LUIZ CARLOS MIYADAIRA RIBEIRO JR.

**UMA ARQUITETURA DE SOFTWARE PARA SISTEMAS ESPAÇO-
TEMPORAIS BASEADOS NA WEB PARA AGRICULTURA DE
PRECISÃO**

**São Paulo
2007**

LUIZ CARLOS MIYADAIRA RIBEIRO JR.

**UMA ARQUITETURA DE SOFTWARE PARA SISTEMAS ESPAÇO-
TEMPORAIS BASEADOS NA WEB PARA AGRICULTURA DE
PRECISÃO**

Tese apresentada à Escola Politécnica da
Universidade de São Paulo para a obtenção
do título de Doutor em Engenharia.

**São Paulo
2007**

LUIZ CARLOS MIYADAIRA RIBEIRO JR.

**UMA ARQUITETURA DE SOFTWARE PARA SISTEMAS ESPAÇO-
TEMPORAIS BASEADOS NA WEB PARA AGRICULTURA DE
PRECISÃO**

Tese apresentada à Escola Politécnica da
Universidade de São Paulo para a obtenção
do título de Doutor em Engenharia.

Área de concentração: Sistemas Digitais

Orientador: Prof. Livre-Docente Antonio
Mauro Saraiva.

**São Paulo
2007**

FICHA CATALOGRÁFICA

Ribeiro Junior, Luiz Carlos Miyadaira

Uma arquitetura de software para sistemas espaço-temporais baseados na Web para agricultura de precisão / L.C.M. Ribeiro Junior. -- São Paulo, 2007.

190 p.

Tese (Doutorado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Arquitetura de software 2.Sistemas de informação 3.Padrões de software 4.Agricultura de precisão I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

DEDICATÓRIA

Dedico este trabalho a minha amada esposa Karla
e a nosso tesouro mais precioso, nossa filha Luiza

AGRADECIMENTOS

Ao meu orientador, professor Saraiva. Obrigado por compartilhar comigo sua experiência, obrigado por me abrir as portas dessa instituição e obrigado pela amizade de sempre.

Aos funcionários e professores desta instituição, EPUSP que trabalham sem medir esforços para manter a excelência desta Universidade.

Ao amigo e colega Edson Murakami. Durante os últimos 5 anos trocamos muitas idéias e experiências. Finalmente, concluímos essa jornada e suas idéias foram fundamentais para conclusão deste trabalho.

Ao amigo de todas as horas, Mario Kubo.

Minha mãezinha, Hatsuko, sempre ela. Esta tese é fruto da formação e caráter que você me ensinou. Muito obrigado!

À minha esposa, Karla, minha incentivadora, meu porto seguro, meu grande amor. Muito obrigado por sua ajuda neste trabalho, mas acima de tudo, muito obrigado pelo seu empenho como esposa, sua dedicação como mãe e sua coragem como mulher.

Obrigado Luiza, por existir.

*“Trate bem a terra. Ela não foi doada a você
pelos seus pais. Ela foi emprestada a você
pelos seus filhos.”*

(Antigo provérbio do Quênia)

RESUMO

A demanda por sistemas de informações para Agricultura de Precisão (AP) tem sido crescente nos últimos anos, especialmente sistemas que manipulam informações espaciais e temporais baseados na Web. Normalmente as análises dos dados coletados em campo são realizadas por uma grande diversidade de softwares, muitas vezes proprietários e monolíticos, com pouca capacidade de compartilhamento de informações, além de não atenderem todas as necessidades desse tipo de sistema. A construção de sistemas de informações espaço-temporais para AP que permitem o gerenciamento das atividades agrícolas em uma abordagem integrada exige uma arquitetura flexível, interoperável e componentizada. Nesse contexto, esse trabalho de tese tem como objetivo desenvolver o projeto de uma arquitetura de software que permita a construção de sistemas espaço-temporais voltados para AP baseados na Web. Essa arquitetura é proposta num conjunto de três cenários arquiteturais baseados na plataforma JEE com suas especificações e um catálogo de padrões de análise e padrões de projeto espaço-temporais. Esses padrões definem soluções para problemas de análise e projeto que envolvem manipulação de dados espaço-temporais, possibilitando reutilização de conhecimento especializado e de código. Como resultado, este trabalho apresenta, como estudo de caso, a implementação de um componente para apresentação de mapas de produtividade em interface Web. Esse componente é baseado em um padrão de projeto do catálogo de padrões proposto e mostra como o conhecimento agrícola e o conhecimento técnico podem ser embutidos em um componente para utilização na construção de sistemas de informações espaço-temporais para AP.

Palavras-chave: Arquitetura de Software. Sistemas de Informações. Padrões de Software. Agricultura de Precisão.

ABSTRACT

Lately there has been a growing demand for information systems for Precision Agriculture (P.A.), especially for web-based systems that manipulate both spatial and temporal information. Field-collected data are often analyzed with various kinds of software, usually privately owned or monolithically used, also limited in scope and unable to share information. The development of P.A. spatio-temporal information systems that allow the management of agricultural activities through an integrative approach demands a flexible architecture, interoperable and component-structured. In this context, it is the objective of this thesis to present the development of a software architecture design that allows the construction of web-based spatio-temporal systems for Precision Agriculture. This software architecture is proposed in a set of three architectural scenes based on the JEE platform, with its specifications, and a catalog of both spatio-temporal analysis patterns and spatio-temporal design patterns. Such patterns define solutions to both analysis and design problems that involve spatio-temporal data manipulation, thus making possible the reuse of specialized knowledge and the reuse of code. This work presents as a result, in form of a case study, the implementation of a web-component for the presentation of productivity maps on web-interface. This web-component is based on a design pattern from the patterns catalog proposed in this work and it shows how agricultural understanding and technical understanding can be embedded in a component for construction of spatio-temporal information systems for P. A.

Keywords: Software Architecture. Information Systems. Software Patterns. Precision Agriculture.

ÍNDICE DE FIGURAS

FIGURA 2.1. ETAPAS DA AGRICULTURA DE PRECISÃO	27
FIGURA 2.2. (A) AMOSTRAGEM REGULAR, (B) AMOSTRAGEM ALEATÓRIA E (C) AMOSTRAGEM ALEATÓRIA ESTRATIFICADA	30
FIGURA 2.3. PACOTES DE CASOS DE USO DO MOSAICO	33
FIGURA 2.4. MODELO DE CASOS DE USO DO PACOTE “INSERIR DADOS DE CAMPO”	34
FIGURA 2.5. DIAGRAMA DE PACOTES COM OS SUBSISTEMAS DO MOSAICO	34
FIGURA 2.6. SUBSISTEMA DADOS DE CAMPO	36
FIGURA 2.7. PROJEÇÕES CARTOGRÁFICAS	37
FIGURA 2.8. SISTEMA DE COORDENADAS UTM	41
FIGURA 2.9. RELACIONAMENTOS TOPOLÓGICOS	45
FIGURA 2.10. TEMPO LINEARMENTE ORDENADO	46
FIGURA 2.11. TEMPO RAMIFICADO	46
FIGURA 2.12. OPERAÇÕES TEMPORAIS	49
FIGURA 3.1. MODELO GISER.....	55
FIGURA 3.2. MODELO DE ENTIDADES GEOGRÁFICAS	56
FIGURA 3.3. ENTIDADE GEOGRÁFICA CIDADE.....	57
FIGURA 3.4. (A) AGREGAÇÃO ESPACIAL. (B) UM EXEMPLO	57
FIGURA 3.5. AGRUPAMENTO ESPACIAL E UM EXEMPLO	58
FIGURA 3.6. CONTRUÇÕES ESPAÇO-TEMPORAIS	58
FIGURA 3.7. UM EXEMPLO DE MODELAGEM DE UM FENÔMENO ESPAÇO-TEMPORAL	59
FIGURA 3.8. PICTOGRAMAS ESPACIAIS	60
FIGURA 3.9. PICTOGRAMAS TEMPORAIS	60
FIGURA 3.10. HIERARQUIA DE TIPOS ESPACIAIS.....	61
FIGURA 3.11. RELACIONAMENTOS TOPOLÓGICOS	62
FIGURA 3.12. MODELAGEM TEMPORAL COM MADS.....	62
FIGURA 3.13. ESQUEMA ESPACIAL CONSTRUÍDO COM PVL.....	64
FIGURA 3.14. EXEMPLO DE UTILIZAÇÃO DO PVL ESPAÇO-TEMPORAL	65
FIGURA 3.15. SÍMBOLOS DE CLASSES	66
FIGURA 3.16. RELACIONAMENTO COVERAGE	66
FIGURA 3.17. RELACIONAMENTO <i>CONTAINMENT</i>	67
FIGURA 3.18. RELACIONAMENTO <i>PARTITION</i>	67
FIGURA 3.19. CLASSE E RELACIONAMENTO TEMPORAL DO MODELO GEOOOA.....	68

FIGURA 3.20. MODELO GEO-OMT	69
FIGURA 3.21. TUML: ESTEREÓTIPOS TEMPORAIS.....	71
FIGURA 3.22. MODELO GEOFRAME.....	73
FIGURA 3.23. MODELO TEMPORAL PARA O GEOFRAME.....	74
FIGURA 3.24. FRAMEWORK GEOFRAME-T	75
FIGURA 3.25. MODELO CONCEITUAL TERRALIB.....	77
FIGURA 3.26. INSTANCIAÇÃO DE GEOMETRIAS COM TERRALIB	78
FIGURA 3.27. OPERAÇÕES ESPACIAIS COM TERRALIB	79
FIGURA 3.28. ARQUITETURA DO DEEGREE.....	81
FIGURA 3.29. EVOLUÇÃO CRONOLÓGICA DOS MODELOS CONCEITUAIS DE DADOS PARA SIG	88
FIGURA 4.1. CENÁRIO DE APLICAÇÕES JEE MULTI-CAMADAS.....	95
FIGURA 4.2. CENÁRIO DE APLICAÇÕES JEE COM CLIENTES <i>STAND-ALONE</i>	96
FIGURA 4.3. CENÁRIO DE APLICAÇÕES JEE <i>WEB-CENTRIC</i>	96
FIGURA 4.4. CENÁRIO DE APLICAÇÕES JEE B2B.....	97
FIGURA 5.1. RELACIONAMENTO ENTRE MODELO DE REFERÊNCIA, ESTILO DE ARQUITETURA E ARQUITETURA DE REFERÊNCIA IMPLEMENTANDO UMA ARQUITETURA DE SOFTWARE ..	102
FIGURA 5.2. VISÃO GERAL DA ARQUITETURA DE REFERÊNCIA.....	102
FIGURA 5.3. ARQUITETURA DE APLICAÇÃO JEE BASEADO NA <i>WEB</i>	104
FIGURA 5.4. DIAGRAMA DE SEQÜÊNCIA PARA INSERIR UM TALHÃO	105
FIGURA 5.5. ARQUITETURA DE APLICAÇÃO JEE MULTICAMADAS	107
FIGURA 5.6. ARQUITETURA DE APLICAÇÃO JEE <i>STAND-ALONE</i>	109
FIGURA 5.7. ESTEREÓTIPOS ESPAÇO-TEMPORAIS.....	114
FIGURA 5.8. CATÁLOGO DE PADRÕES DE ANÁLISE E PROJETO ESPAÇO-TEMPORAIS	116
FIGURA 5.9. PADRÃO DE ANÁLISE TALHÃO E AMOSTRAS	118
FIGURA 5.10. PADRÃO DE ANÁLISE MAPAS DE TALHÕES.....	119
FIGURA 5.11. DIAGRAMA DE CLASSES: APRESENTAR OBJETOS ESPACIAIS	121
FIGURA 5.12. DIAGRAMA DE SEQÜÊNCIA: APRESENTAR OBJETOS ESPACIAIS.....	121
FIGURA 5.13. DIAGRAMA DE CLASSE: ATRIBUTOS TEMPORAIS	123
FIGURA 5.14. DIAGRAMA DE SEQÜÊNCIA: ATRIBUTOS TEMPORAIS	123
FIGURA 5.15. DIAGRAMA DE CLASSES: EXEMPLO DE ATRIBUTOS TEMPORAL	124
FIGURA 5.16. DIAGRAMA DE SEQÜÊNCIA: EXEMPLO DE ATRIBUTOS TEMPORAL	124
FIGURA 5.17. DIAGRAMA DE CLASSE: RELACIONAMENTOS TEMPORAIS.....	125

