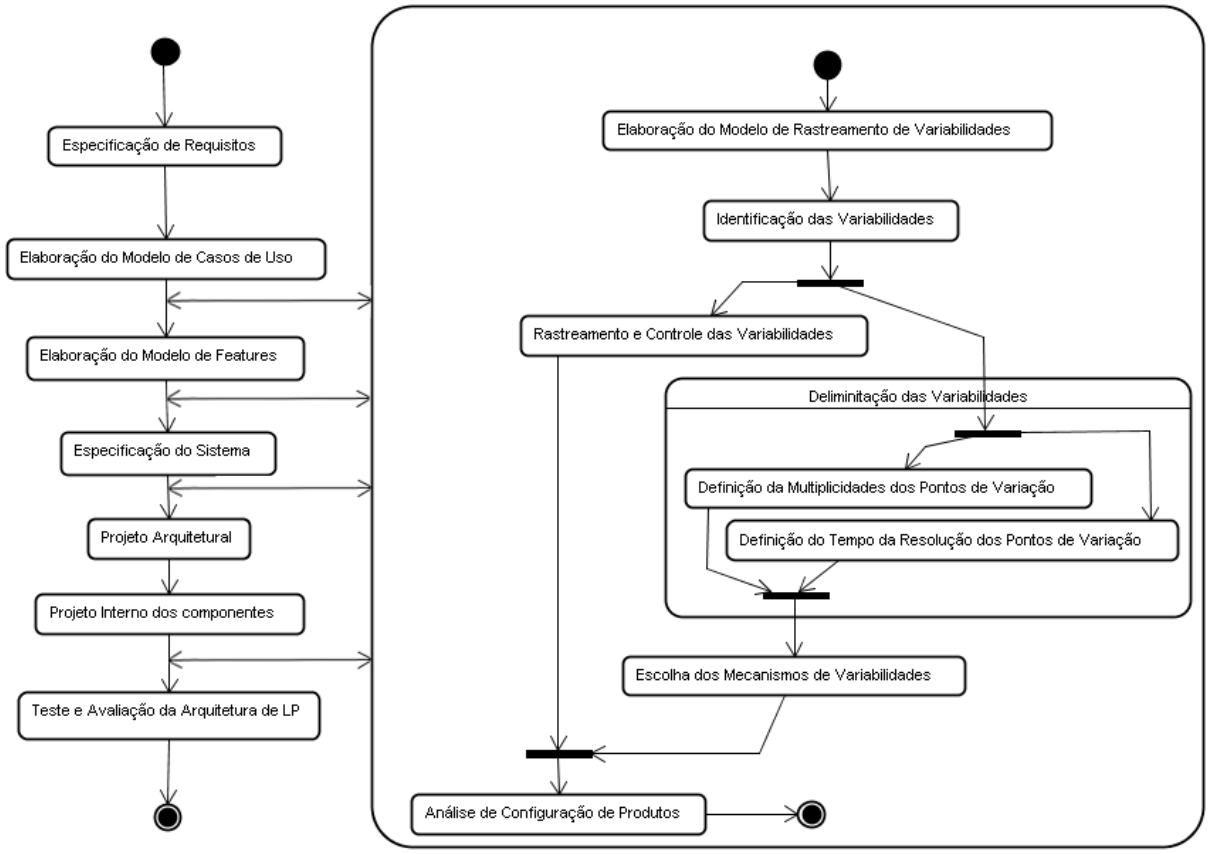


Figura 17: Interação entre as atividades e o Gerenciamento de Variabilidades - Fonte: (OLIVEIRA JUNIOR, 2005)

3.4 Considerações Finais

O Método PLUS é utilizado como base para a extensão do método OOWS principalmente na fase de especificação da Solução, pois os artefatos criados neste método são utilizados para a concepção de uma LPS para aplicações Web.

O PGV auxilia o gerente de LPS a coordenar os artefatos gerados pelo processo de desenvolvimento de software, sendo que estes são gerenciados de acordo com suas variabilidades para a criação de aplicações específicas de acordo com as características configuradas. Além disso, este processo permite melhorar na identificação dos elementos afetados a partir de uma característica selecionada.

No próximo capítulo, é tratado a proposta de extensão para o método OOWS utilizando os conceitos expostos nos capítulos 2 e 3.

Capítulo

4

SPL-OOWS – Uma extensão do método OOWS para linha de produto de software

Uma LPS tem por objetivo desenvolver aplicações para um determinado domínio, a partir de um núcleo de artefatos. O domínio de aplicações Web, por si só já possui elementos comuns que podem ser reutilizados e representados como um domínio semente a partir do qual características específicas de cada domínio de aplicação podem ser modeladas. O objetivo do método SPL-OOWS é a criação de uma infra-estrutura básica e padronização de etapas para possibilitar que aplicações Web sejam representadas como uma LPS. Para tal, estende o método OOWS (FONS, 2001) com conceitos de LPS.

Para apoiar a extensão do OOWS, buscou-se duas abordagens de LPS compatíveis com este método que são: Gomma (2005) e Oliveira Junior (2005). Essas abordagens são compatíveis com o OOWS porque têm etapas de desenvolvimento similares e também utilizam UML. Oliveira Júnior (2005) foi incorporado, pois possui um processo de gerenciamento de variabilidade mais rigoroso que o proposto por Gomaa (2005). Além dessas abordagens, o SPL-OOWS também utiliza o modelo de características proposto por Czarnecki et al. (2005). Este modelo incorpora importantes extensões, bem como possui uma ferramenta de apoio à diagramação.

Para ilustrar a descrição do método SPL-OOWS é utilizado como exemplo uma aplicação de um comércio eletrônico. Esta aplicação consiste de uma loja fictícia que vende livros pela Internet. Para comprar um livro, o usuário do sistema deve estar cadastrado e ter

adicionado o livro desejado à uma lista de compra (carrinho de compras). Quando, finalizar sua escolha, ele poderá concretizar a compra.

O método SPL-OOWS concentra-se na extensão das fases de desenvolvimento do método OOWS incluindo nessas conceitos de LPS. Cada uma dessas fases consiste de várias etapas. As extensões propostas concentram-se principalmente nas etapas de: modelagem conceitual, modelagem navegacional e de apresentação. A Figura 18 ilustra as etapas do método SPL-OOWS em que se pode ver as extensões propostas que são: elaboração do modelo de características ao longo das etapas da modelagem conceitual e acréscimo dos modelos de infra-estrutura básica na modelagem conceitual e a camada de infra-estrutura básica na arquitetura do desenvolvimento da solução.

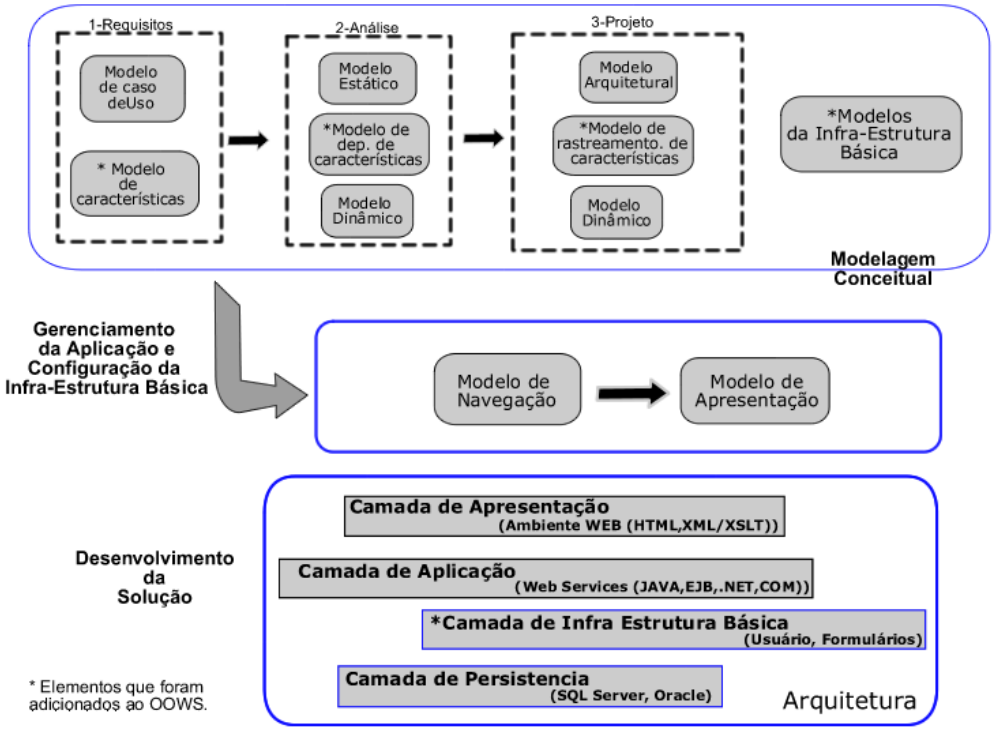


Figura 18: SPL-OOWS - Visão geral

Este capítulo apresenta cada uma das etapas do SPL-OOWS destacando as extensões propostas e os modelos elaborados em cada uma delas.

4.1 Definição da Infra-Estrutura Básica

A infra-estrutura básica consiste de elementos comuns para a concepção de uma aplicação Web para qualquer domínio. Esses elementos poderão ser configurados e gerenciados com a finalidade de serem utilizados para a criação de domínios específicos como portais, comércio eletrônico ou transações financeiras. A infra-estrutura básica pode ser estendida para cada domínio, constituindo assim uma hierarquia de LPS para aplicações Web.

O modelo de características que representa a infra-estrutura básica é apresentado na Figura 19 em um diagrama de característica (Czarnecki et al., 2005). Ele consiste de três características principais, uma que representa o Gerenciamento de Usuários, outra que representa o Gerenciamento de Formulários e o Gerenciamento de Banco de dados.

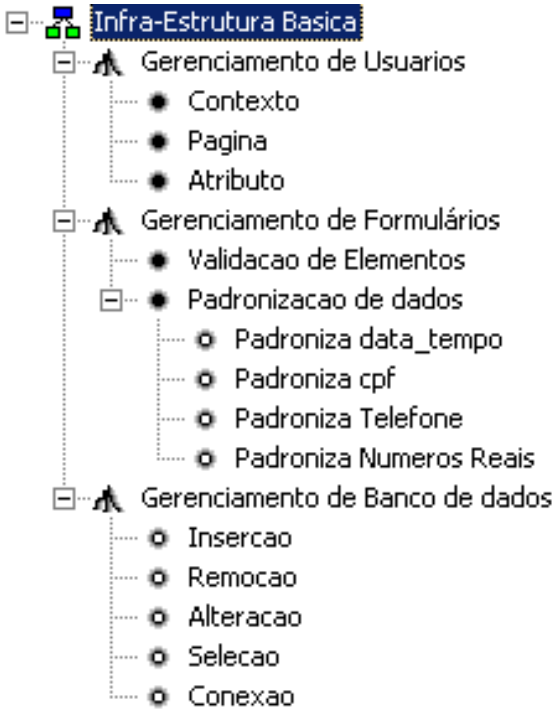


Figura 19: Infra Estrutura básica

O Gerenciamento de Usuários foi concebido utilizando os princípios do modelo de usuário do OO-Method (PASTOR et al., 2001), porém foi modificado para atender as necessidades de uma LPS. O Gerenciamento de Usuários é utilizado para

atribuir as permissões de acesso aos contextos navegacionais para o domínio da LPS. O Gerenciador de Formulários foi incluído para gerenciar os dados enviados a partir de um formulário e o Gerenciador de banco de dados é o responsável por realizar as operações no SGBD utilizando interfaces pré-definidas.

4.1.1 Gerenciamento de Usuário

O modelo de usuário do OO-Method (PASTOR et al., 2001) especifica quais são os tipos de usuário que acessarão o sistema e qual a visão de cada tipo em relação aos contextos navegacionais. SPL-OOWS propõe uma forma de gerenciamento do modelo de Usuário de modo a torná-lo reutilizável para ser configurado para qualquer domínio de aplicação. Esta proposta é compatível com o OOWS no qual aplicações Web são compostas de contextos navegacionais, os quais estão ligados aos usuários que acessam esses contextos, mesmo que eles sejam anônimos. O Gerenciamento de Usuários é representado por um diagrama de casos de uso, conforme ilustra a Figura 20. Este contém um usuário representando o administrador da aplicação, pois é ele quem atribuirá os contextos a cada grupo de usuário. As associações entre os usuários e os contextos representam a permissão ou não de acesso às operações de grupos de usuários ao contexto navegacional. O caso de uso `Gerenciar Usuário`, representado na Figura 20, controlará as atividades do usuário nos sistemas.

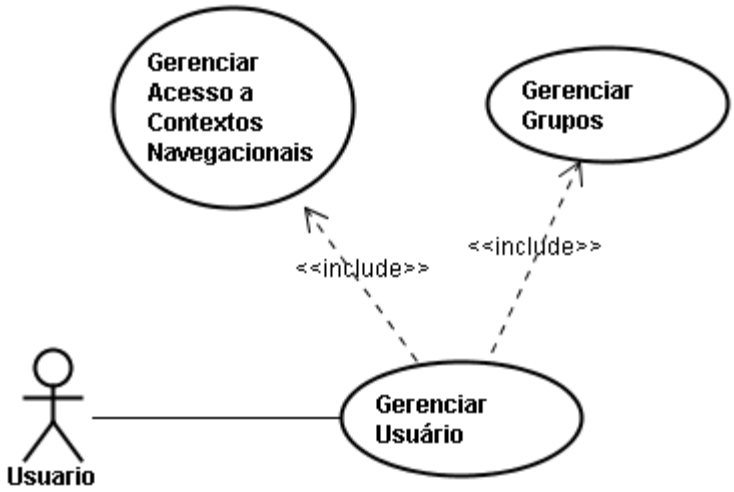


Figura 20: Casos de Uso do modelo de Usuário

O caso de uso `Gerenciar Acesso a Contextos Navegacionais` visa verificar se o usuário tem acesso aos contextos solicitados. Já o caso de uso `Gerenciar Grupos` controla quais grupos, o usuário da aplicação pertence, podendo assim permitir ou negar o acesso ao contexto navegacional pretendido.

Tabela 5: Descrição do caso de uso Gerenciar Usuário

Número: 01
Caso de Uso: Gerenciar Usuário.
Tipo: Obrigatório.
Objetivo: Controlar as atividades do usuário de acordo com suas permissões contidas nos grupos de usuários e configuradas nos contextos navegacionais.
Casos de Uso relacionados: Acessar Contextos Navegacionais.
Atores: Usuário registrado ou anônimo.
Pré-Condição: Usuário não está logado no sistema ou não possui permissões.
Pós-Condição: Usuário logado ou com permissões de acesso.

Tabela 6 : Desrição do caso de uso Gerenciar acesos a contextos navegacionais

Número: 02
Caso de Uso: Gerenciar acessos a contextos navegacionais.
Tipo: Obrigatório.
Objetivo: Permitir ou negar os acessos dos grupos de usuários dentro dos contextos navegacionais.
Casos de Uso relacionados: Gerenciar Usuário.
Atores: Usuário registrado ou anônimo.
Pré-Condição: Usuário acessa um contexto navegacional.
Pós-Condição: O usuário terá ou não permissão para acessar este contexto.

Tabela 7: Descrição do caso de uso Gerenciar grupos de usuários

Número: 03
Caso de Uso: Gerenciar grupos de usuários.
Tipo: Obrigatório.
Objetivo: Atribuir cada usuário a um grupo de usuários que terão acesso aos contextos do sistema.
Casos de Uso relacionados: Gerenciar Usuário, Gerenciar acessos a contextos navegacionais.
Atores: Usuário registrado ou anônimo.
Pré-Condição: Usuário não possui grupo de usuários associado.
Pós-Condição: O usuário terá um grupo de usuários associado.

Após a concepção do modelo de caso de uso, devem ser elaborados os diagramas de estado correspondente ao mesmo. A Figura 21 apresenta o diagrama de estados do caso de uso Gerenciar Usuário. Quando um usuário tenta acessar um contexto da aplicação Web, é preciso fazer a validação para saber se o usuário pode ou não ter acesso a este conteúdo. A função do gerenciamento de usuário é verificar as permissões do usuário e permitir ou não o acesso aos contextos do sistema. Esta atribuição é realizada assim que o usuário entra no sistema (realiza o login). As configurações do grupo conterão as informações de permissão de acesso aos conteúdos no contexto navegacional.

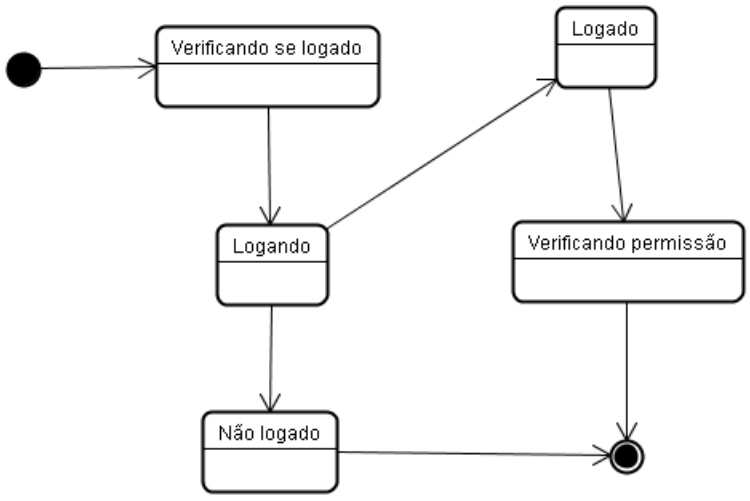


Figura 21: Diagrama de Estados do Caso de Uso Gerenciar Usuário

A Figura 22 mostra como é realizado o controle do acesso do usuário em relação ao contexto navegacional. Este controle consiste em consultar a base de dados de usuário relacionada a um grupo para então verificar se o usuário poderá ou não acessar o contexto desejado.

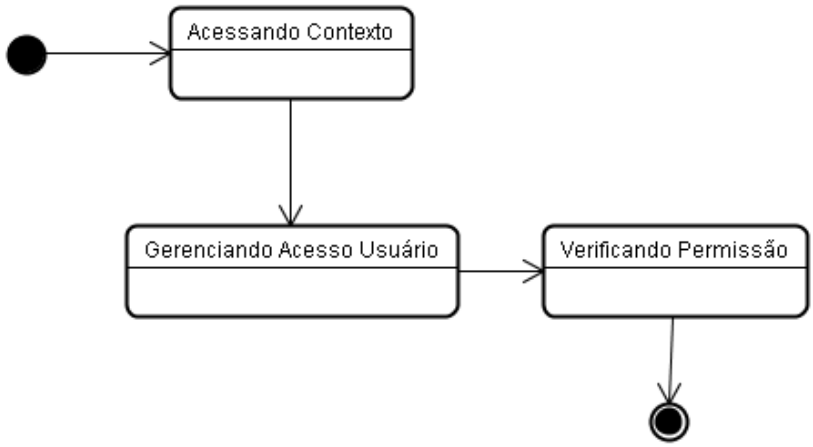


Figura 22: Diagrama de Estados do Caso de Uso Acessar Contextos navegacionais

Para o desenvolvimento do gerenciamento de usuário a partir da realização do levantamento de requisitos, foram identificadas quatro entidades principais: Usuário, Grupo de Usuário, Contexto Navegacional e Objeto do Contexto. A Figura 23 mostra o diagrama estático que representa os relacionamentos entre os objetos do modelo de Usuário.

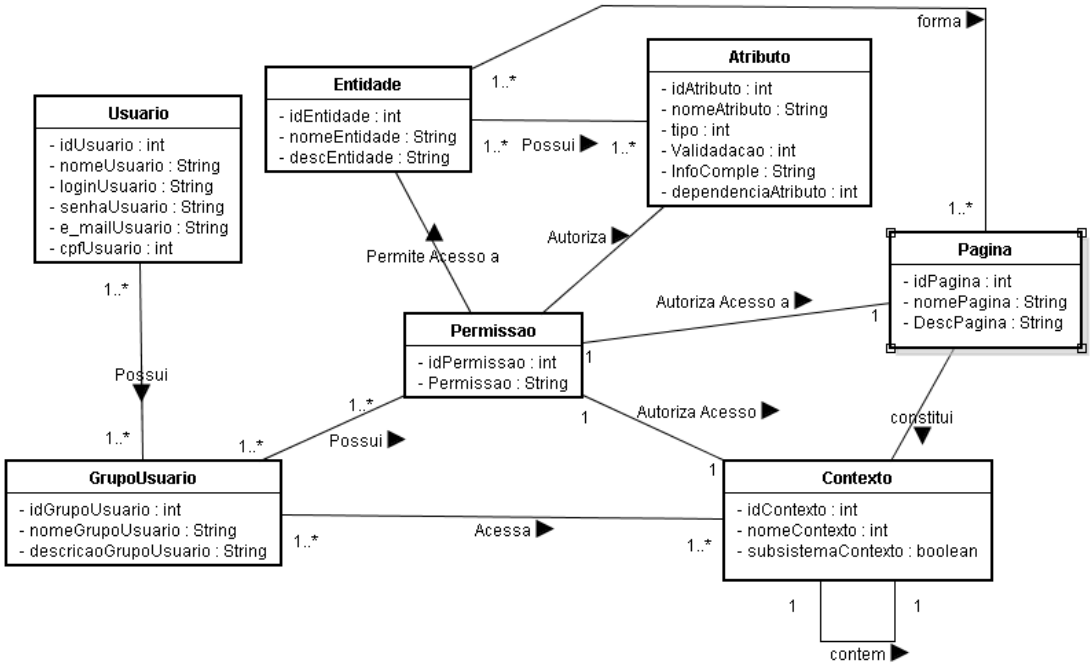


Figura 23: Modelo de Usuário Proposto

A classe `Usuário`, representa as informações dos usuários que acessam a aplicação Web e a classe `GrupoUsuario` manterá o cadastro de grupos de usuários que serão utilizados para gerenciar as permissões para mais de um usuário. Quando um Usuário está inserido em um `GrupoUsuario` ele possui as permissões e os acessos correspondente ao grupo. A classe `Permissão` é responsável por armazenar as atribuições de segurança de cada grupo em relação a: `Contexto`, `Página`, `Entidade` e `Atributo`. A classe `Contexto` representa os cadastros de todos os contextos existentes na aplicação. Cada `contexto` é composto por várias `Páginas` que por sua vez é formado por várias `Entidades` que possuem vários Atributos.