FIGURA 5.18. DIAGRAMA DE SEQÜÊNCIA: RELACIONAMENTOS TEMPORAIS	126
FIGURA 5.19. DIAGRAMA DE CLASSE: EXEMPLO DE RELACIONAMENTO TEMPORAL.	126
FIGURA 5.20. DIAGRAMA DE SEQÜÊNCIA: EXEMPLO DE RELACIONAMENTO TEMPORAL.	127
FIGURA 5.21. DIAGRAMA DE CLASSES: OBJETOS DE NEGÓCIO ESPAÇO-TEMPORAIS.....	128
FIGURA 5.22. DIAGRAMA DE CLASSES: EXEMPLO OBJETOS DE NEGÓCIO ESPAÇO-TEMPORAIS	128
FIGURA 5.23. DIAGRAMA DE CLASSES: PERSISTIR OBJETOS ESPAÇO-TEMPORAIS.....	130
FIGURA 5.24. DIAGRAMA DE SEQÜÊNCIA: PERSISTIR OBJETOS ESPAÇO-TEMPORAIS	131
FIGURA 5.25. DIAGRAMA DE CLASSE: EXEMPLO DE PERSISTIR OBJETOS ESPAÇO-TEMPORAIS	131
FIGURA 5.26. DIAGRAMA DE CLASSE: CONSULTAR OBJETOS ESPAÇO-TEMPORAIS.....	133
FIGURA 5.27. DIAGRAMA DE SEQÜÊNCIA: CONSULTAR OBJETOS ESPAÇO-TEMPORAIS	133
FIGURA 5.28. DIAGRAMA DE CLASSE: MONTAR OBJETOS ESPAÇO-TEMPORAIS COMPLEXOS	134
FIGURA 5.29. DIAGRAMA DE SEQÜÊNCIA: MONTAR OBJETOS ESPAÇO-TEMPORAIS COMPLEXOS	135
FIGURA 5.30. DIAGRAMA DE CLASSE: EXEMPLO DE MONTAR TALHÃO	135
FIGURA 5.31. DIAGRAMA DE SEQÜÊNCIA: EXEMPLO DE MONTAR TALHÃO	136
FIGURA 5.32. MODELO DE OBJETOS ESPACIAIS	139
FIGURA 5.33. EXTENSÃO DO MODELO MOSAICO.....	141
FIGURA 5.34. MAPEAMENTO DO MODELO DA ANÁLISE PARA MODELO DE PROJETO	142
FIGURA 5.35. MODELO DE OBJETOS DE TIPOS TEMPORAIS.....	142
FIGURA 5.36. MODELO DE PROJETO DADOS DE CAMPO.....	145
FIGURA 5.37. MODELO DE DADOS ESPAÇO-TEMPORAL	146
FIGURA 5.38. MAPEAMENTO O/R: TAG CLASS.....	148
FIGURA 5.39. MAPEAMENTO O/R: TAG SUBCLASS.....	148
FIGURA 5.40. MAPEAMENTO O/R: TAG PROPERTY	148
FIGURA 5.41. MAPEAMENTO O/R: TAG ASSOCIATION	149
FIGURA 5.42. EXEMPLO DE MAPEAMENTO O/R	150
FIGURA 5.43. INTERCÂMBIO DE DADOS ESPAÇO-TEMPORAIS	154
FIGURA 6.1. ESQUEMA DE CONVERSÃO DO PAML → SVG	158
FIGURA 6.2. ESTRUTURA BÁSICA DO DOCUMENTO SVG	159
FIGURA 6.3. FORMAS GEOMÉTRICAS DO SVG	160

FIGURA 6.4. TEXTO EM SVG	161
FIGURA 6.5. TRECHO DO ARQUIVO LAYOUT.XSL	163
FIGURA 6.6. DIAGRAMA DE CLASSE: COMPONENTE VIEWTALHAO	164
FIGURA 6.7. CENÁRIO ARQUITETURAL UTILIZADO PARA IMPLEMENTAÇÃO E TESTE DO COMPONENTE VIEWTALHÃO.....	165
FIGURA 6.8. MAPA DO TALHÃO DE AMOSTRAS BRUTAS (UMIDADE DO GRÃO DE MILHO)	166
FIGURA 6.9. MAPA DO TALHÃO DE AMOSTRAS FILTRADAS (UMIDADE DO GRÃO DE MILHO) .	167
FIGURA 6.10. MAPA DO TALHÃO DE AMOSTRAS FILTRADAS (TAXA DE PRODUTIVIDADE).....	168
FIGURA 6.11. TRECHO DE PÁGINA JSP COM CHAMADA PARA O COMPONENTE VIEWTALHAO	169

LISTA DE QUADROS

QUADRO 2.1. PRINCIPAIS PROJEÇÕES CARTOGRÁFICAS.....	38
QUADRO 2.2. VISÃO DE CAMPOS	43
QUADRO 3.1. PVL PARA MODELAR GEOMETRIAS DE OBJETOS	65
QUADRO 3.2. FUNCIONALIDADES DO DEEGREE.	82
QUADRO 3.3. COMPARAÇÃO ENTRE MODELO DE REQUISITOS E MODELO DE ANÁLISE	83
QUADRO 3.4. COMPARAÇÃO ENTRE MODELO DE ANÁLISE E MODELO DE PROJETO	84
QUADRO 3.5. COMPARAÇÃO ENTRE PADRÕES DE ANÁLISE E PADRÕES DE PROJETO	87
QUADRO 5.1. AVALIAÇÃO DAS PROPRIEDADES DE QUALIDADE DOS CENÁRIOS DE ARQUITETURA.	112
QUADRO 5.2. OPERAÇÕES GEOGRÁFICAS.....	141
QUADRO 5.3. OPERAÇÕES TEMPORAIS.....	144
QUADRO 6.1. INFRA-ESTRUTURA DE SOFTWARE DO COMPONENTE VIEWTALHAO	165

LISTA DE ABREVIATURAS E SIGLAS

AP	Agricultura de Precisão
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
ASCII	American Standard Code for Information Interchange
ATAM	Architecture Trade-off Analysis Method
CASE	Computer-aided Software Engineering
CORBA	Common Object Request Broker Architecture
COTS	Commercial Off-The-Shelf
CRUD	Create, Read, Update and Delete
CSS	Cascading Style Sheets
CS-W	Web Catalog Service
DAO	Data Access Object
DTD	Document Type Definition
DW	Data Warehouse
EER	Extended Entity- Relationship
EJB	Enterprise Java Beans
EPUSP	Escola Politécnica da Universidade de São Paulo
Geo-ER	Geographic Entity-Relationship
GeoOOA	Geographic Object Oriented Analysis
GIF	Graphics Interchange Format
GISER	Geographical Information Systems Entity Relationship
GML	Geographic Markup Language
GMOD	Geographic Data Model
GNSS	Global Navigation Satellite Systems
GoF	Gang of Four
GPS	Global Positioning System
GUI	Graphical User Interface
HTTP	HyperText Transfer Protocol
ISO	International Organization for Standardization
J2EE	Java 2 plataform, Enterprise Edition
J2ME	Java 2 Micro Edition

JDBC	Java Database Connectivity
JEE	Java platform, Enterprise Edition
JFC	Java Foundation Classes
JMS	Java Message Service
JNI	Java Native Interface
JPEG	Joint Photographic Experts Group
JSF	Java Server Faces
JSP	JavaServer Pages
LAA	Laboratório de Automação Agrícola
MADS	Modeling of Application Data with Spatial-temporal features
MOSAICo	Modelo de Objetos para Sistemas Abertos de Informação de Campo
MVC	Model-View-Controller
OCL	Object Constraint Language
OGC	Open Geospatial Consortium
OLE/COM	Object Linking and Embedding/Component Object Model
OLTP	On-Line Analytical Processing
OMT	Object Modeling Technique
OMT-G	Object Modeling Technique for Geographic Applications
OQL	Object Query Language
PAML	Precision Agriculture Markup Language
PVL	Plug-in for Visual Languages
RMI-IIOP	Remote Method Invocation – Internet Inter-ORB Protocol
SAAM	Software Architecture Analysis Method
SAD-69	South American Datum 1969
SDI	Spatial Data Infrastructure
SGBD-OR	Sistemas de Gerenciamento de Bancos de Dados Objeto-Relacionais
SGDB	Sistemas de Gerenciamento de Bancos de Fados
SIG	Sistemas de Informações Geográficas
SQL	Structured Query Language
SVG	Scalable Vector Graphics
TIFF	Tagged Image File Format

TUML	Temporal Unified Modeling Language
UML	Unified Modeling Language
URL	Uniform Resource Locator
USP	Universidade de São Paulo
UTM	Universal Transversa de Mercator
W3C	World Wide Web Consortium
WCS	Web Coverage Service
WCTS	Web Coordinate Transformation Service
WFS	Web Feature Service
WFS-G	Web Gazetter Service
WMS	Web Map Service
WTS	Web Terrain Service
XML	eXtensible Markup Language
XSL	eXtensible Stylesheet Language
XSLT	XSL Transformations

SUMÁRIO

1	INTRODUÇÃO	20
1.1	MOTIVAÇÃO	20
1.2	OBJETIVOS	23
1.3	ORGANIZAÇÃO DO TRABALHO	24
2	CONCEITOS DE AGRICULTURA DE PRECISÃO E ANÁLISE ESPACIAL	26
2.1	INTRODUÇÃO.....	26
2.2	AGRICULTURA DE PRECISÃO.....	27
2.3	O MOSAICO	32
2.4	ALGUNS CONCEITOS BÁSICOS DE CARTOGRAFIA	36
2.5	SISTEMA DE COORDENADAS.....	39
2.6	GLOBAL NAVIGATION SATELLITE SYSTEMS (GNSS).....	41
2.7	TIPOS DE DADOS GEOGRÁFICOS	42
2.7.1	<i>Representação Geográfica</i>	<i>43</i>
2.7.1.1	<i>Representação Matricial e Vetorial.....</i>	<i>44</i>
2.7.2	<i>Dados temporais.....</i>	<i>45</i>
2.8	CONSIDERAÇÕES FINAIS	49
3	MODELAGEM E CONSTRUÇÃO DE SISTEMAS ESPAÇO-TEMPORAIS.....	51
3.1	INTRODUÇÃO.....	51
3.2	REQUISITOS PARA MODELAGEM ESPAÇO-TEMPORAL	53
3.3	MODELOS BASEADOS NO FORMALISMO ENTIDADE E RELACIONAMENTO	54
3.3.1	<i>GISER.....</i>	<i>54</i>
3.3.2	<i>Geo-ER.....</i>	<i>56</i>
3.3.3	<i>MODUL-R.....</i>	<i>60</i>
3.4	MODELOS BASEADOS NO FORMALISMO ORIENTADO A OBJETOS.....	61
3.4.1	<i>MADS</i>	<i>61</i>
3.4.2	<i>PVL e UML.....</i>	<i>63</i>
3.4.3	<i>GeoOOA.....</i>	<i>66</i>
3.4.4	<i>GMOD.....</i>	<i>68</i>
3.4.5	<i>OMT-G.....</i>	<i>69</i>
3.4.6	<i>Temporal UML</i>	<i>70</i>