Desta forma, a ação da classe `Permissão` contempla todos os acessos possíveis dentro de uma aplicação, desta forma, o administrador do sistema poderá configurar cada acesso de cada elemento através do modelo de usuário.

A Figura 24 ilustra o diagrama de classes incluindo os métodos para cada classe.

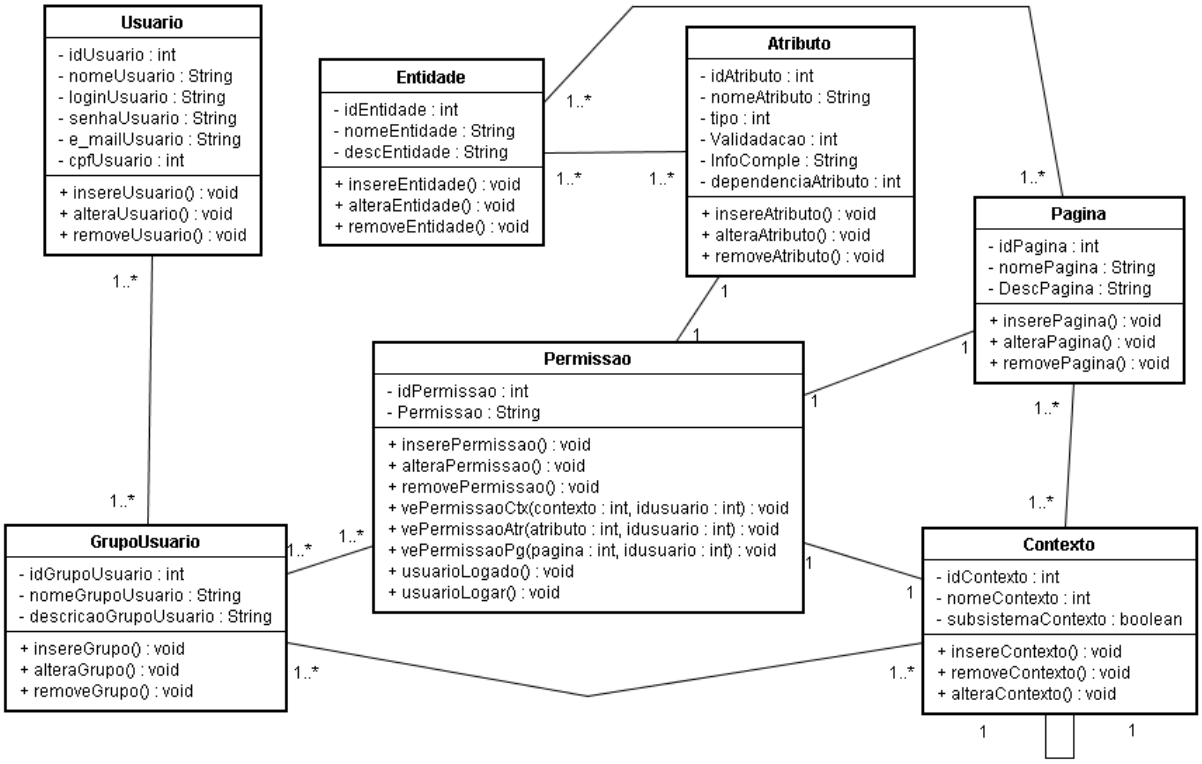


Figura 24: Diagrama de Classes do controle de usuários - completo

A partir dos diagramas de classes descritos, foram criadas as classes para o modelo de usuário, cada classe foi alocada dentro da camada da lógica do negócio (*Business Logic)*, e pode ser acessada por um ou mais elementos contidos na camada de Interface do negócio (*Business Facade*). Assim sendo, foram criados os métodos para cada classe para que fosse possível a configuração desta infra-estrutura básica.

Além destas implementações, para a classe Permissão foram criados métodos para realizar o controle dos acessos nas páginas, contextos e atributos que as camadas de navegação e apresentação oferecem. A interface controladora é composta dos seguintes métodos:

UsuarioLogar(login, senha) – Este método faz a conexão com o banco de dados e verifica se o usuário que está tentando acessar o sistema utilizando seu login e senha possui permissão para entrar no mesmo.

UsuarioLogado(sessao) – Método que verifica se o usuário está logado, retorna 1 caso positivo e 0 (zero) caso negativo.

VePermissaoCtx(contexto, idusuario) – Verifica se o usuário que está logado possui permissão para acessar o contexto que está sendo passado como parâmetro.

VePermissaoAtr(atributo, idusuario) – Verifica se o usuário logado possui permissão para acessar o atributo de alguma classe, este atributo pode ser um campo que determinados tipos de usuário não podem ter acesso.

VePermissaoPg(pagina, idusuario) – Verifica se o usuário logado possui permissão para acessar uma página que está sendo passada como parâmetro.

A partir destes métodos é possível realizar o controle dos usuários nas páginas, contextos e atributos, basta incluir a chamada do método antes da inclusão da página, contexto ou atributo. Para visualizá-los na forma como foram implementados, consulte o Apêndice - B.

4.1.2 Gerenciamento de Formulários

Quando uma aplicação Web necessita obter informações dos usuários, elas são obtidas por meio de formulários que gravam os dados em campos. Em uma etapa posterior esses dados são gerenciados pela lógica da aplicação. O Gerenciamento de Formulários proposto pelo SPL-OOWS visa generalizar o fluxo de dados entre os usuários e a lógica da aplicação, uma vez que essa funcionalidade está presente em todas as aplicações Web.

O Gerenciamento de Formulários utiliza recursos de validação de elementos e um controlador de dados. A validação dos elementos consiste em configurar a obrigatoriedade e os tipos de dados que estão sendo enviados para a aplicação de modo a filtrar os dados para que estes sejam compatíveis com o sistema. O controlador de dados possui a função de adaptar os dados que estão sendo enviados ao sistema para que estes sejam compatíveis com os padrões do Sistema de Gerenciamento de Banco de Dados (SGBD) em uso. Por exemplo: se no campo Data de Nascimento o usuário do sistema digitou “14/05/1985”, para que esta informação seja persistida, primeiro o validador de elementos verifica se a data digitada é válida, em seguida, o controlador de dados transformará esta data em um padrão aceito pelo

SGBD. Supondo que o padrão de data do SGBD seja: “yyyy-mm-dd”, onde yyyy é equivalente ao ano, mm equivale ao mês e dd equivale ao dia. Cabe ao controlador de dados transformar a data que está no padrão dd/mm/yyyy para yyyy-mm-dd.

O gerenciamento de formulários consiste no desenvolvimento de um componente que captura as informações enviadas pelos formulários e transformá-las em um padrão configurado pelo analista. Os métodos a seguir ilustram os tipos de funcionalidades que podem ser incluídas no gerenciador de formulários:

ConfiguraCPF(CPF, padrao) – Método responsável por padronizar a inserção de CPF por intermédio do parâmetro “padrao”.

ConverteDataBD(data, padraoBD) – Método que converte a data enviada como parâmetro para o padrão aceito pelo banco de dados através do parâmetro “padraoBD”.

ConverteParaCaixaalta(texto) – Converte o texto passado por parâmetro para caixa alta.

ConfiguraCEP(CEP, padrao) – Configura o padrão do CEP de acordo com o padrão pré-estabelecido.

ConfiguraTelefone(Tel, padrao) – Configura o Telefone enviado como parâmetro de acordo com o parâmetro “padrao”.

4.1.3 Gerenciamento de Banco de dados

O gerenciamento de banco de dados trata das informações que serão armazenadas no banco de dados através de uma interface única de instanciação, sendo que esta interface proverá o acesso à base de dados independente da tecnologia de banco de dados que será adotada pelo desenvolvedor.

A interface do gerenciamento de banco de dados fornecerá as seguintes operações: Inserção (INSERT), Remoção (DELETE), Alteração (UPDATE) e Visualização (SELECT), para realizar as operações de banco de dados. O desenvolvedor deverá definir qual operação deseja realizar e qual a entidade do banco de dados sofrerá as alterações. Por exemplo, se existe a entidade no banco chamada Cliente, é necessário inserir informações como Nome,

Cpf e Rg. A operação utilizando o gerenciador de banco de dados seria similar a: Insere('Cliente', 'Nome,Cpf,Rg', 'Bruno,11111111111,111111'), e desta maneira o gerenciador de banco de dados realizaria a operação de acordo com o banco que estivesse sendo utilizado.

Os métodos que constituem o gerenciador de banco de dados são:

Insere(tabela, campos, dados) – Método responsável por realizar as inserções dos dados no banco de dados recebendo como parâmetros o nome da tabela "tabela", um vetor composto dos nomes dos campos "campos" e um vetor com o os dados que deverão ser introduzidos no banco de dados.

Remove(tabela, camposChave, valorChave) – Método que tem a função de realizar a exclusão da linha especificada pelos campos chave "camposChave" e valores "valorChave" destes da tabela passada pelo parâmetro "tabela".

Altera(tabela, campos, dados, condicao) – Realiza as alterações dos campos passados como parâmetros como "campos" e incluindo os valores destes campos através do vetor "dados" a partir da condição "condicao".

Conecta(host, usuário, senha) – Faz a conexão da aplicação com o sistema gerenciador de banco de dados, sendo que recebe como parâmetro o endereço do SGBD e o usuário e senha do usuário autorizado a acessar os dados.

4.2 Modelagem Conceitual

A etapa de modelagem conceitual do método OOWS consiste no levantamento de requisitos e elaboração dos modelos dinâmico, semântico e funcional. No método SPL-OOWS é utilizada a notação UML estendida para LPS do método PLUS (GOMAA, 2005). Para o desenvolvimento da modelagem conceitual do SPL-OOWS foram utilizados modelos: o modelo de caso de uso (diagrama de casos de uso), modelo estático (diagrama de classes), modelo dinâmico (diagrama de colaboração, diagrama de sequência), diagrama arquitetural, e modelo Arquitetural (diagrama de componentes). Além desses modelos, existe a necessidade de gerenciamento das variabilidades da LPS que é representada pela notação de Oliveira Junior (2005) por meio do modelo de dependência de características.

A modelagem conceitual tem três etapas: Requisitos, Análise e Projeto. Em Requisitos são desenvolvidos os modelos de caso de uso e o de características. Em Análise são desenvolvidos os modelos: estático, dinâmico interativo e de dependência de características. Em projeto são desenvolvidos o modelo arquitetural, o diagrama de seqüência, de classes e de componentes mais o modelo de rastreamento das variabilidades.

4.2.1 Requisitos

A etapa de requisitos tem por objetivo capturar e representar os principais conceitos e funcionalidades do sistema, assim como os requisitos não-funcionais. De acordo com Gomaa (2005), no desenvolvimento de uma LPS, devem ser desenvolvidos os modelos de caso de uso e características.

Modelo de caso de uso

O modelo de caso de uso do sistema exemplo do comércio eletrônico é ilustrado na Figura 25 em um diagrama de caso de uso com os estereótipos propostos por Gomaa (2005).

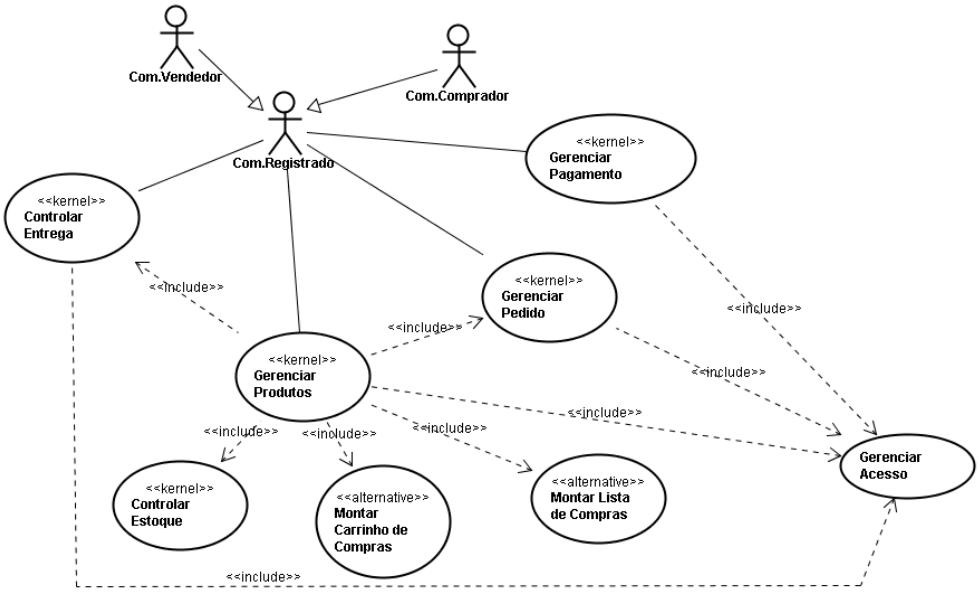


Figura 25: Modelo de caso de uso do sistema de comercio eletrônico

A seguir tem-se uma descrição sumária de cada caso de uso:

- Controlar Entrega - gerenciar a entrega de acordo com a solicitação do cliente. Esta entrega pode ser via transportadora ou correios.
- Gerenciar Produtos - controlar o cadastro de produtos no mercado eletrônico.
- Controlar Estoque - controlar a entrada e saída de produtos.
- Gerenciar carrinho de compras - incluir produtos para efetuar a compra.
- Gerenciar lista de compras – compor uma lista de produtos desejados com o intuito de criação de um orçamento.
- Gerenciar Pagamento - gerenciar o pagamento de um pedido, podendo assumir qual a forma de pagamento e parcelamento da compra.
- Gerenciar pedido - armazenar as informações dos produtos de um pedido, podendo finalizar a venda ou cancelar o pedido.
- Gerenciar Acesso - gerenciar o acesso de cada tipo de usuário permitindo ou não o acesso a determinados conteúdos na aplicação.

Modelo de características

Após as descrições dos casos de uso, o próximo passo é a elaboração do modelo de características da aplicação, cuja estrutura foi apresentada na seção 3.2.

Na Figura 26, a característica `Estoque Produtos` é uma característica obrigatória, enquanto que `Carrinho de Compra` é uma característica opcional.

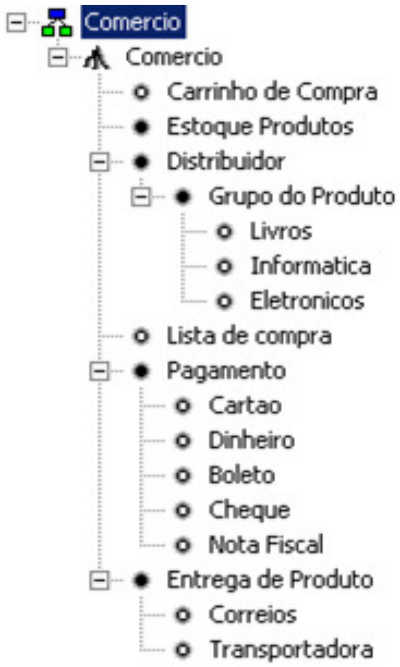


Figura 26: Modelo de características do sistema de comércio eletrônico

4.2.2 Análise

A etapa de análise tem como objetivo manter uma especificação dos requisitos precisa usando a linguagem dos desenvolvedores. Esta etapa foi adaptada a partir dos métodos de Olveira Junior (2005) e Gomaa (2005), em que são gerados os modelos estático, de dependência de características – rastreamento das variabilidades, o modelo dinâmico de estados e o modelo dinâmico interativo.

Modelo dinâmico de estados

O modelo dinâmico de estados representa os estados de um caso de uso e como se dá a transição desses estados. A partir deste modelo será especificado o modelo estático que compõe a etapa de análise.

No desenvolvimento deste modelo é importante que o desenvolvedor da LPS observe as alternativas especificadas no modelo de características, pois elas influenciam no modelo dinâmico de estados, à medida em que se escolhe ou não determinadas características, o fluxo de informações ilustrado neste modelo será afetado, necessitando de alterações.

A Figura 27 ilustra os estados relativos ao caso de uso `Controlar Entrega`. O estado `Modificando Estoque` representa que ao efetuar a venda e iniciar o processo de entrega faz-se necessário a atualização do estoque, em seguida inicia-se o processo de gerenciamento da entrega que é responsável por gerenciar as formas de envio da encomenda (Correios ou Transportadora). O último estado, `Recebendo Produto`, representa a conclusão do controlar entrega, onde o produto é entregue ou não (caso haja algum problema de entrega) ao cliente.

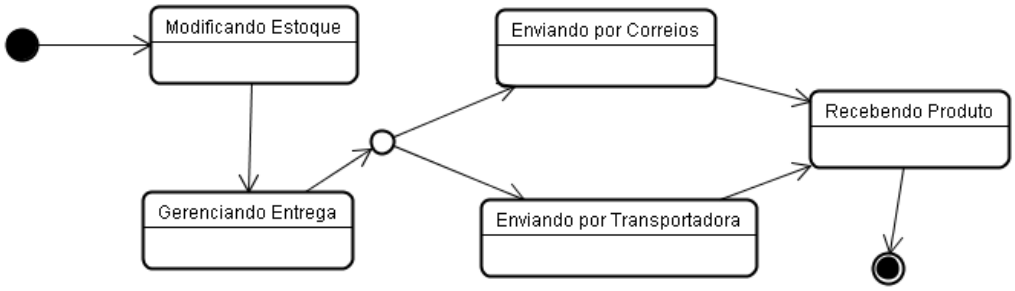


Figura 27: Diagrama de Estados do Caso de Uso Controlar Entrega

A Figura 28 representa os estados do caso de uso `Gerenciar Pagamento`. A partir do pedido criado pelo cliente, este escolhe qual será a forma de pagamento de seu pedido (Boleto, Cheque, Dinheiro ou Cartão).

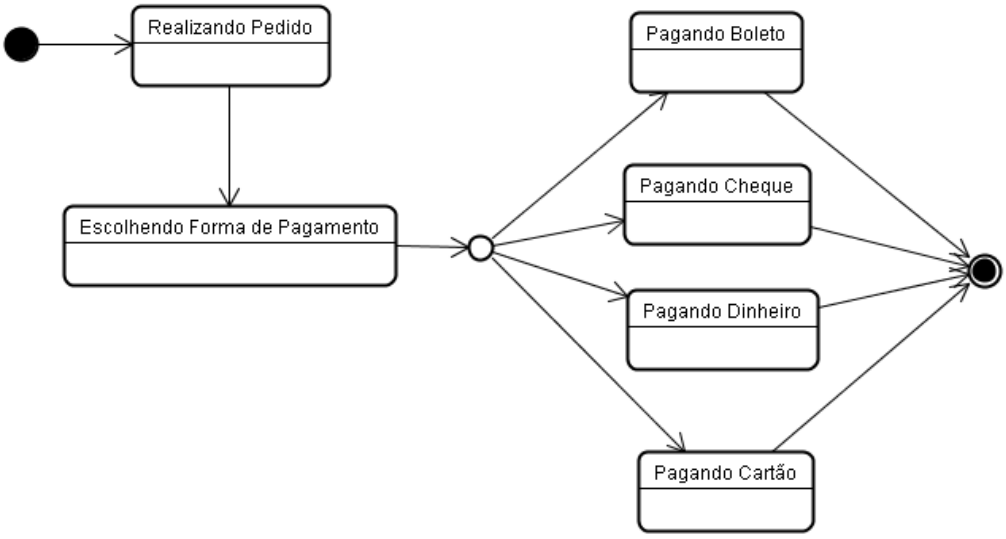


Figura 28: Gerenciar Pagamento

Modelo Estático

A modelagem dos estados de cada caso de uso auxilia na identificação das classes e seus relacionamentos que farão parte do sistema, obtendo-se assim o modelo estático. Este modelo representa a estrutura estática da família de produtos, incluindo os estereótipos de cada classe e os possíveis pontos de variação. A Figura 29 representa o modelo estático do sistema de comércio eletrônico.

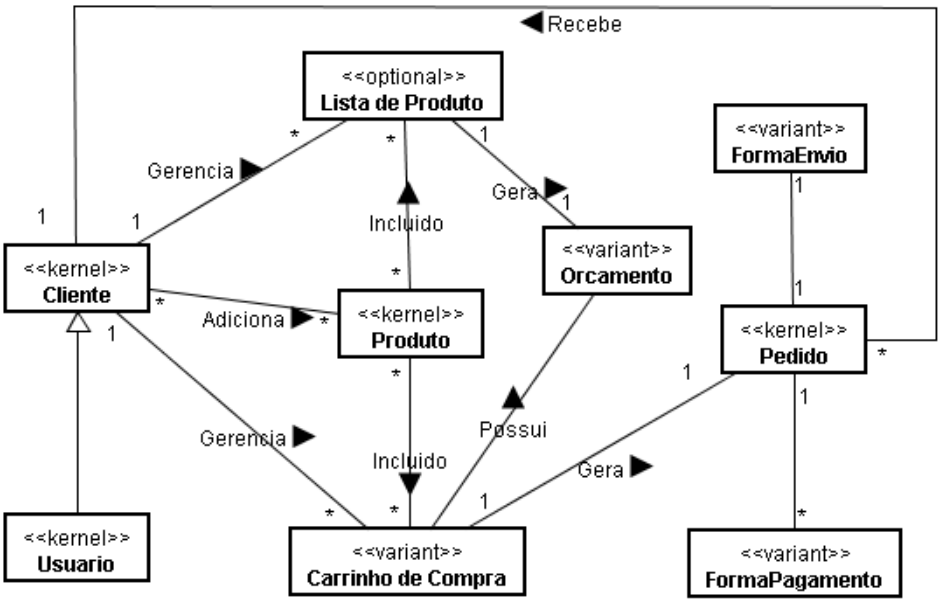


Figura 29: Modelo Estático - e-commerce

O modelo estático é concebido de acordo com os estereótipos de classes definidos por Gomaa (2005), que classifica as classes como sendo: *kernel*, *optional* ou *variant.* As classes `Cliente`, `Produto` e `Pedido` são *kernel* pois possuem papel fundamental para um comércio eletrônico, elas estarão presentes em qualquer aplicação relacionada a este domínio. A classe `Lista de Produto` é *optional* pois não é obrigatório o uso desta em um comércio eletrônico, a lista de produtos tem o papel de criar listas sem a intenção de compra, somente com o intuito de criar um orçamento ou uma lista de desejos. As classes com o estereótipo *variant* são classes que possuem pontos de variação e mudam de acordo com a aplicação a ser desenvolvida, são elas: `Carrinho de compra`, `Orçamento`, `Forma de Envio` e `Forma de Pagamento`. A Classe `Carrinho de Compra` é variável, pois pode possuir diversas formas de configuração, como: a forma de inserção dos produtos no carrinho, o que pode implicar para o estoque a retirada de um produto do estoque para o carrinho ou a não alteração no estoque. A classe `Orçamento` variará de acordo com o que for selecionado para a aplicação, como: a seleção ou não da `Lista de Produto`, a forma como o `Carrinho de Compras` será configurado. A classe `FormaEnvio` será diferente

para cada região ao qual o comércio eletrônico será aplicado, pois a forma de cálculo do frete varia de país para país.