3.4.7	<i>Framework GeoFrame – UML</i>	72
3.5	A BIBLIOTECA TERRALIB	76
3.6	DEEGREE: SOFTWARE LIVRE PARA CONSTRUÇÃO DE SDI.....	79
3.7	ANÁLISE E PROJETO	82
3.8	CONSIDERAÇÕES FINAIS	87
4	PLATAFORMA JAVA ENTERPRISE EDITION (JEE)	90
4.1	INTRODUÇÃO	90
4.2	CONCEITOS BÁSICOS.....	90
4.3	CLIENTES JEE	92
4.4	COMPONENTES WEB	92
4.5	COMPONENTES ENTERPRISE JAVABEANS (EJB)	93
4.6	ARQUITETURA DE APLICAÇÕES JEE.....	94
4.6.1	<i>Aplicações Multicamadas</i>	94
4.6.2	<i>Cliente Stand-alone</i>	95
4.6.3	<i>Aplicações centradas na Web</i>	96
4.6.4	<i>Aplicações Business-to-Business</i>	97
4.7	CONSIDERAÇÕES FINAIS	97
5	UMA PROPOSTA DE ARQUITETURA DE SISTEMAS ESPAÇO-TEMPORAIS PARA AGRICULTURA DE PRECISÃO	99
5.1	INTRODUÇÃO	99
5.2	PROPOSTAS DE CENÁRIOS ARQUITETURAIS.....	101
5.2.1	<i>Arquitetura de Referência</i>	101
5.2.2	<i>Cenário Centrado na Web</i>	103
5.2.3	<i>Cenário Multicamadas</i>	105
5.2.4	<i>Cenário com Cliente Stand-alone</i>	107
5.2.5	<i>Avaliação dos Cenários Arquiteturais</i>	109
5.3	PADRÕES DE ANÁLISE E PROJETO	112
5.3.1	<i>Extensão da UML</i>	113
5.3.2	<i>O Catálogo de Padrões de Análise e Padrões de Projeto</i>	115
5.3.2.1	<i>Padrão de Análise: Talhão e Amostras</i>	117
5.3.2.2	<i>Padrão de Análise: Mapas de Talhões</i>	118
5.3.2.3	<i>Padrão de Projeto: Apresentar Objetos Espaciais</i>	120

5.3.2.4	Padrão de Projeto: Atributos Temporais	122
5.3.2.5	Padrão de Projeto: Relacionamentos Temporais	124
5.3.2.6	Padrão de Projeto: Objetos de Negócio Espaço-Temporais	127
5.3.2.7	Padrão de Projeto: Persistir Objetos Espaço-Temporais	129
5.3.2.8	Padrão de Projeto: Consultar Objetos Espaço-Temporais	132
5.3.2.9	Padrão de Projeto: Montar Objetos Espaço-Temporais Complexos ..	133
5.4	ESPECIFICAÇÃO DAS CAMADAS LÓGICAS	136
5.4.1	<i>Camada de Apresentação</i>	137
5.4.2	<i>Camada de Negócio</i>	138
5.4.3	<i>Camada de Integração e Persistência</i>	145
5.4.3.1	Mapeamento Objeto-Relacional	147
5.4.3.2	Processador de Consultas	150
5.4.3.3	Interoperabilidade com GML	153
5.5	CONSIDERAÇÕES FINAIS	155
6	ESTUDO DE CASO	157
6.1	INTRODUÇÃO	157
6.2	CONVERSÃO DO PAML PARA O SVG	157
6.2.1	<i>Scalable Vector Graphics (SVG)</i>	158
6.2.1.1	O Formato SVG	159
6.2.1.2	Formas Básicas, Caminhos e Texto	160
6.2.2	<i>Transformações PAML para SVG</i>	162
6.3	IMPLEMENTAÇÃO DO COMPONENTE VIEWTALHAO	163
6.3.1	<i>Infraestrutura de Software</i>	165
6.4	RESULTADOS ALCANÇADOS	166
6.5	CONSIDERAÇÕES FINAIS	168
7	CONSIDERAÇÕES FINAIS	170
7.1	CONCLUSÕES	170
7.2	PRINCIPAIS CONTRIBUIÇÕES	172
7.3	TRABALHOS FUTUROS	174
8	REFERÊNCIAS	177

1 INTRODUÇÃO

1.1 Motivação

Sistemas de Informação Geográfica (SIG) são sistemas computacionais capazes de capturar, armazenar, consultar, manipular, analisar e imprimir dados referenciados espacialmente em relação à superfície da Terra (ASSAD; SANO, 1998). Entretanto, não se pode esquecer de outras variáveis que vão além do geoposicionamento dos fenômenos geográficos, tais como a temporalidade dos dados.

Segundo Silva (1999), fenômenos relacionados ao mundo real podem ser descritos de três maneiras: espacial, temporal e temática. Espacial quando a variação depende de lugar para lugar (declividade, altitude, profundidade do solo); temporal quando a variação depende do tempo (densidade demográfica, profundidade do solo) e temática quando as variações são detectadas através de mudanças de características (geologia, cobertura vegetal). Estas três maneiras de se observar os fenômenos que ocorrem na superfície da Terra são, coletivamente, denominadas dados espaciais.

Desse modo, em sistemas de informações voltados para Agricultura de Precisão (AP), pode-se naturalmente identificar a existência de dados cuja descrição requer esses três enfoques: espacial, temático e temporal, tais como posicionamento de talhões e amostras de cultura ou pragas e variabilidades climáticas, tornando evidente a utilidade de SIG em sistemas ligados ao gerenciamento de culturas agrícolas. Segundo Saraiva (1998), é cada vez maior o consenso que não basta se basear apenas no resultado final do processo de produção para aferir e validar a adequação da estratégia de manejo e precisão adotada. Percebe-se então, que a geração de informações para tomada de decisões no processo produtivo pode ser feita através de dados do passado, presente e até mesmo previsões futuras.

Uma das etapas do processo de sistemas voltados para AP é realizar o processamento dos dados coletados em campo, com o objetivo de obter informações úteis para apoiar a tomada de decisões e a elaboração de estratégias

de gerenciamento agrícola que levem em conta a variabilidade encontrada no campo. Se por um lado, esses sistemas auxiliam na otimização do uso dos recursos agrícolas com objetivo de lucro, a redução dos impactos ambientais da produção agrícola também é esperada, certamente pela otimização da aplicação de insumos agrícolas nocivos ao meio ambiente, tais como pesticidas.

É possível encontrar muitos trabalhos de pesquisa que objetivam construir sistemas de informações ou então sistemas para apoio à construção de sistemas para AP. Murakami (2006) realizou um levantamento sobre alguns desses sistemas. No âmbito acadêmico, o autor concluiu que os sistemas, em sua maioria, são pouco abrangentes e que são projetados para problemas específicos da AP. No âmbito comercial, os sistemas são mais abrangentes e permitem o tratamento de dados gerados na etapa de coleta de dados da AP. Entretanto, conclui que esses sistemas não atendem plenamente as necessidades dos agricultores em termos de funcionalidades, custos e capacidade de integração com outros sistemas.

Percebe-se, por exemplo, que em estudos relacionados a temas específicos da AP, tal como estudo de variabilidades na produção, as pesquisas utilizam várias soluções para fazer parte das análises necessárias. Por exemplo, pode-se encontrar na literatura trabalhos que procuram estudar as variabilidades existentes em uma determinada cultura para auxiliar no processo de tomada de decisões. Mascarin e Molin (2006) realizaram um experimento onde constataram a variabilidade espacial da taxa de produtividade de um pomar de laranjas. Nesse experimento, utilizaram o software GS+[®] (*Geostatistical for Environmental Sciences*) para realizar a análise geoestatística e o semivariograma para o conjunto de dados e o SIG SSToolbox[®] para obter o mapa de produtividade. Miguel *et al.* (2006) estudou a variabilidade espacial e temporal de propriedades físico-hídricas de um solo para pastagens. Eles obtiveram as medições em três momentos distintos, de acordo com a quantidade de vezes que o rebanho havia passado pelo local. Nesse experimento, eles utilizaram o programa STAT para realizar a análise estatística dos dados. Contudo, os autores não detalharam como trataram a temporalidade dos dados.

Esses exemplos dão uma mostra de que pesquisadores precisam conviver com uma grande heterogeneidade de softwares para realizar seus experimentos, sem uma abordagem integrada para tal.

Outros trabalhos procuram desenvolver sistemas para auxiliar a tomada de decisões, tal como Silva *et al.* (2003) que propõem um sistema cliente-servidor

desenvolvido na linguagem Delphi que permite realizar a avaliação e preparação de projetos agro-industriais. Há sistemas que possuem um escopo bem definido, com objetivos bem específicos. Damjanovic-Zivic e Majdandzic (2001) apresentam um sistema integrado para planejamento e crescimento de uvas e gerenciamento do processo de produção de vinhos. Lassen *et al.* (2003) apresentam um sistema para apoio à decisão, baseado na plataforma Web. É um portal que disponibiliza informações regionais, relativas a dados climáticos, modelos de decisão para controle de doenças, incidência de pestes e ervas daninhas em cereais, etc. Na mesma linha, o sistema PlantelInfo (JENSEN *et al.*, 2000) também disponibiliza informações para tomada de decisões no campo e também é baseado na Web. No entanto, nenhum desses trabalhos lida com dados espaço-temporais e são muito específicos para determinadas aplicações do campo.

Miranda *et al.* (2002) propõem um *framework* para construção de aplicações que possibilita a publicação de informações geográficas na Internet e é desenvolvido na plataforma JEE. O *framework* fornece representação de dados geográficos baseados em GML (*Geographic Markup Language*), mas não implementa aspectos temporais. Também não possui implementação de funções geográficas e temporais.

Há também trabalhos de pesquisa voltados para modelagem de sistemas espaço-temporais, não especificamente para AP. Trata-se de trabalhos cujo principal objetivo é prover uma base notacional com semântica suficiente para modelar sistemas com aspectos espaciais, temporais ou ambos (BÉDARD *et al.*, 1996; HADZILACOS; TRYFONA, 1997; SHEKHAR *et al.*, 1997; BORGES; DAVIS; LAENDER, 2001; LISBOA FILHO, 2000; ROCHA, 2001). Esses trabalhos definem um formalismo para a modelagem de sistemas geográficos e nenhum deles, até por se tratarem de notações formais e genéricas, embutem os requisitos para AP e se restringem à modelagem conceitual de sistemas, deixando de lado aspectos físicos de implementação e tecnologias afins.