Modelo dinâmico interativo

Após o desenvolvimento do modelo estático, é necessário criar o modelo dinâmico interativo de cada caso de uso especificado na etapa de requisitos. Este modelo representa a troca de mensagens entre as classes do sistema para concretizar um caso de uso. A Figura 30 ilustra o modelo dinâmico interativo do caso de uso `Gerenciar Entrega`.

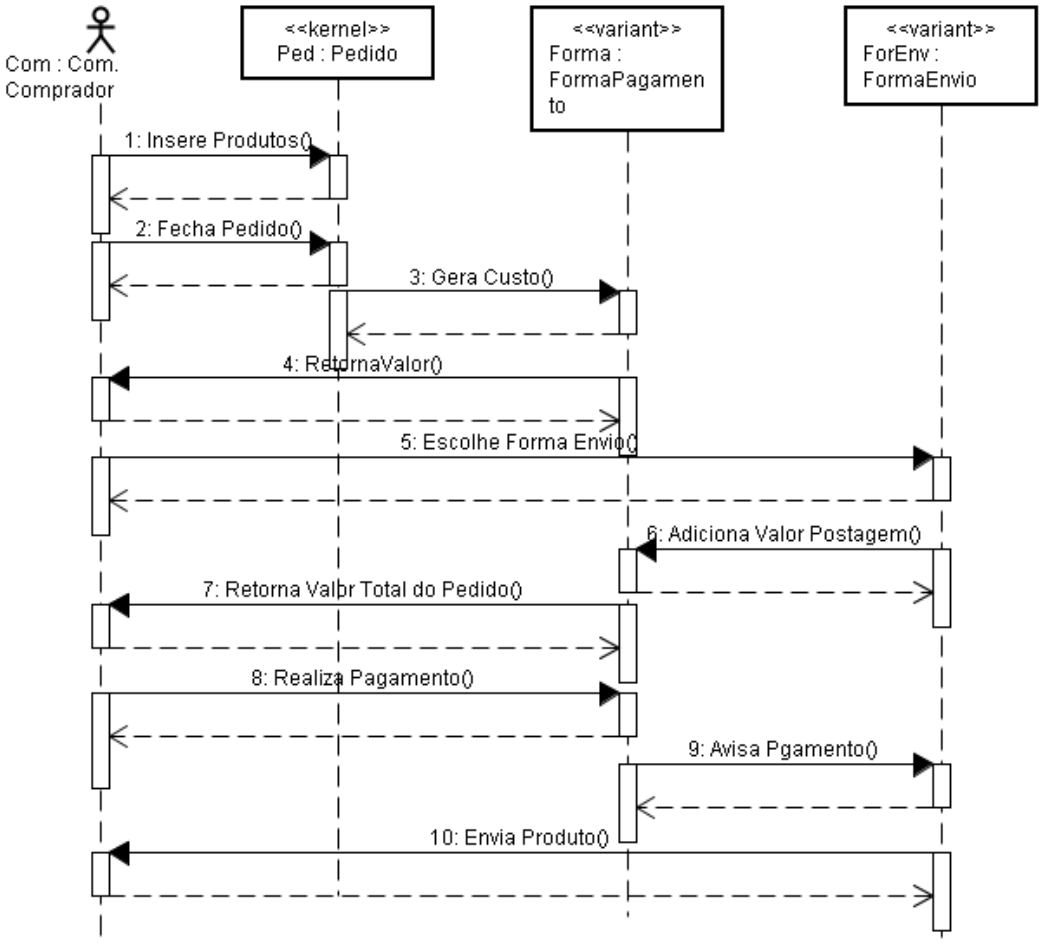


Figura 30: Modelo dinâmico Interativo - Gerenciar entrega de um sistema de comércio eletrônico

O modelo dinâmico interativo é representado pelo diagrama de seqüência. Para o caso de uso `Gerenciar Entrega` temos as seguintes mensagens: o comprador insere os produtos (etapa 1) que deseja comprar em um `Carrinho de Compras`. Ao concluir a

escolha dos produtos, o cliente fecha o pedido (etapa 2) que terá um custo e o valor do pedido é retornado ao cliente (etapas 3 e 4). Em seguida, chega o momento do cliente selecionar a forma de envio dos seus produtos (etapa 5). Após esta escolha, o valor do frete é incluído no pedido (etapa 6) e é mostrado ao cliente (etapa 7) que escolherá a forma de pagamento e efetuará o pagamento (etapa 8). O sistema ao receber a informação do banco ou agência de cartões de que o pagamento foi efetivado, envia um aviso de que os produtos podem ser enviados (etapa 9), assim os produtos são enviados ao cliente (etapa 10).

Modelo de rastreamento das variabilidades

O modelo de rastreamento das variabilidades representa o cruzamento entre as características e os casos de uso desenvolvidos. Esse cruzamento possibilita a identificação de artefatos da LPS. Com isso, é possível identificar quais casos de uso que são afetados quando uma característica é selecionada. A partir dos casos de uso afetados, é necessário realizar alterações nos modelos dinâmico e interativo, refazendo as conexões entre os estados para manter a coerência entre este modelo e o diagrama de casos de uso. A Tabela 8 ilustra o modelo de dependência de características do sistema de comércio eletrônico.

Tabela 8: Modelo de rastreamento de variabilidades do comércio eletrônico

Características/Casos de Uso	Controlar Entrega	Gerenciar Produtos	Controlar Estoque	Gerenciar Pagamento	Gerenciar Pedidos	Gerenciar Acesso
Carrinho de Compra	•	•	•	•	•	•
Estoque Produtos	•	•	•			•
Distribuidor		•				•
Grupo do produto		•				•
Lista de Compra		•			•	•
Pagamento				•		•
Cartão				•		•
Dinheiro				•		•
Boleto				•		•
Cheque				•		•
Nota Fiscal				•		•
Entrega de Produto	•				•	•
Transportadora	•				•	•
Correios	•				•	•

Após a etapa de análise, inicia-se o Projeto, esta fase consiste em definir qual a arquitetura do sistema, assim como quais serão os componentes utilizados e suas conexões.

4.2.3 Projeto

Nesta etapa, é realizado o mapeamento dos modelos desenvolvidos na análise para modelos que se aproximam da implementação da aplicação. Assim, são desenvolvidos os seguintes diagramas: modelo arquitetural, diagramas de seqüência, diagrama de classe com os métodos e atributos e o diagrama de componentes. Além desses artefatos, é nesta etapa que se faz o gerenciamento das variabilidades a partir do modelo de rastreamento de variabilidades.

Modelo Arquitetural

O modelo Arquitetural do método OOWS é representado por três diagramas: O diagrama de pacotes, diagrama de seqüência e o diagrama de classes.

Diagrama de pacotes

O diagrama de pacotes das aplicações é criado de acordo com a arquitetura de implementação proposta por Fons et al. (2001) no método OOWS. Porém, é adicionado o pacote da infra-estrutura básica que contém o gerenciador de usuários (`Gerenciador Usuario`), o gerenciador de formulários (`Gerenciador Form`) e o Gerenciador de Banco de dados (`Gerenciador BD`). A Figura 31 ilustra o diagrama arquitetural do método SPL-OOWS para a aplicação de comércio eletrônico.

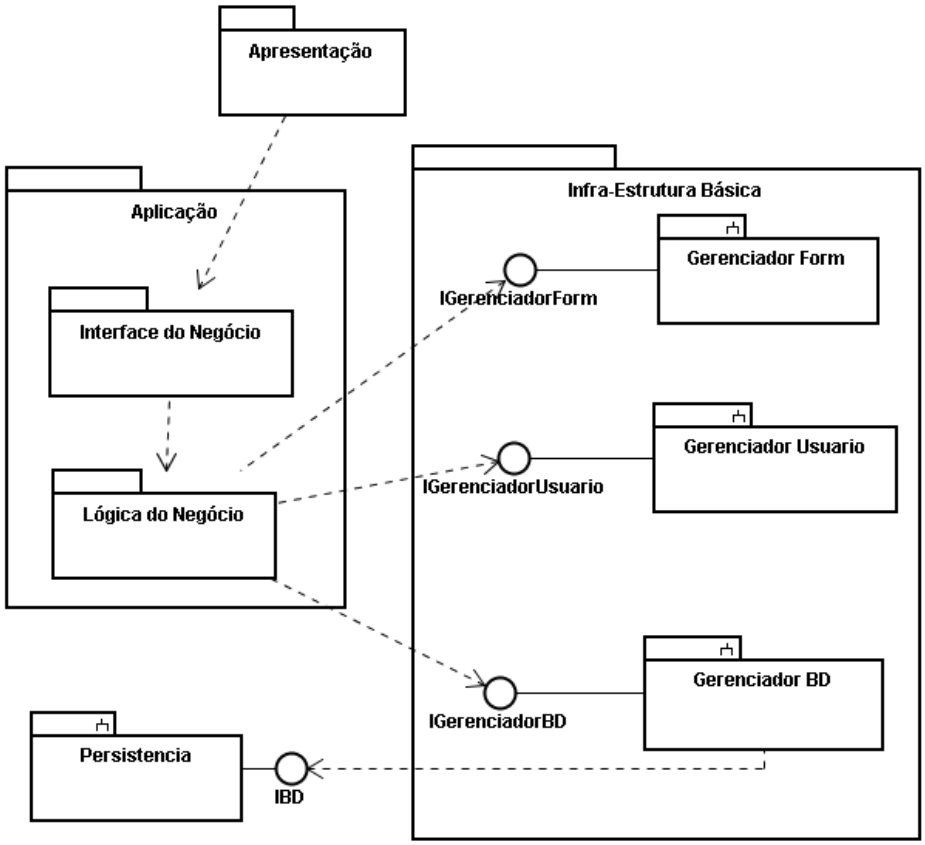


Figura 31: Diagrama Arquitetural de Implementação.

O diagrama arquitetural representa que a camada de `Apresentação` interage com o pacote `Interface do Negócio` que por sua vez troca informações com a `Lógica do Negócio`. Este pacote trata da lógica do negócio e caso necessário envia ou extrai dados da `Infra-Estrutura Básica`, que por sua vez possui os gerenciadores básicos,

`Gerenciador Form`, `Gerenciador BD` e `Gerenciador Usuário`, que podem ser acessados através de suas interfaces de comunicação com os objetos: I`GerenciadorForm`, I`GerenciadorBD` e I`GerenciadorUsuário` respectivamente.

Diagrama de seqüência

Os diagramas de seqüência definem as seqüências das operações no sistema, a diferença deste diagrama com o diagrama dinâmico interativo é que este possui o nome dos métodos de cada classe para a realização da seqüencia. Sendo assim, o diagrama dinâmico interativo na fase de projeto (Figura 30) terá a disposição descrita na Figura 32.

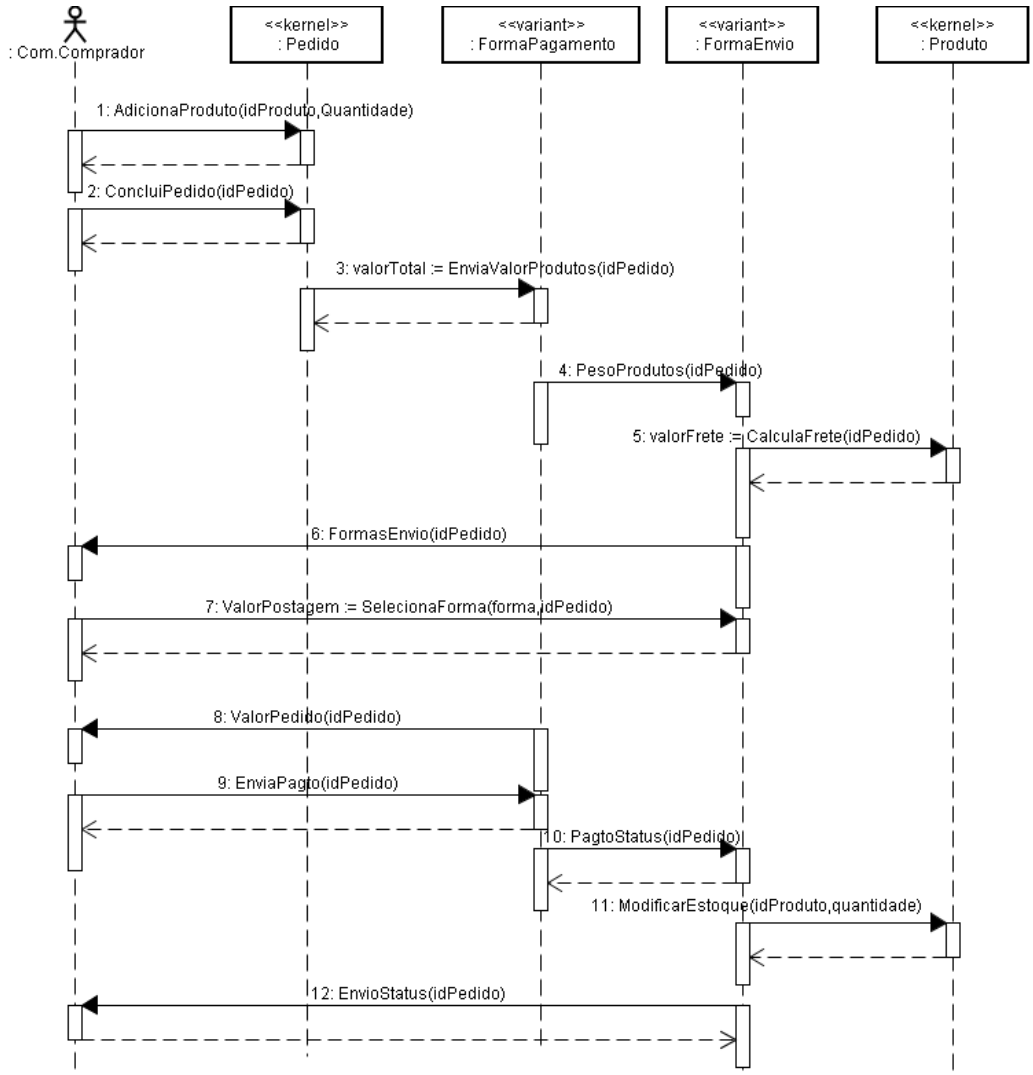


Figura 32: Diagrama de seqüência da fase de projeto do caso de uso Gerenciar Entrega do sistema de comércio eletrônico

O cliente realiza a operação de adição de produtos no carrinho de compras (1), então a conclusão da escolha dos pedidos ocorre e o usuário opta por finalizar o pedido (2). Então o valor da venda de cada produto é enviado ao sistema (3) que por sua vez realiza a operação de soma dos valores do pedido (4) que é enviado a forma de pagamento (5). Então é necessário calcular o frete e as formas de envio dos produtos (6 e 7). Então o cliente recebe as informações sobre as formas de pagamento, assim como as formas de envio dos produtos (8). Quando o cliente realiza o pagamento, então o sistema recebe a confirmação do cliente e confirma no banco se o pagamento ocorreu (9). Caso o pagamento tenha ocorrido (10), inicia-se o processo de envio dos produtos que envolve a atualização do estoque (11) e enfim a remessa deste produto ao cliente (12).

Diagrama de classe

Na etapa de Projeto, o diagrama de classe com os atributos e métodos mostra como será a estrutura de implementação da solução. A Figura 33 mostra como o diagrama de classes da aplicação de um comércio eletrônico.

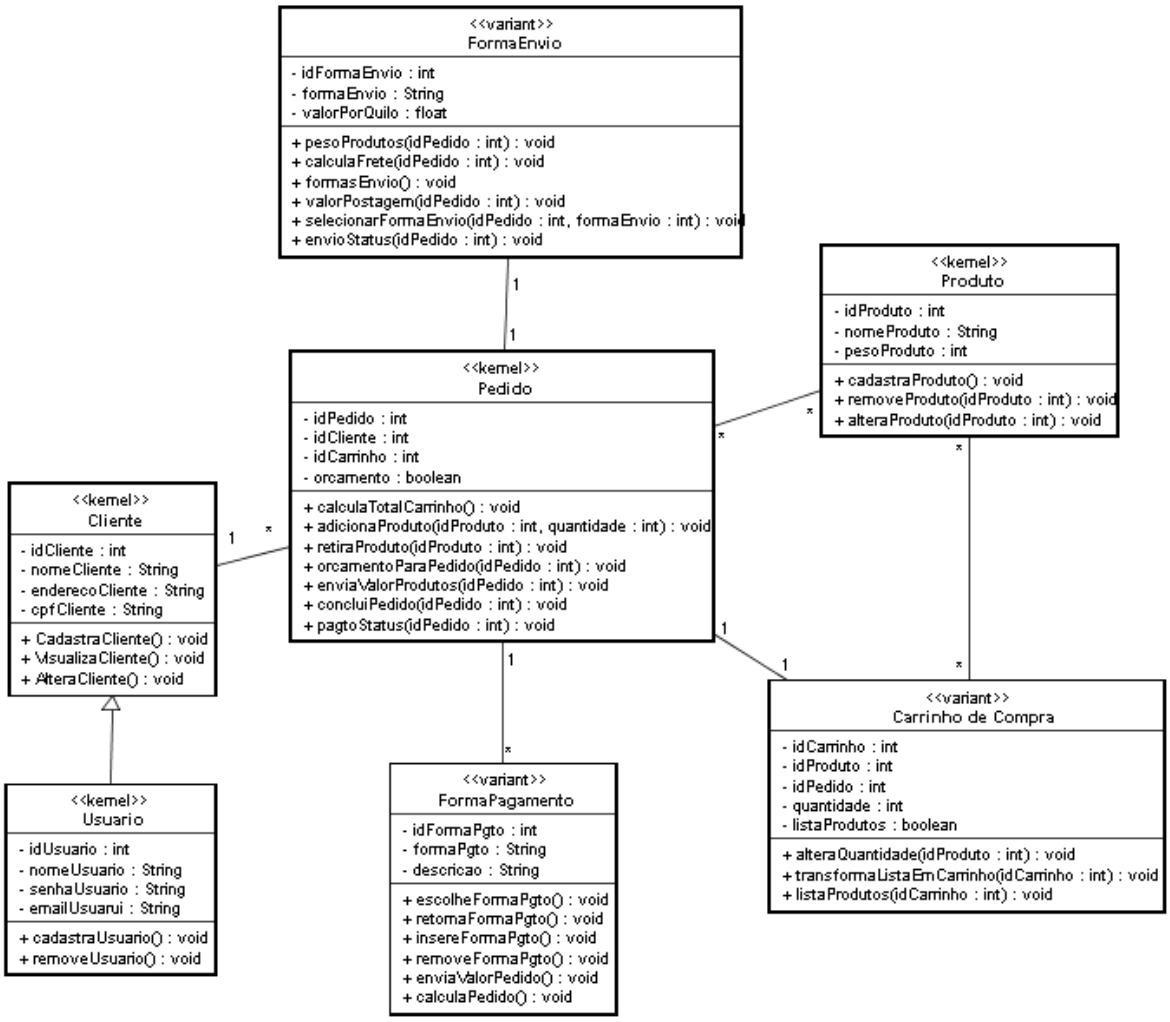


Figura 33: Diagrama de Classes de um comércio eletrônico

O diagrama de classes difere do modelo estático, pois para o desenvolvimento da solução há necessidade de normalizar algumas situações para sanar problemas que podem ser gerados como a duplicidade de informações. Por exemplo, o objeto de negócio `Lista de produtos` possui as mesmas funcionalidades do `Carrinho de Compra`, assim foi unificado ao carrinho e pode ser identificado por meio da variável lógica (booleana) `listaProdutos`, caso seja verdadeiro, então o carrinho de compras é uma lista de produtos, caso contrário é um `Carrinho de Compras` que por sua vez gera um `Pedido` com `FormasPagamento` e `FormaEnvio`. O mesmo foi realizado com o objeto de negócio `Orcamento`, sendo que este é identificado caso a variável booleana `orcamento` esteja configurada para verdadeiro na classe `Pedido`.

Além do diagrama de classes, o gerenciamento das variabilidades que são demarcadas com os estereótipos <<variant>> e <<optional>> é realizado para caracterizar a LPS.

Gerenciamento das variabilidades

A partir do modelo de rastreamento de variabilidades de cada característica, é possível identificar quais serão os objetos afetados a partir de uma característica opcional selecionada ou não. Deste modo, a consistência do sistema é mantida uma vez que ao selecionar uma característica é necessário utilizar dos objetos que fazem parte do caso de uso a qual aquela característica depende. Para isso, o desenvolvedor deve consultar o modelo de rastreamento de variabilidades para visualizar quais serão os elementos afetados ao selecionar uma característica.

Uma classe marcada com o estereótipo <<optional>> fará ou não parte do produto de software criado pela LPS, pois ela é parte de uma característica que também possuirá propriedades opcionais, sendo assim, ao não selecionar uma característica ligada a uma classe opcional, então esta classe não fará parte do produto criado pela LPS.

No estudo de caso ilustrativo do comércio eletrônico existem algumas classes que possuem o estereótipo <<variant>> isto significa que estes possuem pontos de variação, sendo assim eles precisam ser gerenciados para que a aplicação seja desenvolvida de acordo com as características configuradas e selecionadas a partir do modelo de características do domínio da LPS. Por exemplo, tomando como características a utilização da característica carrinho de compras os elementos afetados no sistema de acordo com a Tabela 8 serão: Controlar Entrega, Controlar Estoque, Gerenciar Produtos, Gerenciar Pagamento, Gerenciar Pedidos e Gerenciar Acesso. Porém, ao selecionar a característica lista de produtos, teremos os seguintes elementos afetados: Gerenciar Produtos, Gerenciar Pedidos e Gerenciar Acesso.

4.3 Configurações da Aplicação

Para instanciar uma aplicação a partir dos modelos de domínio definidos nas fases descritas acima, faz-se necessário a configuração dos elementos dos modelos associados aos pontos de

variação. Esta seção trata da configuração de uma aplicação realizando a modelagem navegacional e a modelagem de apresentação. A primeira trata do desenvolvimento dos contextos navegacionais assim como do acesso de cada usuário a esses contextos, enquanto a segunda trata da definição da maneira como os elementos do modelo serão apresentados ao usuário da aplicação.

4.3.1 Criação e configuração do Modelo Navegacional

A modelagem navegacional no OOWS consiste da definição da semântica de navegação do sistema. Um mapa da navegação mostra a acessibilidade e a visibilidade associadas a cada tipo de usuário.

Um contexto navegacional em sua essência consiste de páginas que dão acesso aos elementos do sistema e são representados pelo estereótipo `<<context>>`. Esses elementos são compostos de objetos de apresentação que oferecem as funcionalidades da aplicação Web para o usuário. A modelagem navegacional representa quais são os objetos de apresentação que um determinado usuário ou grupo de usuários podem acessar. Cada contexto pode incluir um ou mais subsistemas que são criados para permitir a interoperabilidade do domínio facilitando na identificação dos contextos afetados a partir de uma configuração da aplicação utilizando os modelos de rastreamento de variabilidades. Os subsistemas são representados pelo estereótipo `<<subsystem>>` e desempenham o papel de complementar as atividades do contexto com funcionalidades específicas de cada subsistema.

Os contextos navegacionais são desenvolvidos a partir dos objetos modelados nas etapas de Análise e Projeto, são utilizados principalmente os diagramas: dinâmico de estados e de seqüência. Do primeiro deve-se extrair os contextos navegacionais, e do segundo extrai-se os objetos de cada contexto navegacional. Cada contexto permitirá a ligação com os objetos do sistema. A Figura 34 mostra o modelo navegacional do sistema de comércio eletrônico, onde se tem o contexto navegacional dos usuários registrado (Vendedor ou Comprador) e anônimo.

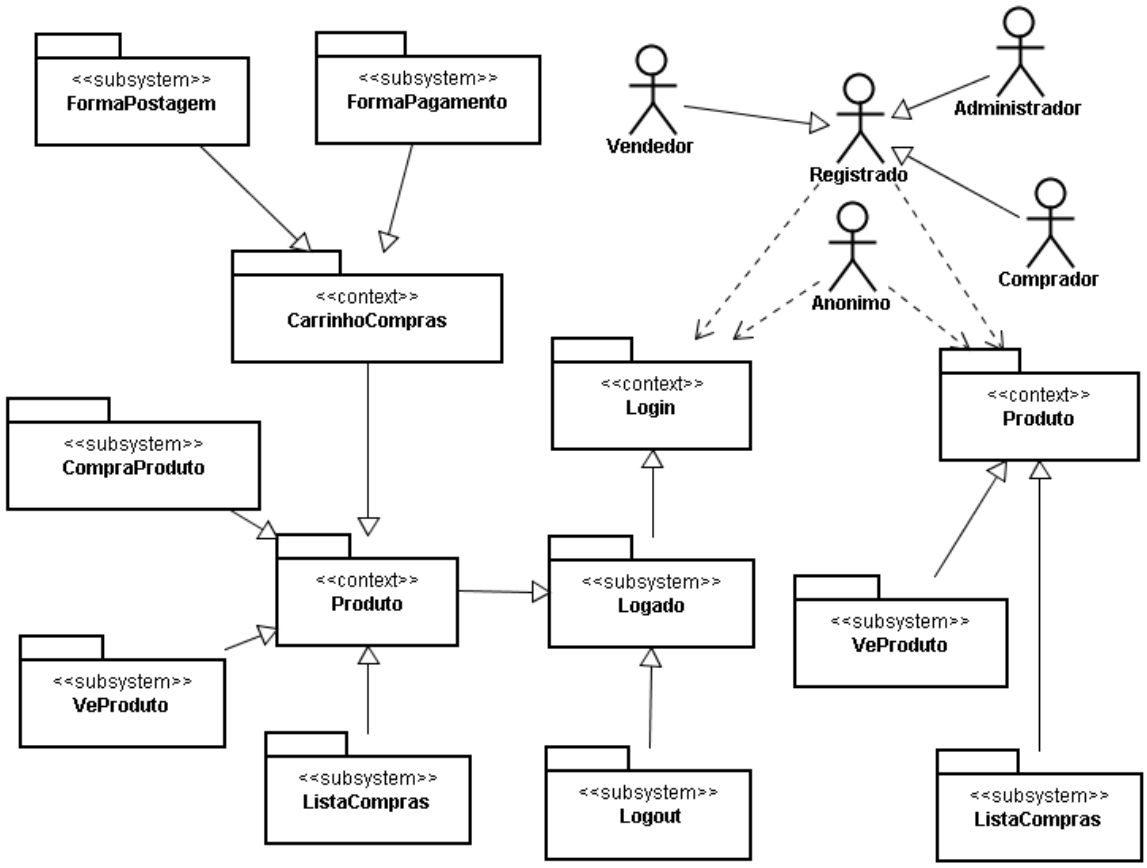


Figura 34: Modelo Navegacional do comércio eletrônico

Em SPL-OOWS, o modelo Navegacional é dependente do modelo de Usuário, pois na configuração das permissões são definidos quais os contextos que poderão ser acessados por cada tipo de usuário. No modelo navegacional representado pela Figura 34, temos que quando há o login (o usuário é registrado) o contexto navegacional de produto é diferente de quando o usuário está como Anônimo, no primeiro caso, o usuário poderá efetuar as compras e acompanhar sua compra utilizando o carrinho de compras - `CarrinhoCompras`, ou até mesmo criar uma lista de compras – `ListaCompras` enquanto que no segundo caso, só é possível visualizar as compras – `ListaCompras` e criar uma lista de compras temporária.

Mapeamento dos contextos navegacionais e o modelo de características

Para auxiliar na portabilidade da aplicação e na influência das características nos contextos navegacionais, é necessário fazer um cruzamento entre o modelo de características e o modelo Navegacional, pois assim, ao selecionar uma característica será possível identificar quais

contextos navegacionais deverão ser utilizados. A Tabela 9 ilustra o mapeamento das características com os contextos navegacionais para o sistema de comércio eletrônico.

Tabela 9 : Mapeamento das Características e os contextos navegacionais de um comércio eletrônico

Característica	Contexto Navegacional
Carrinho de Compra	CarrinhoCompras
Estoque Produtos	Produto, VeProduto
Distribuidor	Produto
Grupo do produto	Produto
Lista de Compra	ListaCompras
Pagamento	CompraProduto, FormaPagamento
Cartão	FormaPagamento
Dinheiro	FormaPagamento
Boleto	FormaPagamento
Cheque	FormaPagamento
Nota Fiscal	CompraProduto
Entrega de Produto	FormaPostagem
Transportadora	FormaPostagem
Correios	FormaPostagem

4.3.2 Criação e configuração do Modelo de Apresentação

O modelo de Apresentação do método SPL-OOWS utiliza os conceitos da Árvore de Ação Hierárquica concebida no método OOWS

A Figura 35 ilustra as características gerenciáveis para a geração da forma de Apresentação do domínio, isto inclui regras de negóocio que serão utilizadas para fornecer uma interface ao usuário final da aplicação. Este diagrama consiste das Unidades de Interação de níveis 2 e 3 que foram expostos na seção 2.4 do capítulo 2.

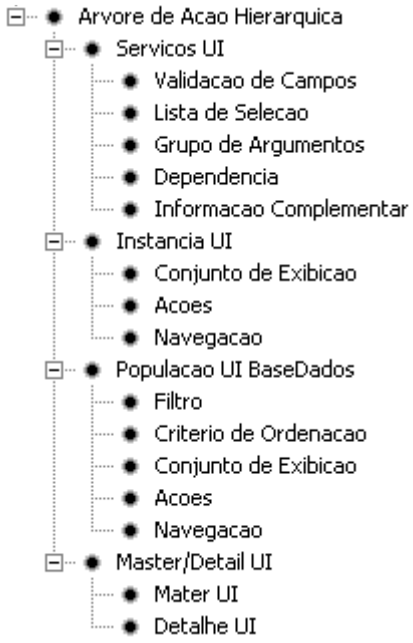


Figura 35: Estrutura Geral da Árvore de Ação Hierarquica

As Unidades de Interação (UI) serão gerenciadas para que a aplicação forneça a interface necessária para o acesso às funcionalidades da aplicação e a auxiliar ao usuário a gerenciar as informações que a aplicação deve gerar. Para a configuração dos elementos da apresentação, deve-se tomar como base o modelo navegacional desenvolvido e configurado na etapa anterior em conjunto com os elementos desenvolvidos na modelagem conceitual.

Para a criação da página de gerenciamento de carrinho de compras, identificou-se quais são as interações que este contexto possuirá, conforme mostra o diagrama interativo de estados apresentado na Figura 36.

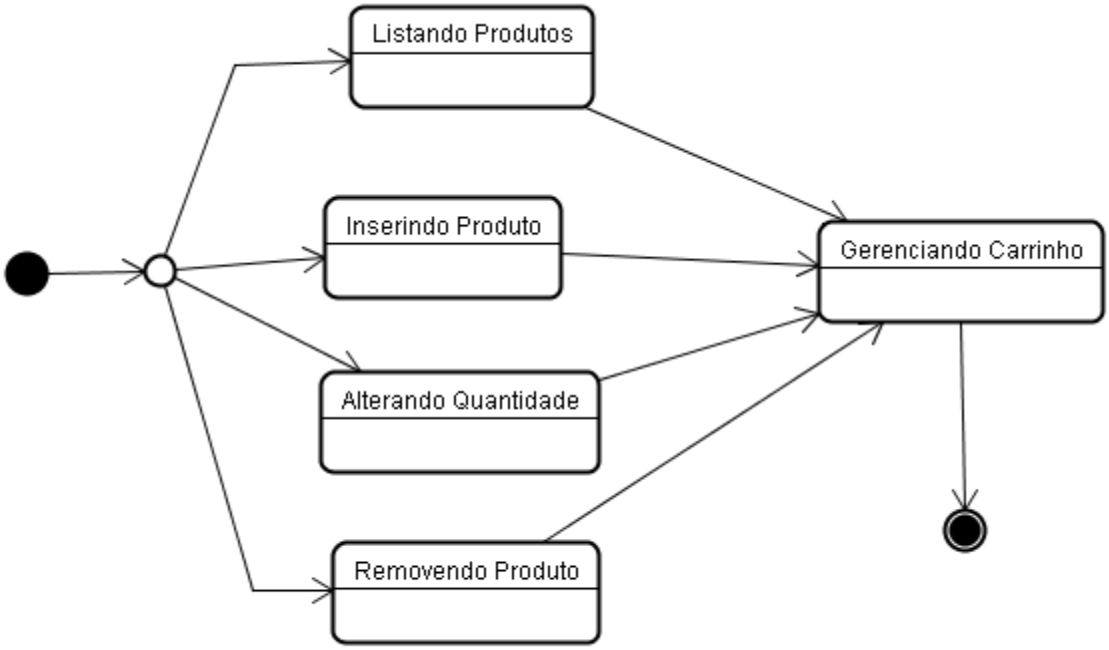


Figura 36 : Diagrama interativo de estados do gerenciamento do carrinho de compras do comércio eletrônico

Do diagrama da figura 36 temos 4 estados para o gerenciamento do carrinho: `Listando Produtos`, `Inserindo Produto`, `Alterando Quantidade` e `Removendo Produto`. Cada um destes estados é apresentado ao usuário de uma única vez, fornecendo a opção de escolha para cada um destes estados. A tabela 10 ilustra a árvore de ação hierárquica para o contexto navegacional de Produto.

Tabela 10: Árvore de Ação Hierárquica do contexto Produto

Nível 2	Nível 3	Descrição
Serviços UI	Validação de campos	Gerenciador de formulários
	Lista de seleção	Lista de produtos
Instância UI	Conjunto de Exibição	NomeProduto, DescricaoProduto.
	Ações	Cadastrar Produto, Excluir Produto, Alterar Produto, Incluir no carrinho de compras, Incluir na Lista de compras
	Navegação	Página inicial, Produtos, Carrinho de compras, lista de compras
População UI	Filtro	Produtos por fornecedor, Produtos por tipo
	Conjunto de exibição	QueryProdutos
	Ações	Cadastrar Produto, Excluir Produto, Alterar Produto, Incluir no carrinho de compras, Incluir na Lista de compras
	Navegação	Página inicial, Produtos, Carrinho de compras, lista de compras
Mestre-específico	Mestre	NomeProduto, DescricaoProduto
	Específico	PesoProduto

4.4 Implementação – Criação de produtos de software

Para criar produtos de software, primeiro, devem ser selecionadas as características que serão utilizadas, em seguida, uma análise dos impactos de escolha destas características deve ser realizado para então iniciar o processo de desenvolvimento dos produtos. A partir das classes especificadas no Projeto, dá-se inicio a implementação dos métodos previstos de acordo com os fluxos a partir dos diagramas de sequência, assim sendo a aplicação deverá ser criada de acordo com a arquitetura proposta. A Figura 37 ilustra o exemplo de estrutura de implementação das camadas do método SPL-OOWS.

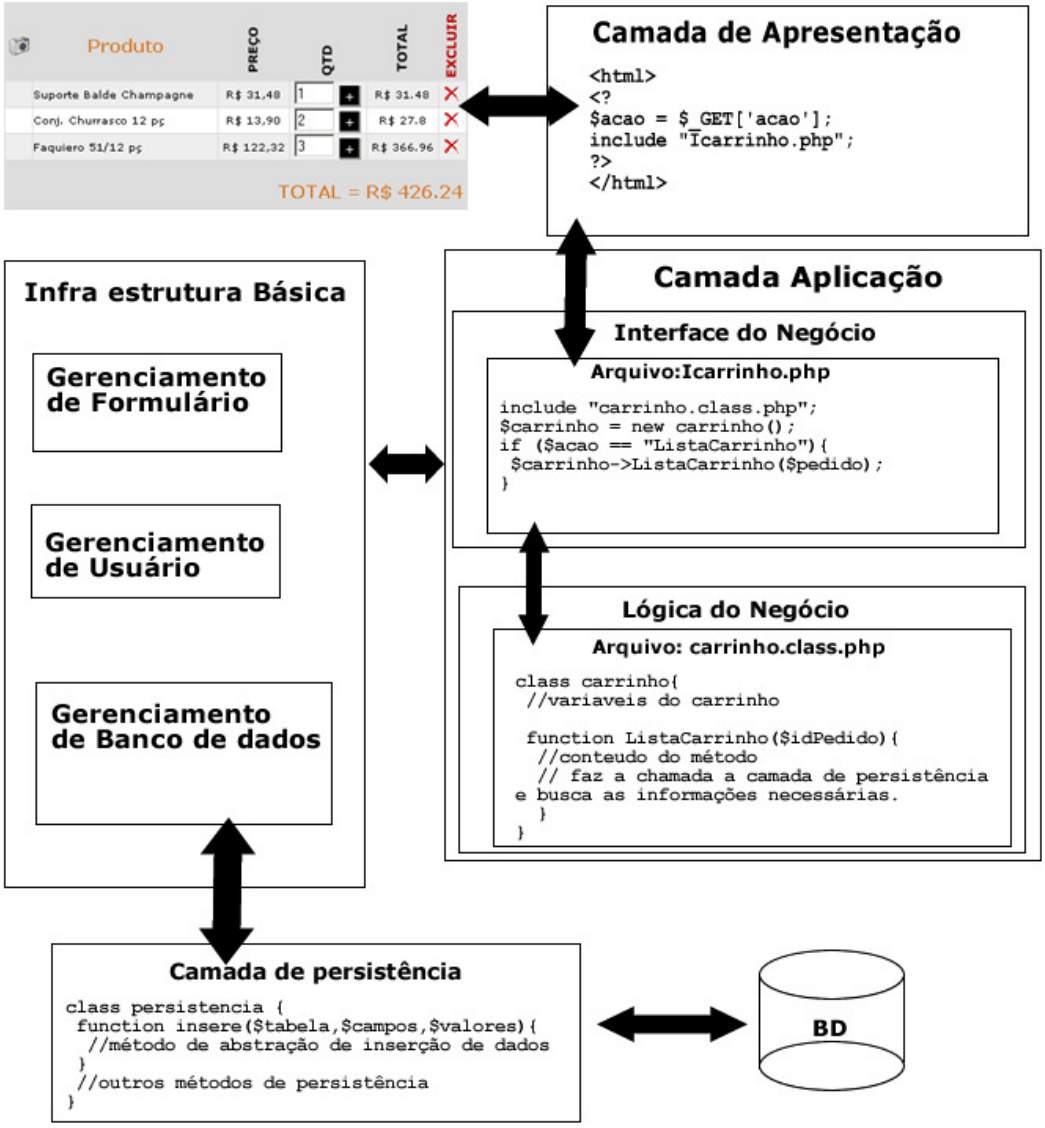


Figura 37: Visão geral da estrutura de implementação do SPL-OOWS

Na Figura 37, tem-se que a camada de `apresentação` conterá os códigos HTML e os *scripts* a serem executados no modo cliente (*javascripts*). A camada de aplicação possui duas subcamadas: `Interface do negócio` e `lógica do negócio`. Além disso, existe a camada da `Infra-Estrutura básica` e a camada de `persistência`. A seguir é descrito como foi implementada cada uma das camadas.

4.4.1 Infra Estrutura Básica

Na fase de implementação, o gerenciamento de usuário é configurado para atender as necessidades do domínio, sendo assim, para a realização do controle de acesso, é necessário cadastrar os usuários, os contextos navegacionais, os atributos da apresentação e as páginas. Com esses elementos cadastrados, o controle é realizado por uma interface do gerenciamento de usuários que tem como característica permitir ou negar o acesso ao objeto requerido. A Figura 38 ilustra a página de configuração permitindo as ações de "Alteração", "Inserção", "Remoção" e "Visualização" para o grupo de usuário "Geral" na página de cadastro de "cliente".

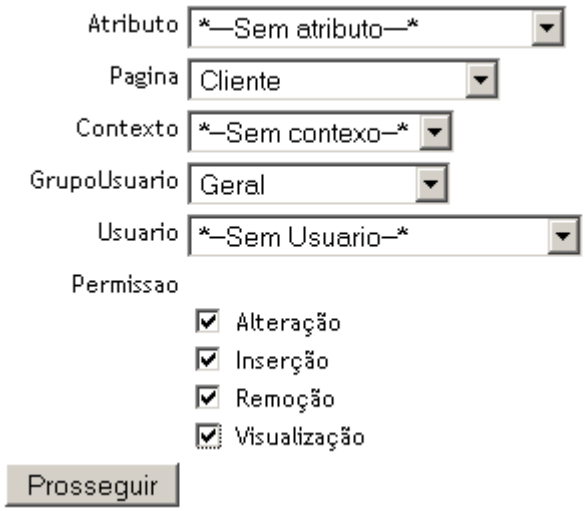


Figura 38: Tela de configuração da Infra-estrutura básica do gerenciamento de usuários

Para a configuração da Infra-estrutura básica do gerenciador de usuários, o programador do domínio deve especificar qual o formato dos dados que serão enviados do formulário para o banco de dados. Esta configuração é realizada pelo elemento gerenciador de formulários a partir da especificação da padronização dos dados. Um exemplo de código criado para o gerenciador de formulários é o campo CPF que está presente na página de cadastro de clientes, assim para padronizar este campo, antes de realizar a operação de cadastro ou alteração na tabela que armazena os dados do cliente, deverá ser especificado o campo e o padrão a ser adotado por este:

```
String cpf
```

```
cpf = form.configCampo($_POST['cpf'],'999.999.999-99');
```

Dessa forma, o campo "cpf" receberá o CPF no padrão pré-estabelecido no segundo argumento do método `configcampo`.

Para o desenvolvimento da camada de persistência, o programador deverá ter conhecimento do sistema gerenciador de banco de dados, pois é necessário conhecer a sintaxe de suas operações. Desta forma, é possível o desenvolvimento dos métodos definidos pela interface do gerenciador de banco de dados.

4.4.2 Camada de Apresentação

A camada de apresentação contém os dados que serão mostrados para o usuário, portanto, nesta camada ficam os arquivos de configuração de estilos das páginas (ex. CSS), arquivos de controle do cliente (Javascripts) e os arquivos de marcação (XML). Além destes, para o desenvolvimento da camada de apresentação faz-se necessário o mapeamento das configurações a partir do modelo de apresentação gerado. Os elementos mapeados desta etapa invocam elementos da lógica do negócio e eles são instanciados conforme a necessidade de apresentação de cada item.

Por exemplo, na página de gerenciamento do carrinho de compras, para a listagem dos produtos que estão armazenados no banco de dados é necessário que haja o acesso ao banco e que o método `listarProduto` da classe do `carrinho de compras` seja executado. Para que isto aconteça na página que irá ser apresentada ao usuário final deverá conter a chamada para o método de listagem de produtos para que os produtos sejam listados e este método poderá ser invocado por meio da chamada do elemento que estará na Interface do negócio na camada de aplicação.

4.4.3 Camada de Aplicação

A camada de aplicação, assim como no método OOWS, continua sendo subdividida em duas subcamadas: a interface do negócio e a lógica do negócio, sendo que aquela trata as informações provindas da camada de apresentação e esta trata da lógica do negócio.

A subcamada da interface do negócio é a responsável por fazer o intermédio entre a camada de apresentação e a subcamada da lógica do negócio. A interface é responsável pelas

instanciações das classes para que os métodos e atributos destas sejam acessados pela aplicação por intermédio da camada de apresentação.

Já a subcamada de lógica do negócio contém todas as operações lógicas da aplicação e ela é instanciada pela subcamada da interface do negócio. É a camada da lógica do negócio que trata das informações no banco de dados.

4.4.4 Camada de Persistência

A camada de persistência faz a interface de comunicação entre os dados enviados pelo sistema e o sistema gerenciador de banco de dados. Ela é dependente da tecnologia de banco de dados adotada e é a responsável por persistir os dados da aplicação.