Nesse sentido, a biblioteca TerraLib (TERRALIB, 2006) é um importante mecanismo de implementação. Destina-se a servir como base para o desenvolvimento cooperativo na comunidade de usuários ou desenvolvedores de SIG e fornece funções para a decodificação de dados geográficos, estruturas de dados espaço-temporais, algoritmos de análise espacial, além de propor um modelo para um banco de dados geográficos (CÂMARA *et al.*, 2000). No entanto, também não carrega os requisitos de negócio da AP.

Conforme discutido, os trabalhos de pesquisa de softwares voltados para AP, além de não serem abrangentes, não definem aspectos técnicos relacionados a requisitos não funcionais que os sistemas podem apresentar. Não há mecanismos claros que definem o mapeamento do modelo conceitual da aplicação a uma solução física.

Esses mecanismos podem ser descritos na forma de padrões. São os chamados padrões de análise (FOWLER, 1997) e os padrões de projeto (GAMMA; VLISSIDES; JOHNSON, 1994). Os padrões de análise são um importante mecanismo para facilitar a reutilização de conhecimento e contribuem para o processo de desenvolvimento de duas maneiras diferentes. Primeiro, os padrões de análise diminuem o esforço de desenvolvimento de modelos abstratos que capturam os principais requisitos de um problema concreto, possibilitando modelos de análise reutilizáveis com exemplos e uma descrição de suas vantagens e limitações. Segundo, padrões de análise facilitam a transformação do modelo de análise para o modelo de projeto.

Por sua vez, padrões de projeto são soluções elegantes e reutilizáveis para problemas recorrentes que podemos encontrar no processo de desenvolvimento de aplicativos para o mundo real. Eles tratam da concepção e da interação entre objetos, definindo ao mesmo tempo um padrão de comunicação que é compartilhado por toda a equipe de desenvolvimento.

Esses mecanismos podem se tornar uma ferramenta poderosa se criados no contexto de uma arquitetura de software flexível e adaptável à diversidade de necessidades, traduzidas pelos requisitos não funcionais das aplicações.

Desse modo, a motivação principal para o desenvolvimento deste trabalho foi a possibilidade de desenvolvimento de mecanismos eficientes para a construção de sistemas para AP, que abrangessem o conhecimento do domínio da AP e o conhecimento técnico para desenvolvimento de sistemas baseados em padrões abertos e flexíveis.

1.2 Objetivos

O objetivo principal desse trabalho é propor uma arquitetura flexível e adaptável à diversidade de necessidades existentes para desenvolvimento de

software voltado para AP. Essa arquitetura deve ser compatível a uma arquitetura de referência, desenvolvida no âmbito do Laboratório de Automação Agrícola (LAA) (MURAKAMI, 2006; MURAKAMI *et al.*, 2006; MURAKAMI; RIBEIRO; SARAIVA, 2003; MURAKAMI *et al.*, 2002, 2003), e que define como modelo de referência o Modelo de Objetos para Sistemas Abertos de Informação de Campo (MOSAICo) (SARAIVA, 1998; SARAIVA; MASSOLA; CUGNASCA, 1998) e um estilo arquitetural em camadas.

Considerando a multiplicidade de aplicações, são propostos diversos cenários arquiteturais baseados em atributos de qualidade, tais como desempenho, escalabilidade e reusabilidade. Como parte da arquitetura de software proposta, este trabalho de pesquisa propõe um catálogo de padrões de análise e padrões de projeto que carregam em suas estruturas soluções aderentes à arquitetura de referência e o conhecimento agrônomo necessário. Esses padrões devem ser organizados por entre as camadas definidas para os cenários arquiteturais.

Pretende-se que a arquitetura de software, aliada aos padrões definidos, permita a construção de componentes de software destinados ao desenvolvimento de aplicações para AP. Tais componentes devem possibilitar a prática intensiva de reutilização, tanto de código como de conhecimento. Com objetivo de validar essa proposta, foi realizada a implementação de um padrão de projeto segundo um cenário arquitetural definido. Por fim, o trabalho foi desenvolvido com adoção de padrões abertos para desenvolvimento baseado na Web.

1.3 Organização do Trabalho

O Capítulo 2 apresenta um levantamento de alguns conceitos básicos sobre AP e de análises espaciais que podem estar envolvidas no processo da AP. Esse capítulo procura mostrar como o MOSAICo atende aos requisitos da AP e o tipo de informação, espacial e temporal, que está envolvida no processo de aquisição, processamento e análise dos dados da AP.

O Capítulo 3 apresenta um estudo sobre modelagem e construção de sistemas espaço-temporais. Esse estudo mostra algumas iniciativas de pesquisa no sentido de se prover uma base notacional e semântica genérica o bastante para se modelar sistemas geográficos. Ainda, o estudo mostra dois softwares para apoio à

construção de sistemas geográficos e alguns conceitos de análise e projeto que serão úteis para o entendimento dos próximos capítulos.

O Capítulo 4 apresenta alguns conceitos sobre a tecnologia *Java platform Enterprise Edition* (JEE). O objetivo é apresentar alguns conceitos úteis para a compreensão da solução arquitetural proposta neste trabalho de pesquisa.

O Capítulo 5 apresenta a proposta da solução arquitetural de tecnologia aberta e flexível. Baseada na plataforma JEE, a arquitetura é composta por três cenários arquiteturais, todos baseados em camadas lógicas (*layers*) e por um catálogo de padrões de análise e padrões de projeto, classificados de acordo com a camada lógica a que pertencem. Esse capítulo apresenta também as especificações de cada camada.

O Capítulo 6 mostra a implementação de um estudo de caso para validar um padrão de projeto definido no capítulo anterior. As tecnologias envolvidas e os resultados obtidos com esse estudo de caso serão discutidos nesse capítulo.

O Capítulo 7 apresenta as considerações finais: conclusões, contribuições e trabalhos futuros.

Por fim, o Capítulo 8 apresenta as referências bibliográficas utilizadas neste trabalho de tese.

2 CONCEITOS DE AGRICULTURA DE PRECISÃO E ANÁLISE ESPACIAL

2.1 Introdução

A Agricultura de Precisão (AP), que tem como objetivo permitir o tratamento das diferenças existentes dentro de um talhão, agregando o conhecimento acumulado pelas ciências agrárias, necessita de sistemas de informações que possibilitem a análise dos dados coletados e mais tarde gerem mapas para aplicações de insumos. Nesse contexto, informações geográficas sobre os pontos coletados e analisados são de grande importância, considerando que a análise dos dados envolve variabilidade espacial, por exemplo de atributos do solo, e os mapas de aplicação de insumos devem apresentar informações geo-posicionadas para aplicação em taxa variável.

Estabelecer correlações de causa e efeito exige que as informações de campo, tais como atributos de solo ou dados de produtividade, estejam integradas com as informações geográficas e temporais. Somente com essa integração é possível analisar se em determinado local há uma relação entre produtividade e aplicação de um determinado insumo, ao longo de alguns anos de manejo.

Estabelecida a relação entre informações geográficas e os requisitos para a AP, é necessário prover um modelo que represente os dois componentes de maneira integrada. Saraiva (1998) e Saraiva, Massola e Cugnasca (1998) elaboraram um modelo de objetos conceitual, o MOSAICo, capturando esses requisitos. Esse modelo representa um amplo levantamento de necessidades cujo objetivo é subsidiar a construção de aplicações computacionais, servindo também como um protocolo de comunicação entre os usuários e desenvolvedores de sistemas, promovendo o entendimento comum do tema em questão.

Os objetivos deste capítulo são: primeiro apresentar alguns conceitos da AP e os principais requisitos levantados em Saraiva (1998) e como foram modelados no MOSAICo. Depois, apresentar alguns temas relacionados à análise espacial, tais como conceitos básicos de cartografia e de informações espaciais e temporais.

2.2 Agricultura de Precisão

Na agricultura convencional, os talhões são tratados de maneira uniforme, sem preocupações com certas variabilidades que o talhão pode apresentar em sua área. As amostras obtidas do solo são homogeneizadas, sendo misturadas e tratadas como uma amostra única composta. Assim, todos os valores obtidos para as variáveis do solo são representantes da média de todo o talhão, que será base para determinar a taxa de aplicação dos insumos. A consequência disso é que essa taxa deverá ser constante para todo o talhão.

Na AP, entretanto, a variabilidade da produtividade é considerada dentro da área de um mesmo talhão. A produtividade não é homogênea em toda a área de um mesmo talhão, mesmo recebendo um tratamento homogêneo. Dessa maneira, a aplicação de insumos pode ser diferenciada de acordo com a produtividade de cada área do talhão, objetivando otimizar a utilização de insumos como agro-químicos, fertilizantes e sementes. Aspectos ambientais também são fatores motivadores para o uso dessa filosofia de manejo, uma vez que a otimização do uso de insumos pode minimizar a degradação do meio-ambiente.

Segundo Saraiva (2003), o processo da AP é dividido em quatro etapas, conforme ilustrado na Figura 2.1.

Figura 2.1. Etapas da Agricultura de Precisão

Fonte: baseado em (Saraiva, 2003)

O objetivo da etapa de coleta de dados é obter a variabilidade da produção e demais fatores, tais como parâmetros do solo, presença de pragas e ervas daninhas. Esses dados devem ser geo-posicionados, de maneira que a variabilidade espacial dentro de um talhão seja evidenciada.

A segunda etapa, gerenciamento de informação, consiste nas operações:

- efetuar correções nos dados coletados em campo, relativas aos erros dos métodos e equipamentos de coleta;
- processar os dados coletados para inicialmente avaliar e quantificar a variabilidade espacial;
- tentar relacionar a variabilidade da produção (colheita) com a dos fatores que influenciam na produção (solo, insumos, pragas, etc), buscando relações de causa-efeito;
- a partir dessas relações obtidas, propor estratégias para o gerenciamento agrícola, que são expressas através de mapas de aplicação de insumos em taxa variável em campo, para atingir as metas desejadas de gerenciamento: metas de produtividade, redução do uso de insumos ou mesmo de redução do impacto ambiental.

Na terceira etapa, é feita a aplicação de insumos agrícolas em taxas diferenciadas. Os mapas resultantes da etapa anterior devem mostrar a variação espacial de variáveis da área trabalhada, como: fertilidade do solo, incidência de ervas daninhas, incidência de pragas ou de doenças, umidade, acidez, estrutura e drenagem do solo, etc. De posse desses mapas o produtor poderá usar o computador e manipulá-los de maneira que possa ser criado um mapa geral de prescrição técnica para área. A avaliação permeia todo o processo e compreende a análise e adequação das estratégias com objetivo de refinar o processo, realimentando o sistema.

Segundo Saraiva (1998, p.31):

A etapa de gerenciamento de informação é a mais crítica, uma vez que já se dispõe de equipamentos para a coleta de dados e equipamentos para aplicação localizada, mas falta justamente o conhecimento, a técnica para transformar o conhecimento da variabilidade das entradas em estratégias efetivas de gerenciamento.

Ainda, as iniciativas documentadas sobre a manipulação de dados coletados seguem diferentes linhas, e concentram-se em aspectos particulares do problema de gerenciamento, envolvendo abordagens parciais da AP, tais como:

- identificação de erros de coleta e o desenvolvimento de algoritmos de correção;
- a análise da variabilidade e a seleção de um método adequado para essa análise;

- a desigual distribuição dos pontos amostrados ao longo do talhão e as formas (métodos) de atribuir valores a cada ponto, por exemplo estimando valores para locais não amostrados;
- a identificação de correlações que expliquem a variabilidade da produtividade, e a seleção dos parâmetros de campo cujos mapas serão conjuntamente analisados;
- as estratégias de gerenciamento a adotar a partir da variabilidade constatada (otimizar a lucratividade, aumentar a produtividade, restringir a perda de insumos para o ambiente);
- o tipo de unidade de manejo a adotar;
- o desenvolvimento de ferramentas computacionais de apoio à manipulação dos dados no gerenciamento.