A separação desta camada faz-se necessária para que haja uma melhor portabilidade da aplicação no que diz respeito a utilização de tecnologias de banco de dados. Sendo assim, a camada de persistência (representada no diagrama de componentes - Figura 31) é um componente que contém as interfaces de comunicação para realizar as operações de Banco de dados específicas para um sistema gerenciador de banco de dados.

4.5 Considerações finais

O Capítulo 4 descreveu as etapas de desenvolvimento do método proposto SPL-OOWS e ilustrou um estudo de caso de um comércio eletrônico. É possível ver mais detalhes sobre a elaboração da modelagem conceitual e das configurações deste domínio no apêndice A – Diagramas do estudo de caso ilustrativo do comércio eletrônico.

O Capítulo 5 contempla as etapas de desenvolvimento do SPL-OOWS com o desenvolvimento de um estudo de caso a fim de ilustrar as formas de desenvolvimento e interações que podem ocorrer ao utilizar o método proposto.

Capítulo

5

Um exemplo de aplicação do método SPL-OOWS

O exemplo apresentado neste capítulo visa explorar a aplicação do SPL-OOWS em outro contexto. É utilizado como domínio de aplicação um gerenciador de referências bibliográficas.

Para o desenvolvimento deste exemplo de aplicação foram utilizadas as seguintes ferramentas: JUDE Community (JUDE, 2008) - para a modelagem UML; e, Feature-plugin (ANTKIEWICZ e CZARNECKI, 2004) - para modelagem de características. Para a fase de implementação foi utilizado o servidor Apache configurado com a linguagem de programação PHP, além do gerenciador de banco de dados MySQL.

Este capítulo apresenta um resumo do domínio de aplicação e em seguida a modelagem da LPS seguindo as mesmas fases e etapas do método SPL-OOWS descritas no capítulo 4.

5.1. Domínio de aplicação – Gerenciador de Referências

O gerenciador de referências têm como objetivo auxiliar no intercâmbio de publicações entre pesquisadores e grupos de estudo de uma mesma área de conhecimento. O gerenciador permite aos pesquisadores incluir a referência de uma publicação e relacioná-la com as áreas de conhecimento, as áreas de concentração e as palavras-chave. As áreas de concentração envolvem um conceito mais amplo que a área de conhecimento, por exemplo: área de conhecimento: Ciência da Computação, áreas de concentração: Engenharia de Software, Arquitetura de Computadores e Inteligência Artificial. Dessa forma, um mesmo

grupo de pesquisa pode ter acesso às referências em um formato consistente, de modo que novos membros do grupo podem conhecer a literatura base para iniciar suas pesquisas, bem como incluir em seus documentos as referências em um formato padrão. No sistema em questão existem 4 (quatro) tipos de usuário: o administrador do sistema; o moderador e o estudante de um grupo de estudo e o usuário anônimo que terá acesso somente à lista de publicações.

5.2 Modelagem conceitual

5.2.1 Requisitos

A Figura 39 mostra o modelo de casos de uso desenvolvido na etapa de requisitos. Neste modelo, tem-se os casos de uso: `Gerenciar Publicações` que trata do cadastro das referências de acordo com as áreas de concentração e `Gerenciar Áreas de Concentração` que consiste em gerenciar as diversas áreas às quais as publicações estão relacionadas. O caso de uso `Gerenciar publicações` é do tipo *kernel,* pois ele faz parte do núcleo de qualquer aplicação que será gerada a partir da LPS. Em contrapartida, o caso de uso `Gerenciar Áreas de Concentração` poderá não fazer parte de um produto de software da LPS por se tratar de um caso de uso *optional*.

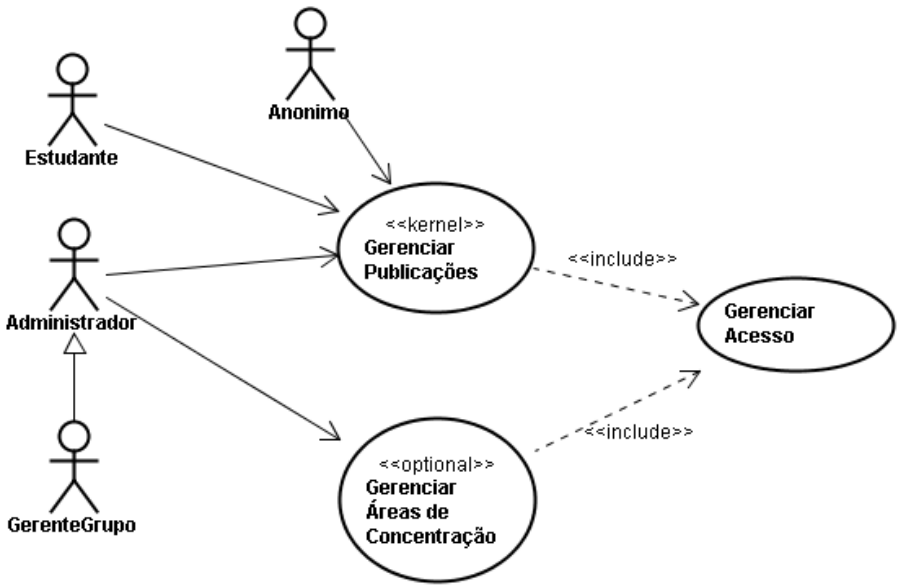


Figura 39: Modelo de Caso de uso do Gerenciador de Referências

Após a criação dos casos de uso, é construído o modelo de características do domínio, conforme apresentado na Figura 40. Neste modelo, as características: `Autor`, `Editora` e `Palavra-chave` são obrigatórias e as características `Area_concentração e Area_conhecimento` são opcionais.

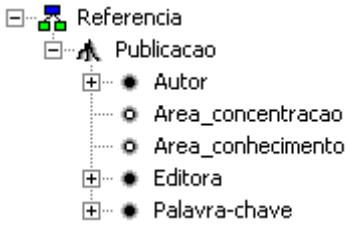


Figura 40: Modelo de características da aplicação do Gerenciador de Referências

5.2.2 Análise

A seguir serão apresentados os modelos obtidos a partir da etapa de análise.

Modelo Dinâmico de Estados

O modelo Dinâmico de Estados apresentado na Figura 41 ilustra os estados que deverão ser realizados durante o gerenciamento das áreas de concentração que engloba os serviços de inserção, remoção e alteração da área de concentração.

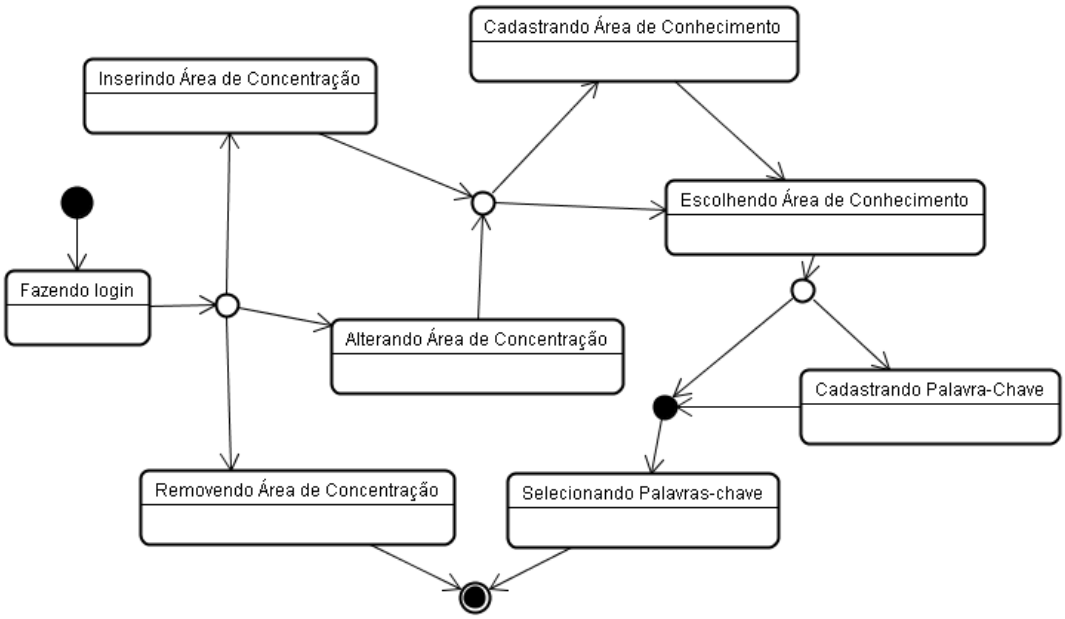


Figura 41: Modelo Dinâmico de Estados do caso de uso Gerenciar Área de Concentração

A seleção da opção de gerenciar áreas de conhecimento dispara a execução do diagrama de estados da Figura 41. Neste caso, ao selecionar a opção de Inserir área de concentração, o usuário passará a registrar a área de concentração, em que deverá selecionar quais são as áreas de conhecimento e as palavras-chave que fazem parte da área de concentração em questão.

A Figura 42 mostra o modelo dinâmico de estados do caso de uso `Gerenciar Publicação`. Neste modelo, a partir do *login* do usuário, dependendo de suas atribuições (ex. gerente de grupo, estudante ou administrador), ele poderá realizar as funções de inserção, remoção e alteração de publicações. Isto implica em selecionar as palavras-chave, as áreas de concentração e as áreas de conhecimento da publicação que está sendo gerenciada.

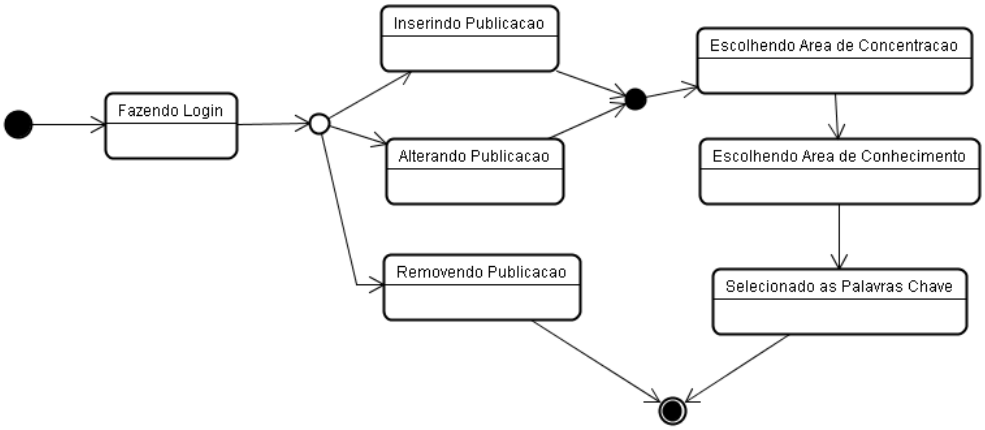


Figura 42: Modelo dinâmico do caso de uso Gerenciar Publicação

Modelo Estático

O modelo estático apresentado na Figura 43 mostra as ações vinculadas às classes da aplicação, assim como os pontos de variação (*variant*) e os objetos que são do núcleo da aplicação (*kernel*) e opcionais (*optional*). Por exemplo, o gerente da LPS pode criar uma aplicação de gerenciamento de publicações que não utilize a classe opcional `AreaConhecimento`. Isto implica na criação de um produto em que o vínculo da Publicação ocorrerá somente com a classe opcional `AreaConcentracao` e as demais classes do núcleo do domínio.

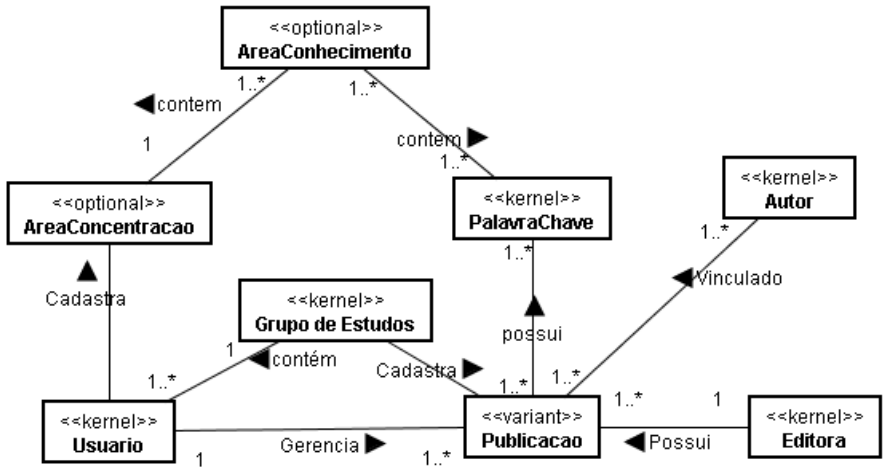


Figura 43: Modelo Estático – Referência

Modelo dinâmico interativo

O modelo dinâmico interativo é apresentado na Figura 44. Ao cadastrar uma área de concentração (1), o usuário deve selecionar as áreas de conhecimento (2). Caso não exista, o usuário deverá cadastrar a área de conhecimento (3) e selecioná-la (4). Feito isso, o usuário deverá selecionar as palavras-chave desta área (5). Não existindo a palavra desejada, o usuário deverá cadastrá-la (6) e então escolhê-la (7).

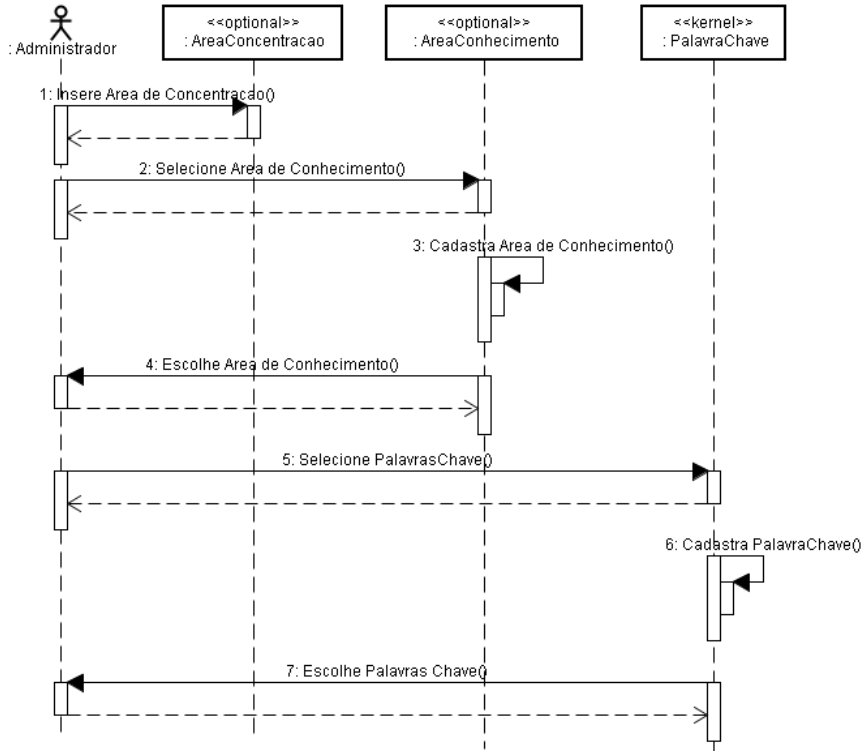


Figura 44: Modelo dinâmico interativo do gerenciamento de área de concentração

A Figura 45 ilustra o modelo dinâmico interativo do caso de uso `Gerenciar Publicações`. Ao inserir uma nova publicação (1) o usuário deverá selecionar os seguintes itens: Editora (2), Area de Concetração (3), Área de Conhecimento(4), Palavras-Chave (5) e o autor. Após essas etapas, o usuário seleciona os autores e finaliza o processo de cadastro da publicação.

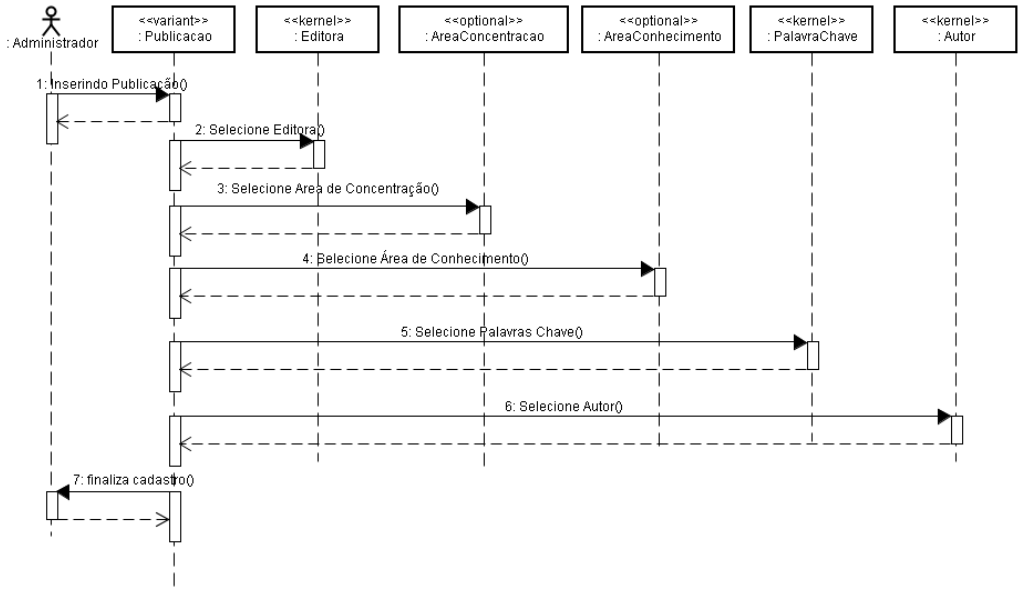


Figura 45: Modelo Dinâmico interativo do gerenciamento de publicações

Modelo de Rastreamento de Variabilidades

O modelo de rastreamento de variabilidades foi elaborado conforme descrição na Seção 4.2.2, em que são marcadas as relações entre características e casos de uso. A Tabela 11 apresenta o modelo de rastreamento das variabilidades para a aplicação do gerenciamento de referências.

Tabela 11: Modelo de Rastreamento das variabilidades para a aplicação do gerenciamento de Referências

Features/Casos de Uso	Gerenciar Publicações	Gerenciar Acesso	Gerenciar Áreas de Concentracao
Editora	•	•	
Palavra-Chave	•	•	•
Autor	•	•	
Área de Concentração	•	•	•
Área de Conhecimento	•	•	•
Publicação	•	•	

5.2.3 Projeto

Modelo Arquitetural

A arquitetura do sistema segue o padrão especificado no capítulo 4, logo, o diagrama de pacotes para a aplicação do gerenciamento de publicações não difere do diagrama de pacotes para o comércio eletrônico (ver Figura 31).

Os diagramas de seqüência apresentados nas Figuras 46 e 47 representam as mesmas atividades expostas nas Figuras 44 e 45 respectivamente. Neste diagrama, são adicionados os métodos das classes conforme descritos no modelo dinâmico interativo.

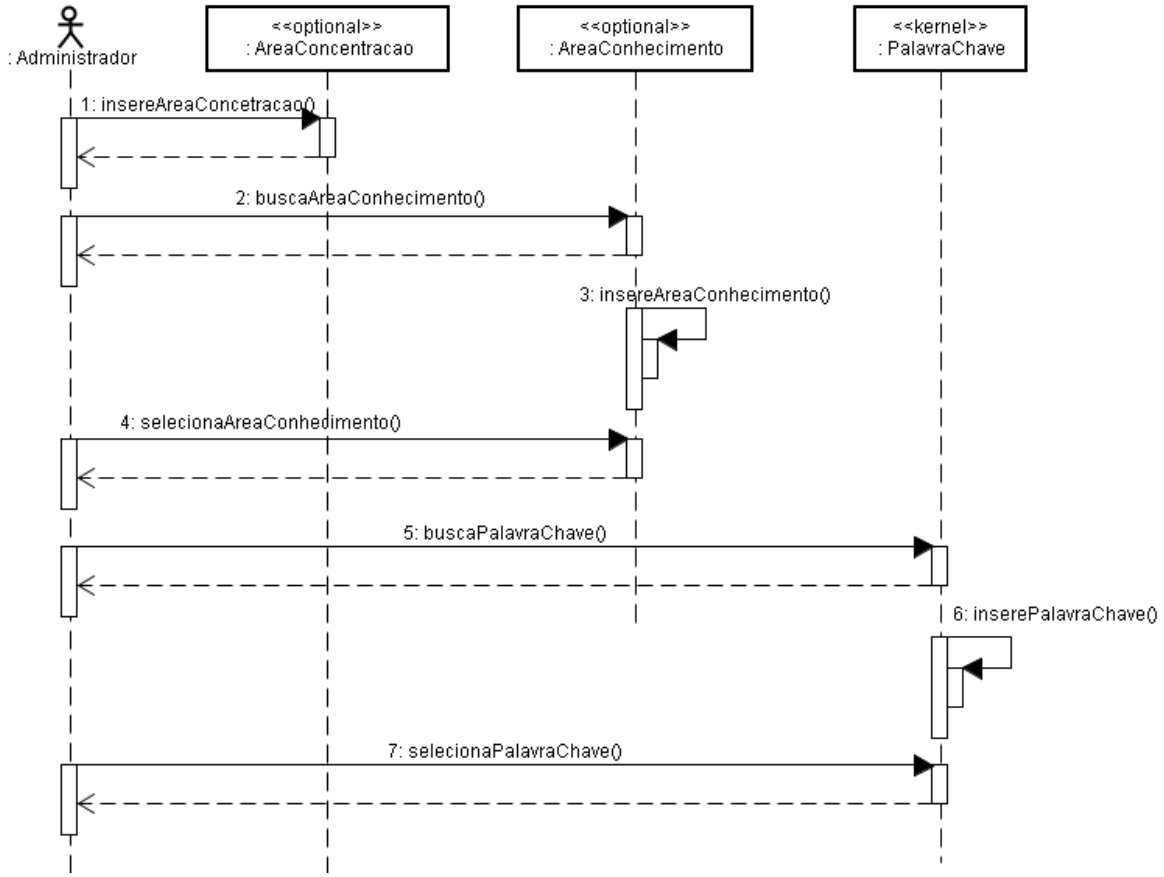


Figura 46: Diagrama de sequencia de gerenciar area de concentração

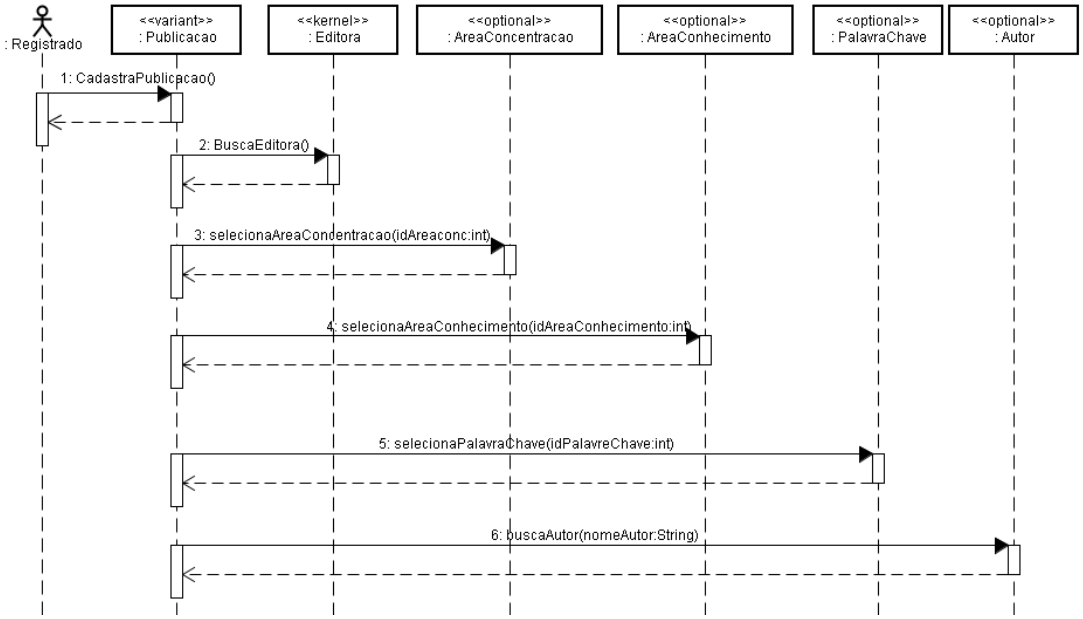


Figura 47: Diagrama de Sequência de Gerenciar publicação

O diagrama de classes apresentado na Figura 48 representa a estrutura lógica da aplicação, bem como seus atributos e métodos os quais serão utilizados na implementação.