Enfim, essa etapa consiste em interpretar as informações dos fatores de produção e correlacioná-las com a variabilidade constatada nos valores de produtividade das culturas agrícolas, de tal forma que se chegue ao correto diagnóstico das causas dessa variabilidade.

Dentro desse contexto, um conceito de grande importância é o de variabilidade espacial. Lamparelli, Rocha e Borghi (2001) afirmam que para a otimização de um processo de produção, necessita-se estudar a variabilidade de algumas características que sejam conectadas ao processo, para que seja possível coletar o máximo de informação e se possa obter relativo controle sobre o mesmo. Segundo Saraiva (1998), a coleta de dados é o ponto de partida para a identificação da variabilidade na produção. Para tanto, é necessário coletar os valores das características da produção (atributos ou parâmetros) atreladas às suas posições geográficas (coordenadas espaciais), para que se consiga inferir sobre as estruturas de variabilidade existentes. Em geral, esta inferência é feita utilizando-se técnicas estatísticas.

Segundo Lamparelli, Rocha e Borghi (2001), a aplicação da Estatística para interpretar e explorar características de uma população da qual não se tem acesso a todos os dados é válida somente se a técnica de amostragem, o tamanho da amostra e as estatísticas (funções capazes de caracterizar a distribuição do produto a ser analisado, como por exemplo, média, variância) forem adequadas ao tipo e proveniência dos dados. Isso porque a base das técnicas estatísticas que oferecem

consistência e segurança aos resultados é inteiramente realizada sobre suposições que devem ser condizentes com os problemas específicos.

O tamanho da amostra é um fator determinante para a coleta de dados geográficos. Há uma estrutura de variabilidade a ser modelada e considerada para eventuais técnicas a serem aplicadas posteriormente, tornando crítica a distribuição das amostras coletadas em uma região. O número de amostras deve ser suficientemente grande para permitir caracterizar a estrutura de variabilidade.

Cada fenômeno da natureza, dependendo da localização, apresenta uma escala de variabilidade diferente. O tamanho da amostra, assim como a frequência em que são coletadas essas amostras (número de amostras) e o esquema de amostragem devem ser adequados à estrutura de variabilidade do fenômeno em questão.

Existem diversos esquemas amostrais para serem considerados. Alguns dos principais e mais utilizados esquemas de amostragem são mostrados na Figura 2.2.

Figura 2.2. (a) Amostragem Regular, (b) Amostragem Aleatória e (c) Amostragem Aleatória Estratificada

Fonte: (LAMPARELLI; ROCHA; BORGHI, 2001)

Regular: A amostragem regular é um esquema em que se coletam amostras em pontos igualmente espaçados e homogeneamente distribuídos por toda a região considerada. É facilmente delineado e de imediata localização, uma vez que o primeiro ponto é localizado. Esse esquema é muito usado, mas cuidados devem ser tomados quando a região possui algum padrão que possa introduzir um viés às observações nas mesmas direções que a grade.

Aleatória: Nesse esquema, os pontos são fixados aleatoriamente por toda a região, e não há problemas de viés, já que o próprio princípio de aleatorização anula qualquer efeito de padrões existentes. Por outro lado, quando o número de amostras é pequeno, elas podem não ser homogeneamente distribuídas, deixando parte da

região sem a exploração de suas propriedades. Outra desvantagem desse esquema é a dificuldade de localização dos pontos a serem amostrados, mesmo utilizando um GPS.

Aleatória estratificada: Esse esquema é, na maioria das vezes, preferido pois tem parte das vantagens do aleatório e do regular. Os pontos são alocados dentro de uma célula da grade (considerado estrato). Com isso, cobre-se parte do problema do viés nas estimativas, ainda mantendo uma distribuição homogênea na área.

Em todos os casos, o número de amostras e o espaçamento entre elas devem ser suficientes para se determinar a estrutura de variabilidade e para que se consiga detectar alguma estrutura de correlação entre as amostras, e obter uma boa estimativa do modelo para explicar o padrão de variabilidade de cada parâmetro ou atributo considerado. Um dos problemas que pode ocorrer, principalmente na primeira exploração da área, considerando um atributo qualquer, é delimitar um espaçamento muito grande entre as amostras, ocasionado principalmente pelo alto custo da amostragem. Daí a necessidade de dados históricos (temporais) para auxiliar em estudos preliminares, diminuindo o risco de ter que se refinar a grade de amostragem e repetir o experimento.

Considerando as várias necessidades e aplicações apresentadas, a utilização de sistemas de informação para armazenamento e processamento dos dados coletados em campo, de maneira integrada, é de suma importância para o sucesso da AP e particularmente sistemas que sejam capazes de representar a variabilidade espacial e temporal dos dados coletados e que forneçam mecanismos para manipulação dessa informação.

Saraiva (1998) realizou um amplo levantamento dos requisitos de software para atender aos objetivos da AP. Esses requisitos foram capturados em um modelo de objetos, o MOSAICo. Trata-se de um modelo conceitual baseado no formalismo da Orientação a Objetos, que não tem como objetivo detalhar um sistema em particular, mas uma classe de sistemas, que orientam o desenvolvimento de aplicações específicas para suporte às atividades de AP. É, portanto, um modelo conceitual em um alto nível de abstração e, propositadamente, sem preocupação com aspectos de implementação. A próxima seção apresenta um resumo do modelo.

2.3 O MOSAICo

O MOSAICo (SARAIVA, 1998; SARAIVA; MASSOLA; CUGNASCA, 1998) captura os principais requisitos do domínio AP, representando as funcionalidades e informações necessárias para o armazenamento dos dados coletados em campo, o processamento desses dados e a geração da informação necessária para a aplicação localizada dos insumos. Os principais requisitos modelados são:

- Manter informações sobre propriedades, culturas e variabilidade espacial e temporal, sendo que os dados de campo de entrada podem possuir diferentes fontes e formatos.
- Manter informações sobre insumos e equipamentos que serão utilizados nas operações que o sistema suporta.
- Permitir análise dos diversos cenários de gerenciamento para possibilitar apoio à tomada de decisão. Esses cenários devem incluir modelos de crescimento, de produção da cultura, modelos climatológicos e modelos mercadológicos de preço de produtos.
- Permitir a correção para a distribuição espacial não uniforme dos dados amostrados, por meio de técnicas estatísticas tal como interpolação.
- Manter as receitas de tarefas, que são o conhecimento agrônomo sobre métodos adequados à realização de determinadas tarefas no sistema. Uma receita, por exemplo, poderia incluir a sequência de operações necessárias para se obter mapas de zonas de manejo uniforme.
- O sistema deve armazenar informações e conhecimento agrônomo que serão utilizados no processo de tomada de decisão. Tal conhecimento é específico de cada cultura e envolve conhecimento sobre cada uma das várias operações de campo, como reparo de solo, plantio, adubação, irrigação para aquela cultura.
- Produzir recomendações de manejo, tais como mapas de aplicação de insumos.
- Manter histórico de operações realizadas no sistema

A abordagem para representar os requisitos do MOSAICo foi o modelo de Casos de Uso. Segundo Booch, Rumbaugh e Jacobson (2005), o modelo de casos

de uso deve capturar todo o comportamento de um sistema e a interação do sistema com o meio externo, que podem ser atores humanos ou autômatos (outros sistemas de informações ou equipamentos). O modelo de Casos de Uso do MOSAICo é composto de três grandes pacotes. A Figura 2.3 apresenta os pacotes de Casos de Uso do MOSAICo.

Figura 2.3. Pacotes de Casos de Uso do MOSAICo

Fonte: baseado em (SARAIVA, 1998)

O pacote Configurar o Sistema contém os casos de uso responsáveis por editar a base de conhecimento, composta de regras de manejo, os dados de insumos e equipamentos de mercado, as receitas de tarefas e os atalhos de automação do sistema. O pacote Inserir Dados de Campo compreende os casos de uso cujas responsabilidades são: o registro de novos proprietários e propriedades, talhões e a entrada de dados de campo amostrados. Por último, o pacote Gerar Mapas consiste de casos de uso para o operador trabalhar sobre os dados disponíveis no sistema para um determinado talhão, efetuando correção de dados coletados e associando esses dados a mapas no sistema. O objetivo final desse pacote é gerar o mapa de recomendação de aplicação para cada insumo e um arquivo para um determinado equipamento de aplicação. A Figura 2.4 ilustra o modelo de casos de uso do pacote Inserir Dados de Campo.

Figura 2.4. Modelo de casos de uso do pacote “Inserir Dados de Campo”

Baseado no modelo de casos de uso, o modelo estático foi gerado. Segundo Booch, Rumbaugh e Jacobson (2005), o modelo estático captura a estrutura do sistema, representada por objetos, atributos e relacionamentos. Esses conceitos são agrupados em classes, que capturam o vocabulário do sistema e são abstrações que representam parte do domínio do problema. A Figura 2.5 mostra o diagrama de pacotes com os subsistemas do MOSAICo e logo após, é apresentado um resumo de cada subsistema.

Figura 2.5. Diagrama de pacotes com os subsistemas do MOSAICo

Fonte: (MURAKAMI et al., 2002)

DadosAgricultivos: compreende a manipulação das classes de objetos relacionados com informações sobre insumos, máquinas e equipamentos de AP.

InformacoesGeograficas: compreende a manipulação das classes relacionadas a informações sobre mapas de diversos tipos, tais como mapas de aplicação de insumos.

ModelosSimulacao: compreende as classes de objetos relacionados a simulações visando obter mapas simulados.

AssistenteTarefas: compreende as classes relacionadas com informações sobre receitas para a realização de determinadas tarefas no sistema, bem como automação dessas receitas.

Especialista: compreende a manipulação das classes relacionadas com conhecimento agrônomo, como as regras de manejo para as diversas operações de aplicação de insumos.

DadosCampo: compreende a manipulação das informações de campos escalares, e também das georreferenciadas não transformadas em mapas como, por exemplo, dados amostrados por coletores de dados com receptor GPS. Esse subsistema é responsável por informações básicas sobre propriedades e proprietários e informações dos limites dos talhões e equipamentos de posicionamento e de coleta de dados.

O MOSAICo, como mostrado na Figura 2.5, foi subdividido em módulos, e cada módulo pode ser considerado um sub-sistema afim. A Figura 2.6 apresenta o diagrama de classes completo do subsistema Dados de Campo.

Figura 2.6. Subsistema Dados de Campo

Os dados coletados em campo vão compor as amostras, que podem ser de vários tipos, como amostras de cultura, amostras de solo, amostras de praga, entre outras e são obtidos de equipamentos de coleta de dados, como monitores de colheita, ou de imagens e fotos digitalizadas, por exemplo. Deve ser possível também a entrada de dados manuais, tanto de variáveis espacialmente referenciadas, como de variáveis escalares.

2.4 Alguns Conceitos Básicos de Cartografia

Uma aplicação para AP requer essencialmente que algumas informações estejam localizadas precisamente, o que só é possível com um sistema de coordenadas para o local. No caso do MOSAICo, tipicamente os subsistemas de Dados de Campo e Informações Geográficas são exemplos de informações espaciais. Nesse contexto é inserida a Cartografia, que tem como objetivo encontrar uma função matemática que permita estabelecer a relação de localização entre uma

característica na superfície terrestre e a mesma no plano, através de um mapa (LAMPARELLI; ROCHA; BORGHI, 2001).

O modelo que melhor estabelece essa relação é o Geóide, que representa a Terra com grande fidelidade. Porém, a complexidade matemática impede que esse modelo seja utilizado para fins mais práticos (D'ALGE, 2001). Desse modo, a cartografia utiliza uma outra forma de representação, o elipsóide, mais fácil de representar matematicamente.

O elipsóide é então transformado em um plano, embora com alguma distorção, para ser mapeada. A distorção ocorre pois não é possível passar de uma superfície curva para uma superfície plana sem essas deformações. Para realizar a transformação, são utilizadas projeções cilíndricas, cônicas e planas, conforme apresentado na Figura 2.7.

Figura 2.7. Projeções cartográficas
Fonte: (Frigoletto)¹

A partir de um elipsóide de referência, é definido o *datum*. Câmara *et al.*, (1996) definem o *datum* como um ponto onde a superfície do elipsóide de referência toca a Terra, sendo caracterizado a partir de uma superfície de referência (*datum* horizontal ou planimétrico) e de uma superfície de nível (*datum* vertical ou altimétrico).

O *datum* planimétrico é uma superfície de referência elipsoidal posicionada com respeito a uma certa região. Sobre esta superfície realizam-se as medições geodésicas que dão vida à rede geodésica planimétrica da região. Outro conceito

¹ Fonte: <http://frigoletto.com.br/Cartograf/projees.htm>

importante é o de *datum* vertical ou altimétrico. Trata-se da superfície de referência usada pelo geodesta para definir as altitudes de pontos da superfície terrestre. Na prática a determinação do *datum* vertical envolve um marégrafo ou uma rede de marégrafos para a medição do nível médio dos mares. Faz-se então um ajuste das medições realizadas para definição da referência “zero” e adota-se um dos marégrafos como ponto de referência do *datum* vertical. No Brasil o ponto de referência para o *datum* vertical é o marégrafo de Imbituba, em Santa Catarina.

Segundo Câmara *et al.* (1996), um dos problemas típicos na criação da base de dados de um SIG aqui no Brasil tem sido a coexistência de dois sistemas geodésicos de referência: Córrego Alegre e SAD-69. Algumas cartas topográficas referem-se ao Córrego Alegre, que é o antigo *datum* planimétrico brasileiro, enquanto outras utilizam como referência o SAD-69, que é o atual *datum* planimétrico. Os usuários de SIG já estão relativamente acostumados a conviver com escolhas de projeção e seleções de *datum* sempre que precisam realizar entrada ou importação de dados. Entretanto, as coordenadas geográficas - na verdade, geodésicas - são definidas sobre a superfície de referência do *datum* selecionado e que, portanto, variam de *datum* para *datum*.

Apesar de o plano possuir distorções em relação ao globo terrestre, pode apresentar a mesma forma de feições geométricas, denominadas projeções conforme; ou pode manter as dimensões da área, denominadas projeções equivalentes; e finalmente, pode conservar a relação entre as distâncias dos pontos representados e as distâncias entre esses mesmos pontos na superfície da Terra, neste caso denominadas projeções eqüidistantes. A escolha de um ou outro tipo de projeção deve levar em conta a que se destina o mapa. O Quadro 2.1 mostra alguns dos principais tipos de projeções para construções de mapas.

Projeção do mapa	Superfície auxiliar	Propriedade
Universal Transversa de Mercator	Cilindro	Conforme
Alberts cônica de área igual	Cone	Equivalente
Mercator	Cilindro	Conforme
Estereográfica Polar	Plana	Conforme

Quadro 2.1. Principais projeções cartográficas

Fonte: adaptado de (LAMPARELLI; ROCHA; BORGHI, 2001)

2.5 Sistema de Coordenadas

Segundo Câmara *et al.* (1996), um objeto geográfico qualquer somente pode ser localizado se puder ser descrito em relação a outros objetos cujas posições sejam previamente conhecidas, ou se tiver sua localização determinada em uma rede coerente de coordenadas. Quando se dispõe de um sistema de coordenadas fixas, pode-se definir a localização de qualquer ponto na superfície terrestre.

D'Alge (2001) apresenta os seguintes sistemas: sistema de coordenadas geográficas, sistema geocêntrico terrestre, sistema de coordenadas planas ou cartesianas e sistema de coordenadas polares.

O sistema de coordenadas geográficas é o sistema mais antigo, onde cada ponto da superfície terrestre é localizado na interseção de um meridiano com um paralelo. Com o sistema de coordenadas geográficas, um ponto na superfície da Terra é representado por um valor de latitude / longitude. Latitude é a distância angular entre um ponto na superfície terrestre e a Linha do Equador, e longitude é a distância angular entre um ponto qualquer na superfície terrestre e o meridiano de origem (0°).

Ainda, segundo D'Alge (2001), o sistema geocêntrico terrestre é um sistema cartesiano tridimensional onde o centro do sistema está no centro do elipsóide adotado como modelo da Terra, o eixo Z coincide com o eixo de revolução do elipsóide, o eixo X está no plano do meridiano de Greenwich e o eixo Y está a 90 graus a leste do eixo X. Nesse sistema, as coordenadas de um ponto são definidas por um terno de valores X, Y e Z.

Já no sistema de coordenadas planas as coordenadas cartográficas estão baseadas em um sistema plano cartesiano e usa a distância a um ponto determinado como base para todas as localizações. Ele é baseado na escolha de dois eixos perpendiculares (eixos horizontal e vertical), cuja interseção, denominada origem, é estabelecida como base para a localização de qualquer ponto do plano. Nele, um ponto é representado por dois números: um correspondendo à projeção sobre o eixo x (horizontal) e o outro sobre o eixo y (vertical). Como as coordenadas planas são associadas matematicamente às coordenadas geográficas, umas podem ser convertidas nas outras.

Finalmente, o sistema de coordenadas polares tem como principal utilização o desenvolvimento das projeções cônicas. Trata-se de um sistema simples, de relação direta com o sistema de coordenadas cartesianas, que substitui o uso de um par de coordenadas (x,y) por uma direção e uma distância para posicionar cada ponto no plano de coordenadas.

No Brasil, muitos dados cartográficos utilizam a projeção Universal Transversa de Mercator (UTM) (1:250.000, 1:100.000, 1:50.000, 1:25.000) (D'ALGE, 2001), um sistema de coordenadas plana, que estabelece a divisão da Terra em 60 fusos de 6° cada. O início dos fusos está no antimeridiano de Greenwich (LAMPARELLI; ROCHA; BORGHI, 2001) e cresce em ordem sentido oeste-leste. Em relação à latitude, o UTM é limitado a 80° S e 84° N (Figura 2.8). Acima desses valores, a projeção adotada é a estereográfica polar.

Relacionam-se, a seguir, as principais características da projeção UTM:

- a superfície de projeção é um cilindro transversal e a projeção é conforme;
- o meridiano central da região de interesse, o equador e os meridianos situados a 90° do meridiano central são representados por retas;
- os outros meridianos e os paralelos são curvas complexas;
- a escala aumenta com a distância em relação ao meridiano central, tornando-se infinita a 90° do meridiano central;
- como a Terra é dividida em 60 fusos de 6° de longitude, o cilindro transversal adotado como superfície de projeção assume 60 posições diferentes, já que seu eixo mantém-se sempre perpendicular ao meridiano central de cada fuso;
- aplica-se ao meridiano central de cada fuso um fator de redução de escala igual a 0,9996, para minimizar as variações de escala dentro do fuso;
- duas linhas aproximadamente retas, uma a leste e outra a oeste, distante cerca de 1°37' do meridiano central, são representadas em verdadeira grandeza.

Figura 2.8. Sistema de coordenadas UTM

FONTE: (Esteio On-Line 006²)

2.6 Global Navigation Satellite Systems (GNSS)

O sistema GNSS engloba os sistemas de posicionamento GPS americano, o GLONASS da Rússia, o GALILEO da União Européia e o Beidou chinês. O posicionamento geográfico das informações, segundo um sistema de coordenadas, é de fundamental importância para a AP. A localização das amostras dentro de um talhão, por exemplo, é um tipo de informação geográfica necessária para as aplicações. Nesse contexto, o sistema americano GPS é largamente utilizado.

O sistema GPS foi criado pelo Departamento de Defesa dos Estados Unidos, na década de 70, e se tornou inteiramente operacional em 1993. É um sistema constituído de 24 satélites, sendo 3 de reserva. São satélites específicos para o sistema GPS e estão dispostos de tal maneira que, a qualquer hora e em qualquer lugar do mundo, pelo menos quatro satélites estarão disponíveis para utilização.

O princípio básico do sistema está na triangulação realizada através dos sinais enviados pelos satélites, via ondas de rádio. Dessa forma, utilizando os

² <http://www.esteio.com.br/newsletters/paginas/006/o-prutm.htm>

satélites como pontos de referência, é possível localizar qualquer ponto na Terra com coordenadas, por exemplo, UTM. A intersecção das distâncias a cada satélite, utilizadas no cálculo, define o ponto onde a antena do receptor GPS se encontra. O receptor GPS calcula as distâncias até os satélites.

2.7 Tipos de Dados Geográficos

Um tipo de informação vital para a AP é o espaço geográfico onde os eventos ocorrem. O espaço geográfico é o meio físico onde as entidades geográficas coexistem e uma entidade geográfica é qualquer entidade identificável do mundo real, possuindo características espaciais e relacionamentos espaciais com outras entidades geográficas.

Dados geográficos ou dados geo-referenciados são dados espaciais em que a dimensão espacial está associada à sua localização na superfície da Terra, num determinado instante ou período de tempo (ASSAD; SANO, 1998).

Os dados geográficos possuem duas características fundamentais: características espaciais e não-espaciais. As características espaciais informam a posição geográfica do fenômeno, sua geometria e o tempo de validade dos dados geográficos e suas variações sobre o tempo. As características não-espaciais descrevem o fenômeno. Enfim, a representação espacial de uma entidade geográfica é a descrição de sua forma geométrica associada à posição geográfica e suas variações temporais. Entretanto, devido à sua relevância, a temporalidade dos dados é geralmente tratada separadamente das propriedades geométricas.

Os dados geográficos possuem propriedades geométricas e topológicas. As propriedades geométricas são propriedades métricas. A partir de feições geométricas primitivas, tais como pontos, linhas e polígonos, os quais representam a geometria das entidades, são estabelecidos os relacionamentos métricos. Esses relacionamentos expressam a métrica das feições com referência a um sistema de coordenadas. De acordo com a geometria, são estabelecidas algumas propriedades geométricas, tais como, comprimento, sinuosidade e orientação para linha; perímetro e área da superfície para polígonos; volume para entidades tridimensionais e forma e inclinação tanto para linhas quanto para polígonos. Já as propriedades topológicas

(não-métricas) são baseadas nas posições relativas dos objetos no espaço, como conectividade, orientação (de, para), adjacência e contenção.