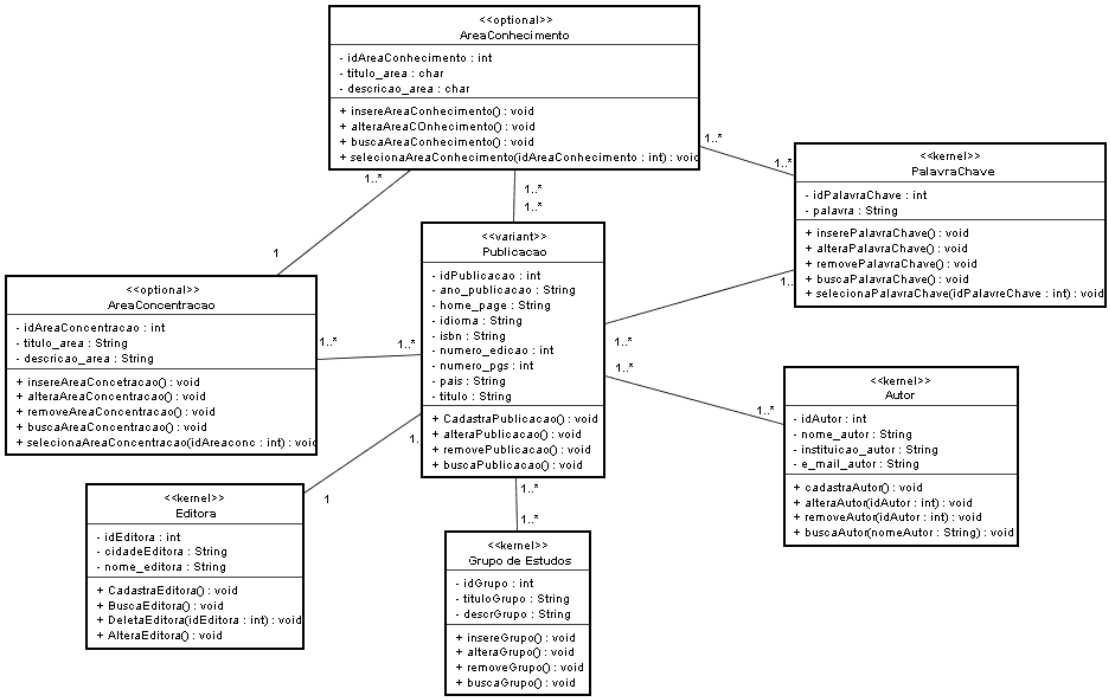


Figura 48: Diagrama de classe do sistema de gerenciamento de publicações

5.3 Configuração da Aplicação

O primeiro modelo a ser configurado é o de navegação para os 4 tipos de usuário do sistema: o administrador do sistema, o moderador, o estudante de um grupo de estudos e o usuário anônimo que terá acesso somente à lista de publicações. Na Figura 49 é apresentado o modelo navegacional dos usuários `Administrador` e do `GerenteGrupo`. Este último não poderá alterar as permissões dos usuários, ou atribuir um usuário estudante como gerente de grupo, pois isto é papel do administrador.

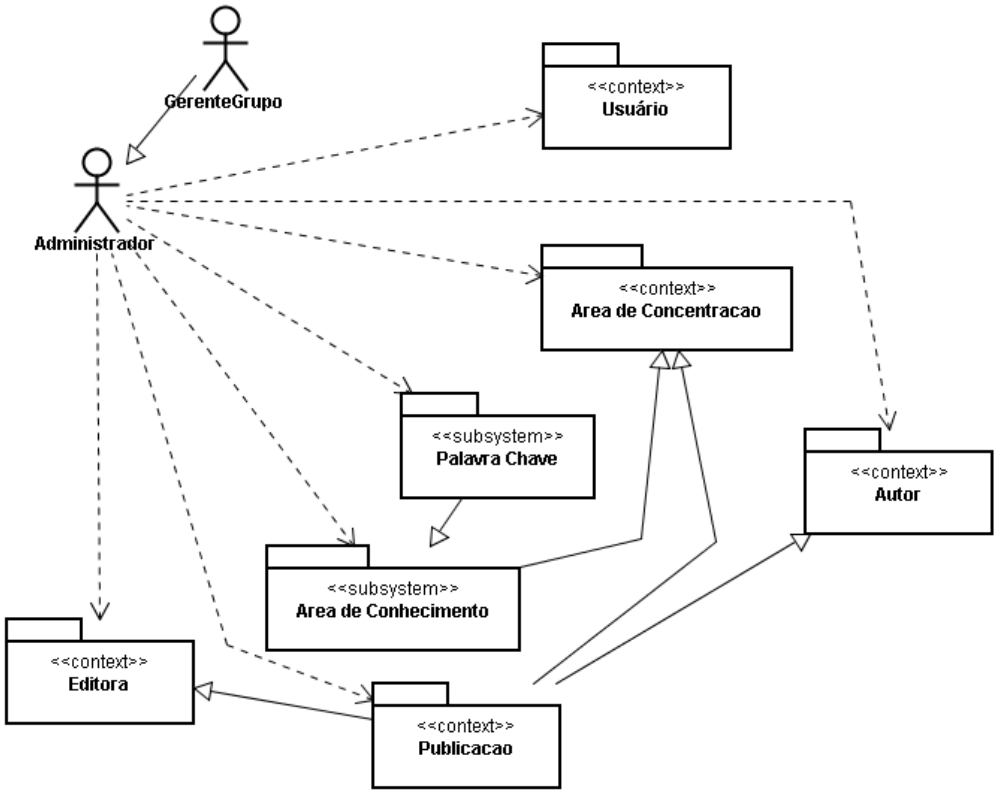


Figura 49: Modelo Navegacional do Usuário Administrador e GerenteGrupo

A Figura 50 mostra o modelo navegacional do usuário `Estudante` que é responsável por inserir publicações para as pesquisas do grupo de estudos. A inserção de uma publicação pode implicar na inclusão de: Área de conhecimento, Área de concentração, Palavras-chave, Autor e Editora para cada publicação, dependendo das configurações da LPS.

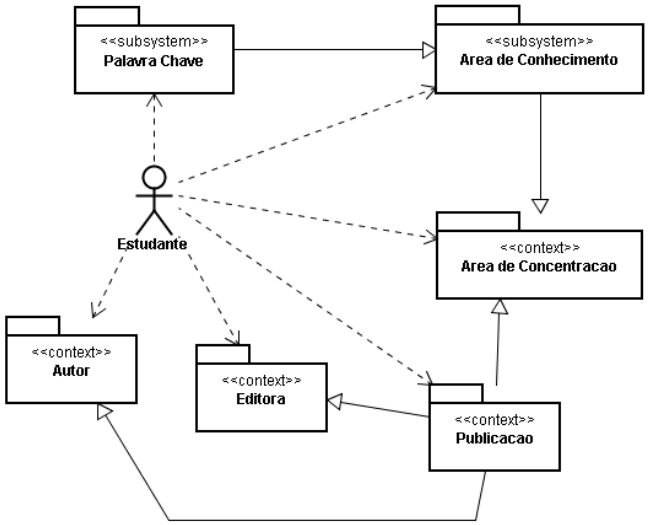


Figura 50: Modelo Navegacional do Usuário EstudanteGrupo

A Figura 51 apresenta o modelo navegacional do usuário Anônimo, onde ele poderá consultar as publicações disponíveis no sistema, sem a permissão para inclusão de novos materiais.

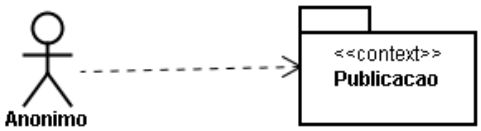


Figura 51: Modelo Navegacional do usuário Anônimo

Com a definição dos modelos navegacionais, é necessário realizar um mapeamento para identificar quais contextos serão afetados a partir de uma configuração no modelo de características.

Tabela 12: Mapeamento dos contextos navegacionais e as características da aplicação de gerenciamento de referências

Características	Contextos Navegacionais
Editora	Editora
Palavra-Chave	Palavra-Chave
Autor	Autor
Area de Concentração	Area de Concentracao
Area de Conhecimento	Area de Conhecimento
Publicação	Publicacao

A partir dos modelos definidos, na configuração do modelo de apresentação, cada contexto navegacional terá sua árvore de ação hierárquica que definirá com precisão quais serão os elementos de apresentação da aplicação. A seguir, para cada contexto, será ilustrada a árvore de ação hierárquica:

Tabela 13: Árvore de ação hierárquica para o contexto navegacional Editora

Nível 2	Nível 3	Descrição
Serviços UI	Validação de campos	Gerenciador de Formulários.
	Lista de seleção	Lista de Editoras, Lista de Editoras por busca.
Instância UI	Conjunto de Exibição	nome_editora, cidade_editora.
	Ações	Cadastrar Editora, Selecionar Editora, Remover Editora, Alterar Editora, Listar Editora.
	Navegação	Publicação, Area de Conhecimento, Area de Concentração, Palavra-chave, Autor.
População UI	Filtro	Editora por cidade, Editora por String de Busca.
	Conjunto de exibição	Query Lista Editora.
	Ações	Cadastrar Editora, Selecionar Editora, Remover Editora, Alterar Editora, Listar Editora.

	Navegação	Publicação, Area de Conhecimento, Area de Concentração, Palavra-chave, Autor, Editora.
Mestre-específico UI	Mestre	nomeEditora, cidadeEditora
	Específico	descriçãoEditora, contatoEditora

Tabela 14: Árvore de Ação Hierárquica para o contexto navegacional Palavra-chave

Nível 2	**Nível 3**	**Descrição**
Serviços UI	Validação de campos	Gerenciador de Formulários.
	Lista de seleção	Busca palavra-chave;
Instância UI	Conjunto de Exibição	Palavra-chave.
	Ações	Cadastrar palavra-chave, Alterar palavra-chave, Excluir palavra-chave, Listar palavras-chave.
	Navegação	Publicação, Area de Conhecimento, Area de Concentração, Palavra-chave, Autor, Editora.
População UI	Filtro	Palavra-chave por área-conhecimento, palavra-chave por área-concentração, palavra-chave por publicação, palavra-chave por autor
	Conjunto de exibição	QueryListaPalavra-chave
	Ações	Cadastrar palavra-chave, Alterar palavra-chave, Excluir palavra-chave, Listar palavras-chave.
	Navegação	Publicação, Area de Conhecimento, Area de Concentração, Palavra-chave, Autor, Editora.
Mestre-específico UI	Mestre	palavra-chave
	Específico	palavrasRelacionadas

Tabela 15: Árvore de Ação Hierárquica para o contexto navegacional Autor

Nível 2	**Nível 3**	**Descrição**
Serviços UI	Validação de campos	Gerenciador de Formulários.

	Lista de seleção	Busca Autor, Lista de autores
Instância UI	Conjunto de Exibição	nomeAutor
	Ações	Cadastrar Autor, Alterar Autor, Excluir Autor, Listar Autor.
	Navegação	Publicação, Area de Conhecimento, Area de Concentração, Palavra-chave, Autor, Editora.
População UI	Filtro	Autor por publicação, autor por área de concentração, autor por área de conhecimento.
	Conjunto de exibição	QueryAutor
	Ações	Cadastrar Autor, Alterar Autor, Excluir Autor, Listar Autor.
	Navegação	Publicação, Area de Conhecimento, Area de Concentração, Palavra-chave, Autor, Editora.
Mestre-específico UI	Mestre	nomeAutor
	Específico	enderecoAutor, instituicaoAutor

Tabela 16: Árvore de Ação Hierárquica para o contexto navegacional Area de Concentração

Nível 2	**Nível 3**	**Descrição**
Serviços UI	Validação de campos	Gerenciador de Formulários.
	Lista de seleção	Lista Área de concentração, Busca área de concentração
Instância UI	Conjunto de Exibição	Área de concentração
	Ações	Cadastrar Area de concetração, Alterar Area de concetração , Excluir Area de concetração, Listar Area de concetração, Cadastrar área de Conhecimento, Selecionar área de conhecimento.
	Navegação	Publicação, Area de Conhecimento, Area de Concentração, Palavra-chave, Autor, Editora.
População UI	Filtro	Area de concentração por palavra-chave, área de concentração por autor, área de concentração por área de conhecimento, área de concentração por publicação.

	Conjunto de exibição	QueryAreaConcentração
	Ações	Cadastrar Area de concetração, Alterar Area de concetração , Excluir Area de concetração, Listar Area de concetração.
	Navegação	Publicação, Area de Conhecimento, Area de Concentração, Palavra-chave, Autor, Editora.
Mestre-específico UI	Mestre	
	Específico	

Tabela 17: Árvore de Ação Hierárquica para o contexto navegacional Área de Conhecimento

Nível 2	**Nível 3**	**Descrição**
Serviços UI	Validação de campos	Gerenciador de Formulários.
	Lista de seleção	Lista Área de conhecimento, Busca área de conhecimento
Instância UI	Conjunto de Exibição	Área de conhecimento
	Ações	Cadastrar Area de conhecimento, Alterar Area de conhecimento , Excluir Area de conhecimento, Listar Área de conhecimento, Cadastrar palavra-chave, Selecionar palavra-chave.
	Navegação	Publicação, Area de Conhecimento, Area de Concentração, Palavra-chave, Autor, Editora.
População UI	Filtro	Area de conhecimento por palavra-chave, área de conhecimento por autor, área de conhecimento por publicação, área de conhecimento por área de concentração.
	Conjunto de exibição	QueryAreaConhecimento
	Ações	Cadastrar Area de conhecimento, Alterar Area de conhecimento , Excluir Area de conhecimento, Listar Area de conhecimento.
	Navegação	Publicação, Area de Conhecimento, Area de Concentração, Palavra-chave, Autor, Editora.
Mestre-específico UI	Mestre	

	Específico	

Tabela 18: Árvore de Ação Hierárquica para o contexto navegacional Publicação

Nível 2	**Nível 3**	**Descrição**
Serviços UI	Validação de campos	Gerenciador de Formulários.
	Lista de seleção	Lista publicações, Busca publicações
Instância UI	Conjunto de Exibição	nomeAutoresPublicacao, nomePublicacao, nomeEditoraPublicacao, areaConcentracao, areaConhecimento, palavras-chave.
	Ações	Cadastrar Publicação, Alterar Publicação , Excluir Publicação, Listar Publicação, Cadastrar área de Concentração, Selecionar área de concentração, Cadastrar área de Conhecimento, Selecionar área de conhecimento, Selecionar Palavra-chave, cadastrar palavra-chave.
	Navegação	Publicação, Area de Conhecimento, Area de Concentração, Palavra-chave, Autor, Editora.
População UI	Filtro	Publicação por palavra-chave, Publicação por autor, Publicação por área de concentração, Publicação por área de conhecimento, Publicação por editora.
	Conjunto de exibição	QueryPublicação
	Ações	Cadastrar Publicação, Alterar Publicação , Excluir Publicação, Listar Publicação.
	Navegação	Publicação, Area de Conhecimento, Area de Concentração, Palavra-chave, Autor, Editora.
Mestre-específico UI	Mestre	
	Específico	

5.4 Implementação - Criação de produtos de software

De acordo com as especificações do método SPL-OOWS foi criado a estrutura de implementação como mostra a Figura 52:

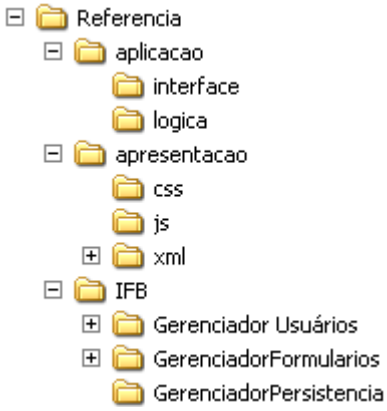


Figura 52 : Estrutura de implementação da aplicação de Referencia

A camada `aplicacao` possui duas camadas, a `interface` e a `lógica` do negócio. Na interface existem os controles que fazem a ligação entre a camada de `apresentacao` e a `lógica`. A Infra-estrutura básica (`IFB`) contém os gerenciadores previamente definidos pelo SPL-OOWS. A camada de apresentação contém os elementos de gerenciamento de apresentação, como os códigos em CSS (`css`), Javascript (`js`) e as definições xml (`XML`).

Tomando como base o desenvolvimento de um produto utilizando todas as características da aplicação, A Figura 53 mostra a configuração do modelo de características.

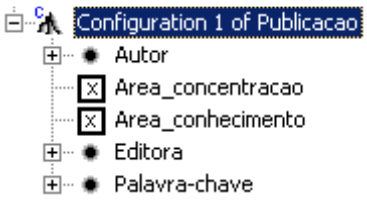


Figura 53: Configuração do modelo de características para a configuração da aplicação de referência

Com essas características selecionadas, a aplicação terá o diagrama de classes representado pela Figura 48, deste diagrama, extrai-se o modelo Entidade-Relacionamento

que servirá como base para o desenvolvimento da estrutura de banco de dados da aplicação de referência. Com o banco de dados criado e as classes já definidas as atividades da lógica do negocio deverão ser criadas, como por exemplo, o método `buscaPublicacao` da classe `Publicacao` que faz a busca no banco de dados a partir de uma palavra passada como parâmetro.

Após o desenvolvimento da lógica da aplicação, os contextos navegacionais foram desenvolvidos em conjunto com os elementos de apresentação de acordo com as especificações de árvore de ação hierárquica para cada contexto. A Visão geral do Sistema é mostrada na Figura 54.

Figura 54: Visão Geral do Sistema com os Contextos navegacionais

A Figura 55 mostra os elementos que fazem parte do contexto navegacional `Autor`, o qual tem acesso às páginas referentes a ele.

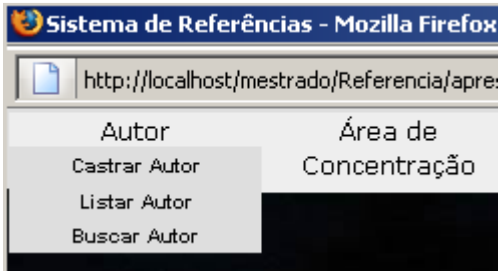


Figura 55: Elementos do contexto navegacional - Autor.

A Figura 56 mostra a tela de cadastro de autor, em que é possível a inserção de um novo `Autor` no sistema. Para Excluir ou alterar o cadastro de um autor, o usuário deverá acessar a página Listar Autor, e na listagem aparecerão as opções de Alterar ou excluir. Como mostra a Figura 57.

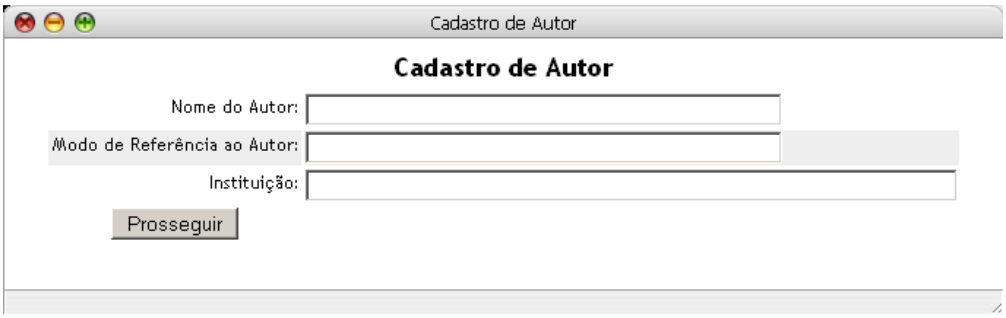


Figura 56: Formulário de cadastro de autor

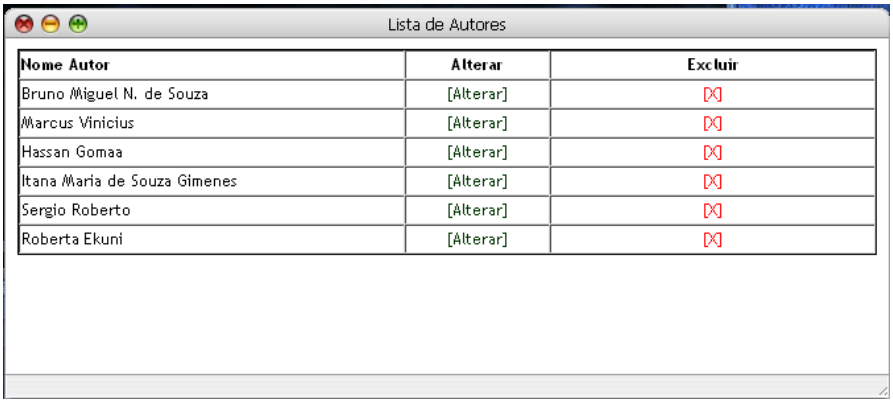


Figura 57: Página de lista de Autores

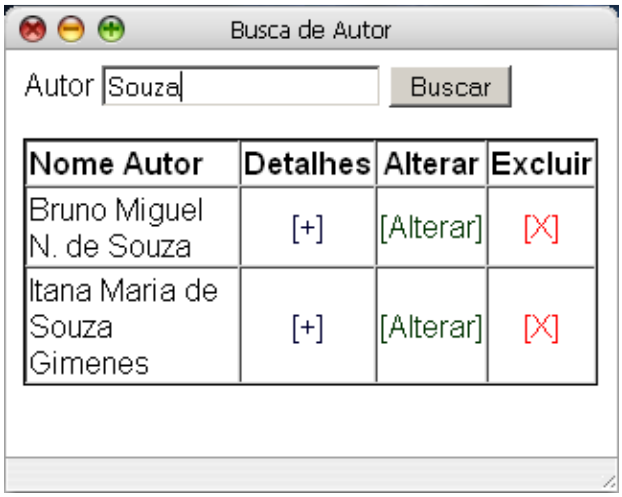


Figura 58: Página de Busca de Autores

Na Figura 59 é exposto as páginas que fazem parte do contexto navegacional de Área de Concentração.