2.7.1 Representação Geográfica

Segundo Goodchild (1992), a realidade geográfica pode ser percebida segundo duas visões: a visão de campos e a visão de objetos. Na visão de campos (também chamada de modelo de campos) o mundo real é visto como uma superfície contínua sobre a qual entidades geográficas variam continuamente segundo diferentes distribuições. Cada fenômeno é visto como uma camada contínua, não existindo nenhuma posição no espaço geográfico que não esteja associada a algum valor correspondente à variável representada.

É comum a subdivisão dessa camada contínua em regiões de mesmo conteúdo, como por exemplo uma superfície de vegetação, onde cada área representa um determinado tipo de vegetação. Normalmente, os fenômenos naturais, físicos ou biológicos são representados por variáveis contínuas como, por exemplo, temperatura, pressão atmosférica e tipo de solo.

O Quadro 2.2 apresenta alguns exemplos de modelagem sobre essa visão.

Visão de Campos	Definição	Exemplos
Amostragem de pontos	O espaço é modelado como um conjunto de pontos onde cada ponto possui atributos que descrevem sua relação com outros pontos.	Modelos numéricos de terreno (pontos regularmente distribuídos), amostragem de solo.
Isolinhas	O espaço é modelado como um conjunto de linhas aninhadas onde cada linha possui um valor associado.	Curvas de nível, curvas de temperatura.
Polígonos	Subdivisão do espaço em polígonos adjacentes, onde cada posição pertence a um só polígono.	Tipos de solo, tipos de vegetação, culturas.

Quadro 2.2. Visão de Campos

Fonte: (BORGES, 1997)

Na visão de objetos (também chamada de modelo de objetos), a realidade é vista como uma superfície ocupada por entidades identificáveis e cada posição (x,y)

do espaço poderá estar ou não ocupada. Cada entidade possui uma determinada posição, geometria e características próprias. A modelagem de fenômenos geográficos na visão de objetos é um processo natural e direto (ex: rios são descritos pela classe ou entidade Rio).

2.7.1.1 Representação Matricial e Vetorial

As duas formas básicas de representação dos dados em um SIG são as formas *matricial* e *vetorial*. A representação em formato *matricial* (também chamada *raster*), o espaço é representado como uma matriz $P(m,n)$ composta de m colunas e n linhas (ASSAD; SANO, 1998). Cada célula possui um número de linha, um número de coluna e um valor correspondente ao atributo estudado, e cada célula é individualmente acessada pelas suas coordenadas.

A representação matricial supõe que o espaço pode ser tratado como uma superfície plana, onde cada célula é associada a uma porção do terreno. Um mesmo mapa pode ser representado por células de diferentes tamanhos (diferentes resoluções), representando diferentes áreas no terreno.

Segundo Borges (1997), as células podem ser de diferentes formatos: triangulares, hexagonais e retangulares (também chamadas de *pixels*). O termo *raster* designa células regulares. No entanto, é usado de forma genérica para representação matricial. Os relacionamentos topológicos no espaço são implicitamente determinados a partir da vizinhança das células e as coordenadas geográficas (longitude, latitude) ou planas (x,y) são obtidas indiretamente a partir da posição da célula na matriz (coluna, linha). Imagens de satélite e modelos digitais de terreno são naturalmente representados no formato matricial. A representação matricial corresponde a visão de campos definido por Goodchild (1992).

No caso de representações vetoriais, correspondentes à visão de objetos, consideram-se três elementos gráficos: ponto, linha e área (ou polígono) para representar a geometria das entidades geográficas. Pontos são representados por um par de coordenadas, linhas por uma seqüência de pontos e polígonos por uma seqüência de linhas onde as coordenadas do ponto inicial e final coincidem. Entidades geográficas lineares, como ruas, divisões político-administrativas ou redes de tráfego, são naturalmente representadas em formato vetorial. As redes são casos especiais de dados vetoriais, em que são utilizados arcos e nós conectados na

representação do fluxo e da direção da rede. As operações topológicas e métricas são comuns em representações vetoriais. Segundo Güting (1994), os relacionamentos espaciais são divididos nas seguintes categorias:

Relacionamentos topológicos: as relações topológicas são consideradas relações que descrevem os conceitos de vizinhança, incidência, sobreposição, mantendo-se invariáveis ante as transformações como escala rotação (por exemplo, dentro de, envolvido por, igual a).

Relacionamentos métricos: As relações métricas são consideradas em termos de distâncias e direções.

Relacionamentos direcionais: as relações direcionais descrevem a orientação no espaço como, por exemplo, norte e sul. As relações relativas à ordem total ou parcial dos objetos espaciais são descritas por preposições como em frente a, atrás, acima e abaixo.

A Figura 2.9 exemplifica as relações topológicas.

Figura 2.9. Relacionamentos Topológicos

2.7.2 Dados temporais

Segundo Dias, Câmara e Davis Jr. (2005), a maioria das aplicações de tecnologia de geoinformação utiliza representações estáticas de fenômenos espaciais, apesar de que muitos fenômenos espaciais são dinâmicos. Isso significa que um talhão, por exemplo, pode ter sua forma alterada no decorrer do tempo, e em um sistema estático, o formato anterior do talhão é ignorado, sendo perdido em detrimento da nova área do talhão. Ainda, informações não espaciais de um sistema voltado para AP também podem ser dinâmicas.

A representação temporal deve considerar três aspectos (EDELWEISS, 1998): ordem, variação e granularidade: quanto à ordem, o tempo pode ser linearmente ordenado, ramificado ou circular. Linearmente ordenado implica total ordenação entre dois pontos no tempo. A Figura 2.10 mostra o tempo linearmente ordenado.

Figura 2.10. Tempo linearmente ordenado

Fonte: (DIAS; CÂMARA; DAVIS JR., 2005)

O tempo ramificado implica na possibilidade de existência de diferentes histórias futuras ou passadas. A restrição linear é abandonada permitindo a possibilidade de dois pontos diferentes serem sucessores (ramificações no futuro) ou antecessores (ramificações no passado) imediatos de um mesmo ponto no tempo. Essa característica abre possibilidade para que um domínio possa ter múltiplos desenvolvimentos futuros (por ex. simulações de mapas de aplicações) (Figura 2.11-a) ou admite múltiplas histórias passadas do domínio em questão (Figura 2.11-b).

Figura 2.11. Tempo Ramificado

Fonte: (DIAS; CÂMARA; DAVIS JR., 2005)

Por sua vez, o tempo circular é utilizado para modelar eventos recorrentes. Neste caso, a periodicidade de sua ocorrência faz com que sempre se volte à mesma referência de tempo. Um exemplo é o ciclo anual de produção de uma determinada cultura.

Quanto à variação, o tempo pode ser contínuo ou discreto. O tempo é contínuo por natureza, mas para sua representação computacional é necessário utilizar uma representação discreta do tempo, na qual a variação temporal

corresponde a uma linha de tempo, composta por uma seqüência de intervalos de tempo consecutivos e com idêntica duração. Por fim, a granularidade temporal é o parâmetro que define a duração do intervalo temporal, tal como ano, mês, dia, hora e minuto.

Em um banco de dados, a dimensão temporal determina as representações de tempo. Um sistema atemporal descreve somente um estado. Isto significa que estados históricos são esquecidos e que a possibilidade de prever ou antecipar estados futuros não pode ser tratada. Por esta razão, fica impossibilitada a análise de processos que causam a troca de estados dos dados, tornando a dinâmica de modelar o mundo limitado quanto à análise, compreensão e previsão.

Por outro lado, um sistema temporal registra essa troca de estados permitindo o estudo dos diversos estados geográficos históricos e futuros. As seguintes questões podem ser abordadas em um sistema temporal:

Onde e quando as mudanças ocorreram?

Que tipos de mudanças ocorreram?

Qual a média das mudanças?

Qual a periodicidade das mudanças?

Além disso, através de dados temporais, um software poderia avaliar:

Se existem padrões temporais;

Que tendências são apresentadas pela informação?

Que processos antecedem as mudanças dos dados?

Tais avaliações podem formar a base de entendimento das causas de mudanças, conduzindo a uma melhor compreensão dos seus efeitos e do comportamento dos processos naturais.

Um sistema temporal deve tratar vários estados dos dados, assim como os instantes em que estes diferentes estados são válidos. Segundo Snodgrass e Ahn (1985) os dados temporais podem ser classificados em quatro tipos:

- dados instantâneos - neste tipo de sistema, os únicos valores perceptíveis são os valores presentes. Cada modificação no valor de uma propriedade pode ser percebida como uma transição do dado. Em uma transição o valor anteriormente armazenado é destruído e somente o último valor está disponível. O estado atual do sistema, composto pelos valores atuais de todas as propriedades, é o único existente.

- dados de tempo de transação: neste tipo de sistema, o valor definido é associado ao instante temporal em que foi realizada a transação, sob forma de um rótulo temporal. Uma relação que utilize esta abordagem pode ser vista como tendo três dimensões (tuplas, atributos, tempo de transação). Os sistemas de tempo de transação permitem a recuperação de informações definidas em algum instante do passado, pois todos os dados passados estão armazenados associados ao seu instante de definição (tempo de transação). Uma tupla passa a ser válida no momento em que é inserida na base de dados. Alterações retroativas não podem ser executadas, bem como erros em tuplas passadas não podem ser corrigidos.
- dados de tempo de validade: corresponde ao tempo em que uma informação é verdadeira no mundo real. A cada informação, é associado o tempo de validade sendo que este tempo deve ser fornecido pelo usuário. Este tipo de sistema registra a história relativa aos dados e não às transações, e permite a correção de erros através da modificação de dados. Os sistemas de tempo de validade admitem, no seu contexto, intervalos de tempo não válidos, e permitem a recuperação de informações válidas em momentos do passado, presente e futuro.
- dados bitemporais: neste tipo, o sistema combina as propriedades de dados de tempo de transação e de tempo de validade, ou seja, ele trata as duas dimensões de tempo. Toda a história dos dados é armazenada. É possível ter acesso a todos os estados passados do sistema, tanto o histórico das transações realizadas como o histórico da validade dos dados. O estado atual do banco de dados é constituído pelos valores atualmente válidos. Valores futuros podem ser definidos através do tempo de validade, sendo possível recuperar o momento em que estes valores foram definidos para eventuais alterações.

Em sistemas temporais, é muito importante considerar as operações de comparação e cálculo de duração entre dados temporais (relacionamentos temporais). Allen (1983 apud DIAS; CÂMARA; DAVIS Jr. 2005, p.146) definiu sete relações temporais: *before* (antes de), *meets* (encontra), *during* (durante), *finishes*

(finaliza com), *equal* (igual a), *overlaps* (sobrepõe) e *starts* (inicia com), A Figura 2.12 ilustra essas operações.

Figura 2.12. Operações Temporais

2.8 Considerações Finais

A grande quantidade de aspectos a serem potencialmente considerados evidencia a complexidade da etapa de gerenciamento de informação e a dificuldade de integrá-los numa abordagem única.

Em primeiro lugar, a modelagem dos aspectos espaciais e temporais, existentes no mundo real, para modelos computacionais não é uma tarefa trivial. A quantidade de pesquisas nessa área de conhecimento - como será apresentada na próxima seção - constata essa afirmação. Em segundo lugar, integrar esses modelos espaciais e temporais ao domínio da aplicação, no caso a AP, pode gerar sistemas monolíticos e complexos, de difícil compreensão e conseqüentemente, comprometer a manutenibilidade e reusabilidade do sistema. Essa tarefa exigirá técnicas de componentização e a definição de uma arquitetura adequada.

Os próximos capítulos abordam essas duas questões. A seguir, apresentamos um estudo de alguns modelos para informações espaciais, temporais

ou espaço-temporais. A maioria deles se concentra na definição conceitual do tema. Serão mostradas também algumas bibliotecas para implementação de SIGs e suas principais características.

3 MODELAGEM E CONSTRUÇÃO DE SISTEMAS ESPAÇO-TEMPORAIS

3.1 Introdução

A maioria dos modelos existentes de sistemas geo-referenciados é baseada nos formalismos de Entidade-Relacionamento e de Orientação a Objetos. No entanto, embora os modelos difiram muito com relação à notação gráfica (sintaxe), se propõem a um objetivo comum: a modelagem conceitual de aplicações geográficas.

A modelagem conceitual apresenta diversas vantagens para a modelagem de aplicações geográficas. Primeiro, por facilitar a execução do projeto lógico, o qual necessita atender às particularidades de um SIG específico. Os usuários podem expressar seus conhecimentos sobre a aplicação usando conceitos que estão mais próximos a eles sem a necessidade de utilizar jargões computacionais. Como a modelagem conceitual independe do software no qual o sistema é implementado, o projeto resultante se mantém válido caso ocorram mudanças de tecnologia. Neste caso, apenas a transformação entre os esquemas conceitual e lógico é afetada. No caso da tecnologia de SIG isso se torna um fator muito importante, uma vez que grandes investimentos são preservados e há uma redução de custos e aumento das chances de sucesso em caso de mudança para tecnologias mais modernas. Por último, a modelagem conceitual facilita a troca de informações entre parceiros de diferentes organizações, uma vez que aumenta a capacidade de entendimento da semântica da informação, facilitando o uso correto da mesma.

Modelagem de dados é o processo de abstração em que somente os elementos essenciais da realidade observada são enfatizados, descartando-se os elementos não essenciais. O processo de modelagem conceitual de banco de dados compreende a descrição e definição dos possíveis conteúdos dos dados, além de estruturas e de regras a eles aplicáveis.

A modelagem conceitual é sempre feita com base em algum formalismo conceitual (ex.: Entidade-Relacionamento, Orientação a Objetos),

independentemente do nível de abstração empregado. O resultado do processo de modelagem, denominado esquema conceitual, é apresentado através de uma linguagem formal de descrição que possui uma sintaxe e uma notação gráfica. Para cada formalismo conceitual, existem diversas linguagens de descrição de esquema que são compatíveis com o formalismo.

Foram encontrados na literatura diversos modelos conceituais de dados específicos para aplicações SIG. Bédard *et al.* (1996) propõem o MODUL-R, modelo derivado do modelo EER (*Extended Entity-Relationship*). Hadzilacos e Tryfona (1997) propõem o modelo Geo-ER (*Geographic Entity-Relationship*), outra extensão do modelo EER. Shekhar *et al.* (1997) propõe o modelo GISER (*Geographical Information Systems Entity Relationship*), também na mesma linha formal dos outros dois anteriores. O modelo GeoOOA (*Geographic Object Oriented Analysis*) utiliza como base o formalismo da orientação a objetos (KOSTERS; PAGEL, 1996) e (KOSTERS; PAGEL; SIX, 1996), estendendo seus principais conceitos (classe, especialização, agregação) para suportar características espaciais. Seguindo o mesmo paradigma de orientação a objetos, o MADS (*Modeling of Application Data with Spatial-temporal features*) é outro modelo proposto por Parent *et al.* (1998) e Parent, Spaccapietra, Zimanyi (1999).

No Brasil também são desenvolvidos projetos no sentido de se disponibilizar modelos conceitualmente completos para representação de dados espaço-temporais. Borges (1997) e Borges, Davis e Laender (2001) propõem o OMT-G, uma linguagem de modelagem de aplicações geográficas baseada na OMT. Pires, Medeiros e Barros (1993) e Lisboa Filho (2000), introduzem o conceito de *framework* para construção de aplicações geográficas. Esses *frameworks* propõem, basicamente, uma hierarquia de classes que podem ser utilizadas para modelagem em nível conceitual de aplicações espaço-temporais.

A seguir são apresentados os principais requisitos para modelagem de SIGs e um breve resumo sobre alguns modelos citados, mostrando as principais características que ajudarão na implementação deste trabalho. Depois, são apresentados como alguns modelos geográficos desenvolvidos pela comunidade científica atendem aos requisitos citados. Ainda, mostramos dois *frameworks* de software para construção de aplicações geográficas e por fim, uma breve descrição dos conceitos de análise e projeto, que serão úteis no desenvolvimento da proposta central deste trabalho.

3.2 Requisitos para Modelagem Espaço-Temporal

Friis-Cristensen, Tryfona e Jensen (2001) apresentaram um levantamento dos principais requisitos para propriedades de dados geográficos e que devem ser considerados no desenvolvimento de notações para modelagem de dados geográficos. São eles:

Localização e extensão: localização espacial e extensão definida pelas coordenadas x, y e z em um sistema de referência específico e representado por pontos, linhas e polígonos.

Extensão temporal: é necessário registrar a existência e mudanças em objetos ao longo do tempo.

Extensões espaciais complexas: deve ser possível representar associações de objetos espaciais, formando objetos complexos, tais como multi-pontos, multi-linhas e multi-polígonos.

Valores temáticos: um objeto pode ter atributos que definem suas propriedades.

Visão entidades e visão campo: no enfoque baseado em entidades, o mundo real consiste de um conjunto de entidades, tais como estradas e construções. Por sua vez, na abordagem baseada em campos, o mundo real consiste em atributos que variam sobre o espaço de maneira contínua, tal como fenômenos clima e altura (GOODCHILD, 1992).

Generalização: relaciona objetos a aspectos de escala.

Restrições: deve ser possível definir restrições a um objeto. Um exemplo de restrição seria a limitação de um determinado valor dentro de um intervalo limitado, tal como a restrição da coordenada de uma amostra dentro da área de um talhão.

Papel: um objeto geográfico pode ser definido de diferentes maneiras dependendo do universo em análise, em um mesmo contexto. Uma mesma instância de um objeto geográfico pode assumir papéis ou tipos diferentes em um mesmo sistema.

Identificador de objetos: um identificador único para objetos é muito útil para troca de dados.

Qualidade de dados: descreve o grau em que os dados são bons. Um dado é considerado bom quando é possível medi-lo, tal como a precisão da posição de um objeto em relação a outro objeto de referência.

Relacionamentos topológicos: relações e conexões entre objetos. A seção 2.7.1. Representação Geográfica desta tese apresenta uma descrição desses relacionamentos.

Relacionamentos métricos: envolvem distâncias e dependem das posições absolutas dos objetos em um sistema de referência. A seção 2.7.1. Representação Geográfica desta tese apresenta uma descrição desses relacionamentos.

Relacionamentos todo-parte: um objeto pode ser composto de outros.

Restrições em relacionamentos: a restrição de relacionamentos depende do tipo de relacionamento entre os objetos. Uma restrição pode ser topológica, por exemplo, não pode ser permitido colocar uma construção na área de um lago.

Lisboa Filho e Lochpe (1999) também definem um conjunto de requisitos, tais como visões de campos e objetos, aspectos temáticos, aspectos espaciais, múltiplas representações, relacionamentos espaciais, aspectos temporais e aspectos de qualidade, que podem ser relacionados com os requisitos apresentados por Friis-Cristensen, Tryfona e Jensen (2001).

3.3 Modelos baseados no Formalismo Entidade e Relacionamento

3.3.1 GISER

O modelo GISER (*Geographic Information System Entity Relational*) (Shekhar *et al.*, 1997) estende o formalismo do modelo EER, adicionando linhas tracejadas para representar campos e relacionamentos espaciais (Figura 3.1). É baseado em quatro conceitos principais: *Space/Time*, *Features*, *Coverages* (definições geográficas de um conjunto de feições e demais atributos descritivos) e *Spatial Objects*.

Space/time representa o espaço no qual fenômenos geográficos e eventos podem ocorrer tendo uma posição relativa e uma direção. Representa uma superfície contínua que pode ser discretizada se necessário, como por exemplo, a superfície da Terra e seus subconjuntos de interesse.

Figura 3.1. Modelo GISER

Fonte: baseado em (SHEKHAR et. al, 1997)

Features representam os fenômenos geográficos como rios, vegetação e cidades. As *features* são modeladas no modelo GISER como campos contínuos variando no tempo e espaço como, por exemplo, tipo de solo, altimetria. O modelo GISER torna explícito o fato de que entidades contínuas precisam ser discretizadas para representação nos sistemas de banco de dados. As *features* são compostas em elementos denominados *coverage*. Uma *feature* pode ter múltiplas *coverages* que variam com as múltiplas representações do fenômeno geográfico e é modelada usando o relacionamento *varies over*. Uma *coverage* consiste num conjunto de objetos espaciais (*spatial object*) distribuídos no tempo e espaço (*space, time*) e que possuem propriedades geométricas e/ou topológicas.

3.3.2 Geo-ER

Hadzilacos e Tryfona (1997) definem em seu trabalho uma extensão do modelo de Entidades e Relacionamentos para suportar aplicações geográficas, o modelo Geo-ER. Esse modelo possui mecanismos semânticos para modelagem de espaço, localização e dimensionamento de objetos, relacionamentos espaciais e atributos espaciais.

A Figura 3.2 mostra o modelo de entidades geográficas do Geo-ER. Neste modelo, o espaço é representado pelo conjunto de entidades POSITIONS, que é relacionado a uma hierarquia de tipos geométricos: *0-Dimensional*, *1-Dimensional* e *2-Dimensional* (ponto, linha e polígono, respectivamente).

Figura 3.2. Modelo de entidades geográficas

Fonte: (HADZILACOS; TRYFONA, 1997)

Um relacionamento especial 1:M (um-para-muitos) foi criado, para relacionar uma entidade geográfica à sua posição no espaço. Desse modo, uma entidade geográfica pode ser modelada de acordo com diferentes pontos de vista. Por exemplo, uma cidade pode ser representada como um ponto ou como um polígono. A Figura 3.3 mostra o exemplo de modelagem da entidade geográfica Cidade (*City*).

Figura 3.3. Entidade Geográfica Cidade

Fonte: (HADZILACOS; TRYFONA, 1997)

Outros dois conceitos são introduzidos para capturar agregações de entidades geográficas, os conjuntos complexos de entidades geográficas. O primeiro conceito é *spatial_part_of*, utilizado para definir agregações por valor, onde as partes são dependentes do todo. Como exemplo, podemos citar a entidade *City*, conforme Figura 3.4, que é composta das partes nome e nome do país (não geográficos), área residencial e área industrial.

Figura 3.4. (a) Agregação Espacial. (b) Um exemplo

Fonte: (HADZILACOS; TRYFONA, 1997)

O segundo conceito é o *spatial_member_of*, definido para representar agregações por referência. Neste caso, as entidades partes são independentes do

todo. A Figura 3.5 mostra o exemplo de uma rede subterrânea, cujas partes são segmentos da rede.

Figura 3.5. Agrupamento Espacial e um exemplo

Fonte: (HADZILACOS; TRYFONA, 1997)

No primeiro caso, a dimensão total das partes não necessariamente é igual à dimensão do todo, mas no segundo caso, obrigatoriamente, a dimensão total das partes sempre será igual a dimensão do todo. Entretanto, a diferença principal