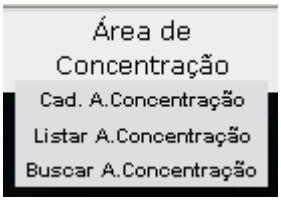


Figura 59: Contexto navegacional Área de Concentração

A partir das configurações criadas para o modelo de apresentação, cada contexto foi agrupado com suas páginas, compondo assim a navegabilidade e apresentação da aplicação. Além disso, com as configurações para cada usuário, há a visibilidade ou não dos contextos navegacionais como mostra os modelos navegacionais das Figuras 49, 50 e 51.

Com a lógica e a apresentação criadas, então, a camada de interface foi criada para prover a interface de comunicação entre a apresentação do sistema e a lógica do negócio. Esta camada fará as chamadas de acordo com as requisições provindas da camada de apresentação que serão executadas na máquina do cliente. Deste modo, o ciclo de desenvolvimento de uma aplicação Web a partir do método SPL-OOWS fica completo, restando testar e manter o sistema.

5.4.1 Exemplo de variação de Produtos

Tomando como base o desenvolvimento de um produto sem a utilização da característica opcional de Area_conhecimento, temos a seleção das características mostradas na Figura 60.

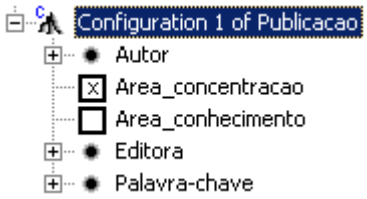


Figura 60: Configuração do modelo de características para a configuração da aplicação de referência

Com essas características selecionadas, os diagramas dinâmico de estados deverão sofrer adaptações para atender estas características, desta forma, a Figura 61 representa o diagrama dinâmico de estados para o caso de uso Gerenciar Área de Concentração e a Figura 62 mostra o diagrama de estados para o caso de uso Gerenciar Publicações .

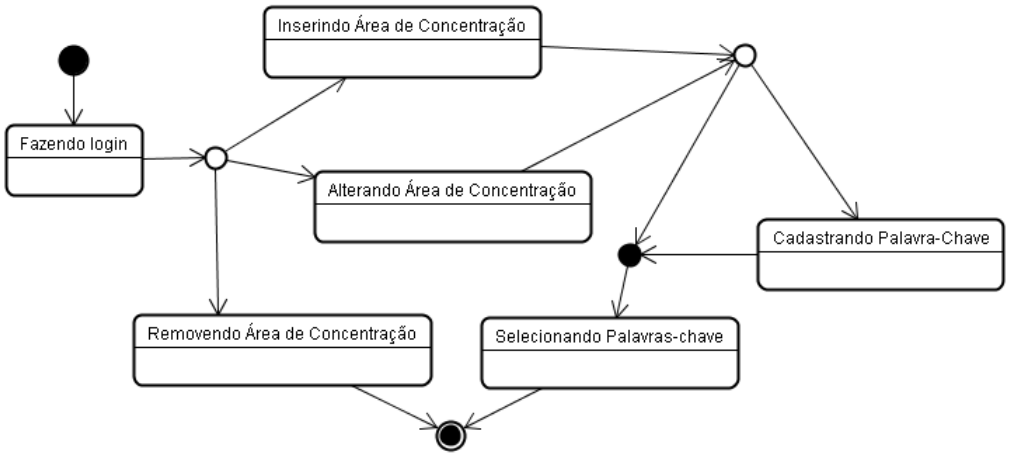


Figura 61: Diagrama dinâmico de estados do caso de uso Gerenciar Área de Concentração

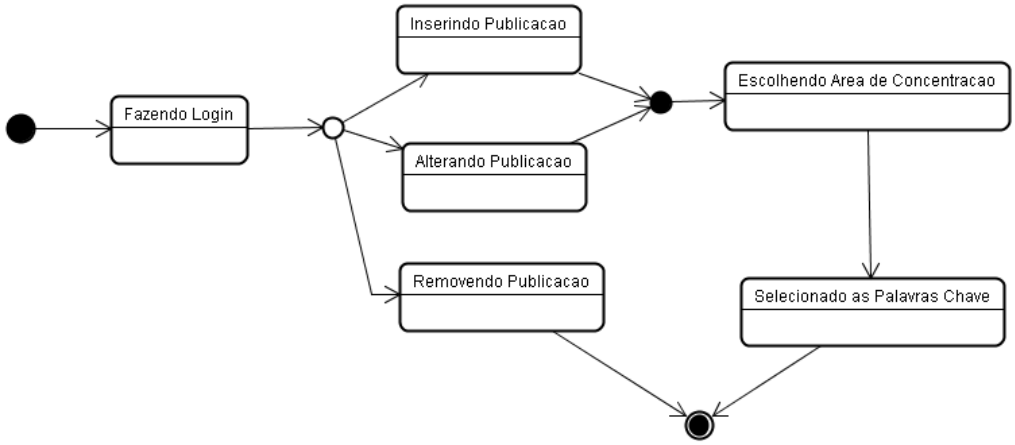


Figura 62: Diagrama dinâmico de estados do caso de uso Gerenciar Publicações

A aplicação terá o diagrama de classes representado pela Figura 63, diferente do diagrama de classes da Figura 48 este, não contém a classe `AreaConhecimento`.

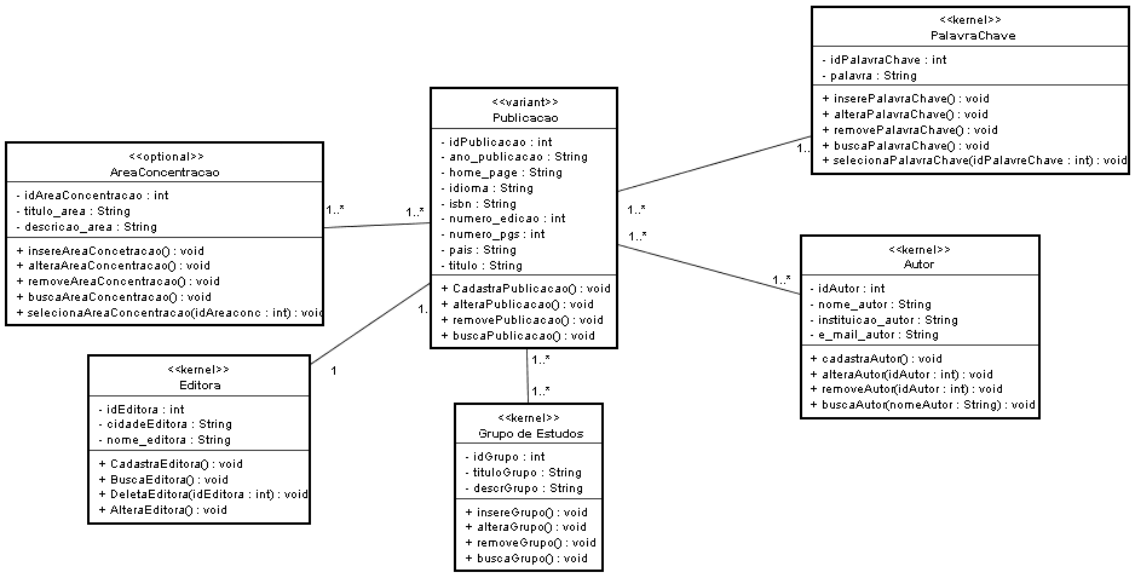


Figura 63: Novo diagrama de classes a partir da seleção do modelo de características

A Figura 64 representa a visão geral do produto gerado a partir da configuração do modelo de características na Figura 60.

Figura 64: Visão Geral do Sistema com os Contextos navegacionais sem a utilização de "Áreas de Conhecimento"

5.5 Avaliação do método

Este exemplo de aplicação permitiu observar os benefícios e os problemas relacionados à aplicação do método SPL-OOWS. Assim, como principais benefícios destacam-se:

- foi criada uma padronização para criação de infra-estrutura básica, composta de Gerenciador de Banco de dados, Gerenciador de Usuários e Gerenciador de Formulários, o que melhora a fase de especificação, pois o desenvolvedor não terá que se preocupar em redefinir os elementos da infra-estrutura, diminuindo assim, este tempo de desenvolvimento. Depois de especificado o domínio de uma LPS, é possível o gerenciamento destes elementos a fim de se gerar uma família de produtos, caracterizando assim uma LPS;

- a utilização do modelo de características e a criação de estereótipos nos casos de uso e classes no desenvolvimento de aplicações Web torna os domínios mais explícitos e assim aumenta as possibilidades de reutilização. Com os métodos tradicionais, não há visibilidade do que é obrigatório e opcional nas aplicações;

- o método permite que com o amadurecimento dos domínios, as características possam evolutivamente se tornar disponível para reutilização, melhorando cada vez mais o tempo de desenvolvimento;

- o método SPL-OOWS corrige problemas do OOWS apontados por Bianchini (2008) como a não definição de um processo de desenvolvimento bem estruturado, pois introduz claramente as etapas de Requisitos, Análise e Projeto para a modelagem conceitual.

Por outro lado também observamos problemas na utilização do método, dos quais destacam-se:

- a adoção de uma abordagem de LPS exige um investimento inicial de tempo e treinamento. Assim, o uso inicial do método pode ser mais lento que um método convencional, pois existe a necessidade de especificação do domínio prevendo as possíveis opções e variabilidades, porém ao desenvolver várias aplicações para um mesmo domínio o investimento inicial é compensado;

- a ausência de ferramentas de apoio para a geração automática de código a partir das definições de uma LPS implica em maior tempo de desenvolvimento;

- o método exige habilidade do desenvolvedor em trabalhar com modelos, o que é não propriamente uma desvantagem, mas pode encontrar barreiras em um mercado cujo desenvolvedores estão acostumados com frameworks de geração de código puramente baseados em linguagem de programação.

Capítulo

6

Conclusão

Esta dissertação propôs o método SPL-OOWS que incorpora técnicas de reutilização ao desenvolvimento de aplicações Web unindo os métodos OOWS (FONS et al., 2001), PLUS (GOMAA, 2005) e o PGV (OLIVEIRA JUNIOR, 2005). Esta união de conceitos permite a construção de famílias de aplicações Web utilizando os conceitos de LPS que podem ser utilizadas para melhorar a produtividade e a qualidade dos produtos de software.

A criação de uma infra-estrutura básica para as aplicações Web evita que o desenvolvedor se preocupe em modelar elementos comuns da maioria dessas aplicações, como o gerenciamento de usuário, a validação de formulários e o gerenciamento do banco de dados, pois esses elementos farão parte de todas as aplicações criadas a partir do SPL-OOWS. Assim, o desenvolvedor tem apenas que configurar esses elementos para atender as necessidades de um produto específico. Outras características de um determinado domínio podem ser acrescentadas à infra-estrutura básica aumentando assim cada vez mais a capacidade de reutilização daquele domínio.

Para a avaliação do método, foram desenvolvidos dois exemplos, um de comércio eletrônico e outro de gerenciamento de publicações. Com isso foi possível constatar que o método SPL-OOWS alcança o objetivo de trazer recursos de reutilização que podem permitir uma maior eficiência ao processo de desenvolvimento de aplicações Web a partir do momento em que existe a infra-estrutura básica para apoiar o desenvolvimento de tarefas comuns de qualquer aplicação Web. Porém, o método pode ser melhorado com extensões como a definição de uma estratégia de desenvolvimento dos modelos dinâmicos de estados que seja alterado de forma automática de acordo com as características selecionadas no modelo de características. Além disso, o método não possui foco no desenvolvimento do modelo de

apresentação, por isso, existem melhorias que podem ser realizadas para o desenvolvimento deste modelo, como a inclusão de conceitos de interfaces plásticas (TRINDADE et al., 2007), que possui como principal propriedade a possibilidade de adaptação para diferentes tipos de contextos.

Pode-se evidenciar que apesar do custo de adoção do método, com o amadurecimento dos domínios, o tempo de desenvolvimento tende a diminuir e a produtividade a aumentar, pois não será necessário criar novos modelos, e sim configurar os modelos existentes para satisfazer as necessidades de uma aplicação diferente de uma mesma família. A utilização do modelo de características e a criação de estereótipos nos casos de uso e classes no desenvolvimento de aplicações Web torna os domínios mais explícitos e assim aumenta as possibilidades de reutilização. O SPL-OOWS também corrige alguns defeitos, como a falta de estruturação clara do OOWS apontados por Bianchini (2008), ao introduzir e definir as etapas de Requisitos, Análise e Projeto.

Como trabalhos futuros, é importante que o método SPL-OOWS seja utilizado em experimentos mais amplos que permitam uma avaliação mais profunda de sua aplicação. Também é importante o projeto e implementação de uma ferramenta que automatize o processo proposto contemplando a geração automática de código. Para isso, é necessário a escolha de uma tecnologia para geração automática de código a partir da qual uma estratégia de implementação possa ser concebida. Outras contribuições esperadas são referentes a criação de novos conceitos que sejam aplicáveis a este método como a introdução de aspectos, a incorporação de modelos orientados a serviços e o monitoramento de contratos eletrônicos para cada serviço incorporado à LPS para aplicações Web.

Referências Bibliográficas

ATKINSON, C. BAYER, J., BUNSE, C., KAMSTIES, E., LAITENBERGER, O., LAQUA, R., MUTHING, D., PAECH, B., WÜST, J., ZETTEL, J. *Component-Based Product-Line Engineering with UML*. Boston: Addison-Wesley, 2001.

ANTKIEWICZ, M., CZARNECKI, K., Feature Plugin: Feature Modeling Plug-In for Eclipse, In: OOPSLA'04 Eclipse Technology eXchange (ETX) Workshop, Oct. 24-28, 2004, Vancouver. Procedings…Vancouver,2004

ARGOUWE, <http://www.pst.informatik.uni-muenchen.de/projekte/uwe/argouwe.shtml> Acesso em 1 de junho de 2007.

BAYER, J.; FLEGE, O.; KNAUBER, P.; LAQUA, R.; SCHMID, K.; WIDEN, T.; DEBAUD, J. PuLSE: a methodology to develop software product lines. In: SYMPOSIUM ON SOFTWARE REUSABILITY, 5., 1999, Los Angeles. Proceedings...Los Angeles, 1999. p. 122-131.

BECKER, M. Towards a general model of variability in product families. In: SOFTWARE VARIABILITY MANAGEMENT WORKSHOP, 2003, Portland. Proceedings… Portland, 2003. p.19-27.

BIANCHINI, S., *Avaliação de métodos de desenvolvimento de aplicações Web*, São Carlos, fevereiro de 2008. p. 113. Dissertação (Mestrado em Ciências de Computação e Matemática Computacional), ICMC-USP.

BOSCH, J., *Design & use of software architectures: adopting and evolving a product-line approach*. Boston: Addison Wesley, 2000.

CARE, http://www.care-t.com/news-events/pr_040105.asp. Acesso em 30 de maio de 2007.

CERI,S., et al., *Designing data-intensive Web Application*. Italia: Morgan Kaufmann, 2004.

CERI.S., et al., Web Modeling Language (WebML): a modeling language for design sites. In: 9th International World Wide Web Conference The Web: The Next Generation, 15-19 de Maio 2000, Amsterdan. Anais eletrônico em < http://www9.org/w9cdrom/177/177.html > acesso em 1 nov. 2008. Amsterdan, 2000.

CLEMENTS, P.; NORTHORP, L., *Software Product Lines: Practices and Patterns*. Boston: Addison-Wesley, 2002.

CZARNECKI, K.; ANTKIEWICZ, M. Mapping Features to Models:A Template Approach Based on Superimposed Variants. Technical Report # 2005-05, Waterloo University -Canada – 2005.

CZARNECKI, K.; HELSEN, S.; EISENECKER, U. Staged configuration through specialization and multi-level configuration of feature models. To appear in special issue on "Software Variability: Process and Management", Software Process Improvement and Practice, 10(2), 2005.

DOMINGUES, A.L. S.; BIANCHINI, S. L.; COSTA, M. L.; FERRARI, F. C.; MALDONADO, J. C.. Web application development methods: A comparision. In: Webmedia´2007: Proceedings of the 13º Brazilian Symposium on Multimedia and the Web, Gramado, Brasil, 2007.

ECLIPSE, <http://www.eclipse.org/>, acesso em 03 de Nov. de 2008

FONS, J., et al., Development of Web Applications from Web Enhanced Conceptual Schemas. In: WWW2003,May 20–24,2003,Budapest,Hungary, Hungria, 2003.

FONS, J., et al., Extending an OO Method to Develop Web Applications. In: CYTED Program, Project VII.18, WEST and the FEDER-CICYT Project with ref. TIC 1FD97-1102 and the Technical University of Valencia, Spain, Espanha – 2001.

GAMMA, E., HELM, R., JOHNSON, R., VLISSIDES, J., Design Patterns Elements of Reusable Object Oriented Software. Addison Wesley Professional Computing Series, 1995

GIMENES, I. et al; O projeto preliminar de WIDE-PL. Estágio de pós-doutorado na Universidade de Waterloo, Canadá, 2005.

GIMENES, I. M. S.; TRAVASSOS, G. H. O enfoque de linha de produto para desenvolvimento de software. In: JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA DA SBC, 22., 2002, Florianópolis. Anais... Florianópolis, 2002. p. 34.

GOMAA, H., *Designing Software Product Lines With UML: From uses cases to Pattern-based software Architecture*. Addison Wesley Object-Oriented Technology Series, 2005.

GRISS, M. L.; FAVARO, J.; D´ALESSANDRO, M. Integrating feature modeling with the RSEB. In: INTERNATIONAL CONFERENCE ON SOFTWARE REUSE, 5., 1998, Washington. Anais… Washington, 1998. p. 76-85.

JACOBSON, I., BOOCH, G., RUMBAUGH, J., The Unified Software Development Process, Addison Wesley, 1999.

JUDE, http://jude.change-vision.com/jude-web/product/community.html, Acessado em 10 de agosto de 2008.

KANG, K., Feature-oriented domain analysis (FODA) - feasibility study. Technical Report CMU/SEI-90-TR-21, SEI/CMU, Pittsburgh, 1990.

KANG, K.; KIM, S.; KIM, K.; KIM, G.; SHIN, E. FORM: A Feature-Oriented Reuse Method with Domain-Specific Refrence Architecture, SEI Technical Report, 1998.

KNAPP, A., et al. Modeling Business process in web applications with ArgoUWE. Munic, Alemanha, 2004.

KOCH, N., "Software Engineering for Adaptive Hypermedia Applications", PhD. Thesis, Reihe Softwaretechnik 12, Uni-Druck Publishing Company, Munich. Alemanha - 2001.

KOCH, N., KRAUS, A., The Expressive Power of UML-Based Web Engineering. Ludwig-Maximilians-Universität München. Alemanha – 2002.

LIMA, F., SCHWABE, D., Aplication Modeling for the Semantic Web. In: Proceedings of the First Conference on Latin American Web Congress, 10-12 de Nov de 2003, Santiago, Chile, 2003.

MOLINA, P., MELIA, S.,PASTOR, O., JUST-UI: a user interface specification model. In: CADUI 2002- Computer-Aided Design of User Interfaces III, Proceedings of the Fourth International Conference on Computer-Aided Design of User Interfaces, Maio, 15-17, 2002, Valenciennes, France,2002.

MOURA, S., SCHWABE, D., Interface Deployment for Hypermedia Applications in the semantic Web. In: LA-WEB 2004, 12-15 de out., 2004, Ribeirão Preto, Brasil, 2004.

NUNES, A., HyperDE Um *Framework* e Ambiente de Desenvolvimento Dirigido por Ontologias para Aplicações Hipermídia. Rio de Janeiro, 2005. Dissertação (Mestrado em Ciência da Computação) PUC-RJ.

OLIVEIRA JUNIOR, E., *Um Processo de gerênciamento de variabilidade para linha de produto de software*. Maringá, abril de 2005. p. 155, Dissertação (Mestrado em Ciência da Computação). PCC-DIN / UEM.

OLIVEIRA JUNIOR, E; GIMENES, I; HUZITA, E; MALDONADO, J. A Variability Management Process for Software Product Lines. In: CASCON 2005, 2005, Toronto. Proceeding of Cascon 2005. Ottawa, Canada: National Research Council Canada, 2005.

PASTOR, O. et. al, The OO-Method Approach for information systems Modelling: From Object-Oriented Conceptual Modeling to Automated Programing. Information Systems, Volume 26, Numero 7, pp. 507-534, Nov. de 2001.

PASTOR, O; INSFRÁN, E., The OO-Method, The methodological support for Oliva Nova Modele Execution System, In: Technical Report – CARE Technologies, 1999.

PELECHANO, V. et al.,Developing Web Applications from Conceptual Models. A Web Services Approach. In: ECOMO03 - MCYT Project with ref. TIC2001-3530-C02-01, Universidad Politecnica de Valencia, Espanha – 2003.

SCHWABE, D., ROSSI, G., An Object Oriented Approach to Web-Based Application Design. In: TAPOS (Theory and Practice of Object Systems), Wiley and Sons, Out. 1998, Nova York, 1998.

SEI - Software Engineering Institute. A framework for software product line practice. Pittsburgh. Disponível em < http://www.sei.cmu.edu/productlines/framework.html >. Acesso em: 4 de nov. de 2008.

SIMONS, M.; CREPS, D.; KLINGLER, C.; LEVINE, L.; ALLEMANG, D. Organization domain modeling (ODM) guidebook, version 2.0. Technical Report STARS-VC-A025/001/00, Lockheed Martin Tactical Defence Systems, 1996.

SOCHOS, P.; PHILIPPOW, I.; RIEBISCH, M. Feature-oriented development of software product lines: mapping feature models to the architecture. Springer, LNCS 3263, p. 138-152. , 2004

SPC - SOFTWARE PRODUCTIVITY CONSORTIUM. Reuse-Driven Software Processes Guidebook. SPC-92019-CMC version 02.00.03 November 1993.

TRINDADE, F. M. , MOREIRA, A.A., PIMENTA, M.S., UsiXML4ALL - Uma Ferramenta para Criação de Aplicativos Multiplataforma, UFRGS, Porto Allegre, 2007. Artigos eletrônicos em <www.usixml.org/index.php?mod=download&file=ext_papers/Trindade-UsiXML4ALL.pdf> acesso em 15 de Novembro de 2008.

VAN GURP, J., BOSCH, J., SVAHNBERG, M. On the notion of variability in software product lines, In: Proc. The Working IEEE/IFIP Conference on Software Architecture (WICSA), Amsterdam, Holanda, 2001.

VAN GURP, J.; BOSCH, J. On the notion of variability in software product lines. In: THE WORKING IEEE/IFIP CONFERENCE ON SOFTWARE ARCHITECTURE, 2001, Amsterdam. Proceedings… Amsterdam, 2001.

WEBRATIO, http://www.webratio.com, acessado em 3 de junho de 2007.

WEISS, D.; CHI TAU, R. L. *Software product-line engineering: a family-based software development process*. Boston: Addison-Wesley, 1999.

ZAUPA, F., et al., Um Processo de Desenvolvimento de Aplicações Web baseado em Serviços. In: SBCARS - Simposio Brasileiro de Componentes, Arquiteturas e Reutilização de Software - Campinas, 2007 / Anais eletrônico < http://www.ic.unicamp.br/sbcars2007/tecnicas/files/sbcars2007-zaupa-processo.pdf >, acesso em 10 de jan. de 2007, 2007.

Apêndice – A – Diagramas do estudo de caso ilustrativo do comércio eletrônico

Descrição dos casos de uso do comércio eletrônico

Caso de Uso: Controlar Entrega
Tipo: Obrigatório (kernel)
Objetivo: Gerenciar a entrega de acordo com a solicitação do cliente. Esta entrega pode ser via transportadora ou correios.
Casos de Uso relacionados: Gerenciar produtos, Gerenciar acesso.
Atores: Com.Registrado
Pré-Condição: Usuário registrado efetuar uma compra e necessitar escolher uma forma de pagamento.
Pós-Condição: Produto ser enviado ao cliente após este efetuar o pagamento.

Caso de Uso: Gerenciar Produtos
Tipo: Obrigatório (kernel)
Objetivo: Controlar o cadastro de produtos no mercado eletrônico.
Casos de Uso relacionados: Controlar entrega, gerenciar pedido e gerenciar acesso
Atores: Com.Registrado
Pré-Condição: Produto não está registrado, ou necessita de alteração ou necessita ser removido.
Pós-Condição: Produto registrado, ou alterado ou removido.

Caso de Uso: Controlar Estoque
Tipo: Obrigatório (kernel)

Objetivo: Controlar a entrada e saída de produtos.
Casos de Uso relacionados: Gerenciar Produtos
Atores: Com.Registrado
Pré-Condição: Necessidade de alteração do número de produtos em estoque.
Pós-Condição: Estoque do produto atualizado.

Caso de Uso: Montar carrinho de compras
Tipo: Alternativo (alternative)
Objetivo: Compor produtos para efetuar a compra.
Casos de Uso relacionados: Gerenciar Produtos.
Atores: Com.Registrado.
Pré-Condição: Produto não está no carrinho de compras, OU, o produto está no carrinho, mas com a quantidade errada, OU, o produto está no carrinho e deverá ser excluído.
Pós-Condição: Produto está no carrinho de compras, OU, a quantidade é alterada, OU, o produto é excluído.

Caso de Uso: Montar Lista de compras
Tipo: Alternativo (alternative)
Objetivo: Compor lista de desejo de produtos com o intuito de criação de um orçamento.
Casos de Uso relacionados: Gerenciar Produtos.
Atores: Com.Registrado
Pré-Condição: Produto não está na lista de compras, OU, o produto está na lista, mas com a quantidade errada, OU, o produto está na lista e deverá ser excluído.
Pós-Condição: Produto está na lista de compras, OU, a quantidade é alterada, OU, o produto é excluído da lista.

Caso de Uso: Gerenciar Pagamento
Tipo: Obrigatório (kernel)
Objetivo: Gerenciar o pagamento de um pedido, podendo assumir qual a forma de pagamento e parcelamento da compra.
Casos de Uso relacionados:
Atores:
Pré-Condição:
Pós-Condição:

Caso de Uso: Gerenciar pedido
Tipo: Obrigatório (kernel)
Objetivo: Armazenar as informações dos produtos de um pedido, podendo finalizar a venda ou cancelar o pedido.
Casos de Uso relacionados:
Atores:
Pré-Condição:
Pós-Condição:

Caso de Uso: Gerenciar Acesso
Tipo: Obrigatório (kernel)
Objetivo: Gerenciar o acesso de cada tipo de usuário permitindo ou não o acesso a determinados conteúdos na aplicação.
Casos de Uso relacionados:
Atores:
Pré-Condição:
Pós-Condição:

Diagramas dinâmicos de Estados do comércio eletrônico

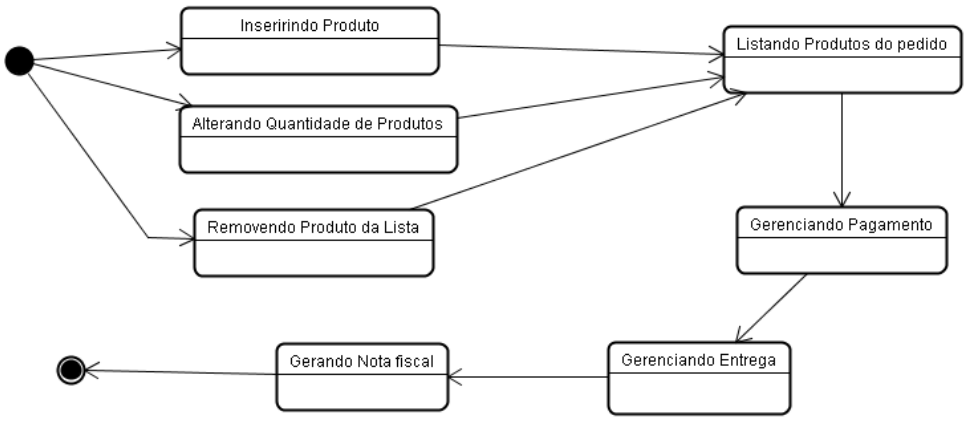


Figura 65: Gerenciar Pedido

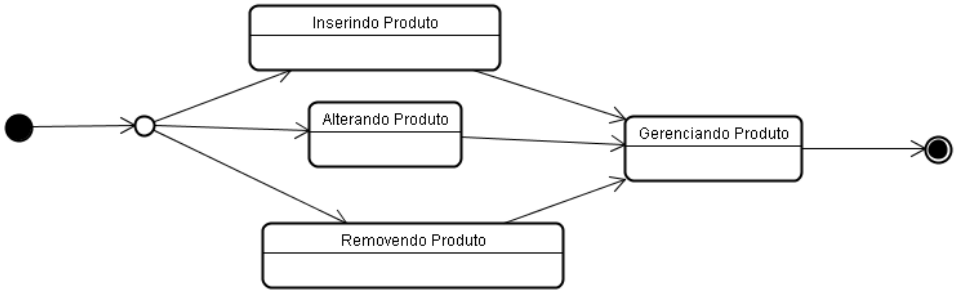


Figura 66: Gerenciar Produtos

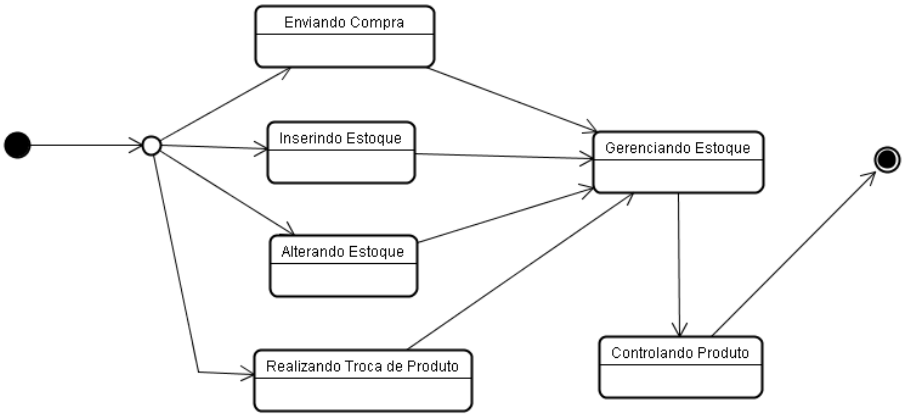


Figura 67 : Controlar Estoque

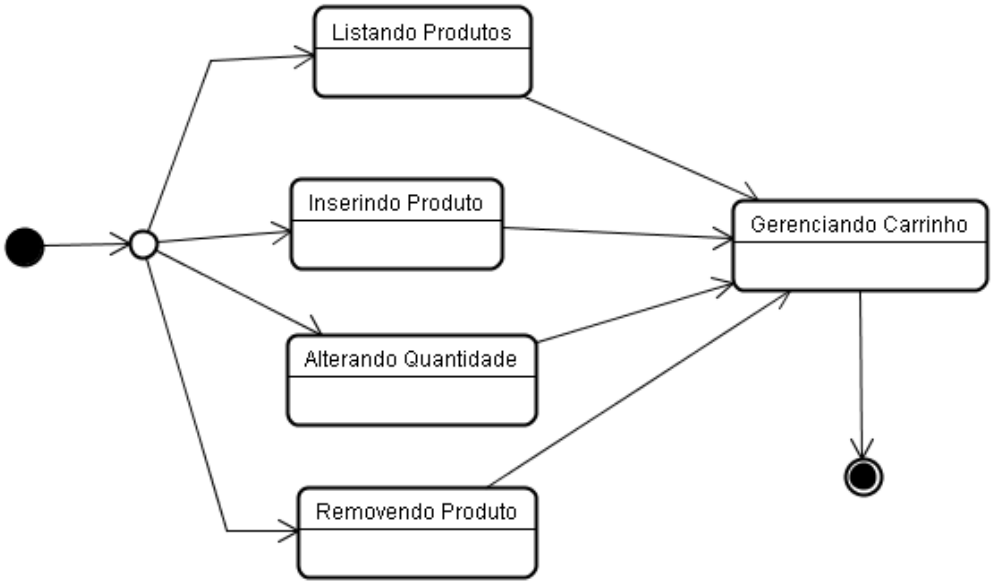


Figura 68 :Montar Carrinho de Compras / Lista de desejos

Apêndice – B – CODIGO FONTE DO CONTROLADOR DE USUÁRIOS

```
<?
class ControleUsuario{
	var $idUser;
	var $NomeUser;

function ControleUsuario(){
	session_start();
	if($this->Logado()){
		$this->idUser = $_SESSION['idUser'];
		$this->NomeUser = $_SESSION['NomeUser'];
	}
	else {
		$this->Logout();
	}
}
```

```
function Logout(){
      session_unset();
      session_destroy();
      return "Você saiu do sistema com sucesso!";
}
function Logar($bd){
      session_unset();
      //session_destroy();
      session_start();
      //include_once "../Connections/blog.php";
      $login = mysql_escape_string($_POST['login']);
      $senha = mysql_escape_string($_POST['senha']);

      $query = "SELECT * FROM Usuario WHERE loginUsuario = '$login'
AND senhaUsuario = '$senha'";
      $sql = $bd->Query($query);


      if(mysql_num_rows($sql) == 1) {
            $res = mysql_fetch_array($sql);
            $_SESSION['idUser'] = $res['idUsuario'];
            $_SESSION['NomeUser'] = $res['nomeUsuario'];
            //$_SESSION['NivelAcesso']                                    =
$res['NivelAcesso_idNivelAcesso'];
            $this->idUser = $_SESSION['idUser'];
            $this->NomeUser = $_SESSION['NomeUser'];
            //$this->idNivelAcesso = $_SESSION['NivelAcesso'];
            return $this->ISLogado();
      }
      else {
            return $this->NotLogado();
      }

}
function Logado(){
            if(isset($_SESSION['idUser'])){
                  //$this->Logado();
                  return 1;
            }
            else {
                  //$this->NotLogado();
                  return 0;
            }
}
function NotLogado($pasta){
      return '<a href="'.$pasta.'">Efetue Login! - Clique

      aqui!</a>';
}
function IsLogado(){
      return 'Você está logado como: '.$this->NomeUser.' | <a
href="sair.php">Sair</a>';

}
function VePermissaoCtx($contexto){

      $select = "SELECT Permissao FROM permissao WHERE
(GrupoUsuario_idGrupoUsuario IN (Select idGrupoUsuario FROM
grupousuario,              usuario_has_grupousuario              WHERE
Usuario_idUsuario='".$this->idUser."' AND GrupoUsuario_idGrupoUsuario
= idGrupoUsuario ) OR
```

```
Usuario_idUsuario IN ( Select idUsuario FROM usuario WHERE idUsuario =
'".$this->idUser."') ) AND Contexto_idCOntexto=".$contexto;

      $sql = mysql_query($select) or die ($select.mysql_error());
      $res = mysql_fetch_array($sql);

      // posição 0 -> Insercao | 1-> Alteração | 2-> Remoção | 3-
>Visualização
      $permissoes = $res['Permissao'];
      $permissao['Insercao'] = $permissoes[0];
      $permissao['Alteracao'] = $permissoes[1];
      $permissao['Remocao'] = $permissoes[2];
      $permissao['Visualizacao'] = $permissoes[3];
      return $permissao;
}
function VePermissaoAtr($atributo){

      $select    =    "SELECT    Permissao    FROM    permissao    WHERE
(GrupoUsuario_idGrupoUsuario    IN    (Select    idGrupoUsuario    FROM
grupousuario,              usuario_has_grupousuario              WHERE
Usuario_idUsuario='".$this->idUser."'  AND  GrupoUsuario_idGrupoUsuario
= idGrupoUsuario  ) OR
Usuario_idUsuario IN ( Select idUsuario FROM usuario WHERE idUsuario =
'".$this->idUser."') ) AND Atributo_idAtributo='".$atributo."'";

      $sql = mysql_query($select) or die ($select.mysql_error());
      $res = mysql_fetch_array($sql);

      // posição 0 -> Insercao | 1-> Alteração | 2-> Remoção | 3-
>Visualização
      $permissoes = $res['Permissao'];
      $permissao['Insercao'] = $permissoes[0];
      $permissao['Alteracao'] = $permissoes[1];
      $permissao['Remocao'] = $permissoes[2];
      $permissao['Visualizacao'] = $permissoes[3];
      return $permissao;
}
function VePermissaoPg($pagina){



$select     =     "SELECT     Permissao     FROM     permissao     WHERE
(GrupoUsuario_idGrupoUsuario    IN    (Select    idGrupoUsuario    FROM
grupousuario,              usuario_has_grupousuario              WHERE
Usuario_idUsuario='".$this->idUser."'  AND  GrupoUsuario_idGrupoUsuario
= idGrupoUsuario  ) OR
Usuario_idUsuario IN ( Select idUsuario FROM usuario WHERE idUsuario =
'".$this->idUser."') ) AND Pagina_idPagina='".$pagina."'";

      $sql = mysql_query($select) or die ($select.mysql_error());
      $res = mysql_fetch_array($sql);

      // posição 0 -> Insercao | 1-> Alteração | 2-> Remoção | 3-
>Visualização
      $permissoes = $res['Permissao'];
      $permissao['Insercao'] = $permissoes[0];
      $permissao['Alteracao'] = $permissoes[1];
      $permissao['Remocao'] = $permissoes[2];
      $permissao['Visualizacao'] = $permissoes[3];

      return $permissao;
```

```
}

}// classe

?>
```

Apêndice – C – código fonte do Gerenciador de Banco de dados para mysql

```
<?

class mysql {

	/*
	Classe para fazer tramitações no banco de dados
	$nomebd = nome do banco de dados
	$usuario = Usuario do BD
	$host = Endereço do Host
	$senha = Senha do banco de dados

	$tabela = nome da tabela a ser manipulada
	$camposTab = arranjo que contem nome dos campos da tabela
	$camposIns = arranjo que contem Valor dos Campos

	$chave = arranjo que contem o(s) nome(s) campo(s) chave
	$chaveVal = arranjo que contem o valor de cada campo chave
	*/
	var $ultimoId;
	function bdsql($nomebd,$usuario,$senha,$host){
		//fazer função para conectar ao banco de dados
		$conn = mysql_connect($host,$usuario,$senha);
		mysql_select_db($nomebd,$conn);
	}
	function Update2($tabela,$campos,$where){
		$query  =  "UPDATE  ".$tabela."  SET  ".$campos."  WHERE
".$where;
		mysql_query($query) or die ($query." - ".mysql_error());
	}
	function InsereBD($tabela,$campoTab,$campoIns){
		//função para inserção no banco de dados
		$ins = "INSERT INTO ".$tabela." ( ";
		$tam = sizeof($campoTab);
		$ins2 = "";
		for($i=0;$i<$tam;$i++)  {
			$ins = $ins.$campoTab[$i];
			//if ($campoTab[$i]!=0)
			$ins2 = $ins2."'".$campoIns[$campoTab[$i]]."'";
			if(($i+1)<$tam){
				$ins = $ins.", ";
```

```
                        $ins2 = $ins2.", ";
                        }
            }
            $ins = $ins.") VALUES ( ".$ins2.")";
            mysql_query($ins)    or    die    ("Erro    :    ".$ins."
".mysql_error());
            $this->ultimoId = mysql_insert_id();
            return "<br> Inserido com sucesso ";
      }
      function VerificaChave($nomeCampo,$vetorChave){
            foreach ($vetorChave as $elemento){
                  if ($elemento == $nomeCampo)
                        return 1;
            }
            return 0;
      }
      function AlteraBD($tabela,$campoTab,$campoIns,$where){
            //função para alteração de um elemento da tabela no banco
de dados
            $alt = "UPDATE ".$tabela." SET ";
            $tam = sizeof($campoTab);
            for($i=0;$i<$tam;$i++)  {
            //if ($this->VerificaChave($campoTab[$i],$chave))
                  //if ($campoTab[$i]!=0)
                  $alt         =         $alt.$campoTab[$i]."         =
"."'".$campoIns[$campoTab[$i]]."'";
                  if(($i+1)<$tam){
                        $alt = $alt.", ";
                        }
            }
            $alt = $alt." WHERE  ".$where." ;";
            mysql_query($alt)    or    die    ("Erro    :    ".$alt."
".mysql_error());
            return "<br> Alterado com sucesso ";
      }
      function DeletaBD($tabela,$chave,$chaveVal){
            //função para remoção de um elemento do banco de dados
            //DELETE FROM <TABELA> WHERE <CAMPO CHAVE>
            $del = "DELETE FROM ".$tabela." WHERE ";
            $tam = sizeof($chave);
            for($i=0;$i<$tam;$i++){
                  $del = $del.$chave[$i]." = ".$chaveVal[$chave[$i]];
                  if(($i+1)<$tam){
                        $del = $del." AND ";
                  }
            }
            mysql_query($del)    or    die    ("Erro    :    ".$del."
".mysql_error());
            return "<br> Deletado com sucesso";
      }
      function Seleciona($campos,$from,$where){

            $select = "SELECT ".$campos." FROM ".$from;
            if ($where!="")
                  $select .=" WHERE ".$where;
            $sql  =  mysql_query($select)  or  die  ("Erro  na  consulta:
".$select." <br>".mysql_error());
            //echo $select."<br><br>";
            return $sql;
      }
      function Query($query){
```

```
            $sql = mysql_query($query) or die ("Erro na Quey:
".$query." <br>".mysql_error());
            return $sql;
      }
      function ConstroiWhere($chave,$chaveVal){
            $tam = sizeof($chave);
            $where ="";
            for($i=0;$i<$tam;$i++){
            $cha = $chave[$i];
                  if($chaveVal[$cha]!="")
                        $where       =       $where.$chave[$i]."       =
".$chaveVal[$cha];
                  else
                        $where = $where.$chave[$i]." = 0 "; // campo
chave esta vazio, então, cadastrar
                  if(($i+1)<$tam){
                        $where = $where." AND ";
                  }
            }
            return $where;
      }
}
?>
```

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Administração
Baixar livros de Agronomia
Baixar livros de Arquitetura
Baixar livros de Artes
Baixar livros de Astronomia
Baixar livros de Biologia Geral
Baixar livros de Ciência da Computação
Baixar livros de Ciência da Informação
Baixar livros de Ciência Política
Baixar livros de Ciências da Saúde
Baixar livros de Comunicação
Baixar livros do Conselho Nacional de Educação - CNE
Baixar livros de Defesa civil
Baixar livros de Direito
Baixar livros de Direitos humanos
Baixar livros de Economia
Baixar livros de Economia Doméstica
Baixar livros de Educação
Baixar livros de Educação - Trânsito
Baixar livros de Educação Física
Baixar livros de Engenharia Aeroespacial
Baixar livros de Farmácia
Baixar livros de Filosofia
Baixar livros de Física
Baixar livros de Geociências
Baixar livros de Geografia
Baixar livros de História
Baixar livros de Línguas

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo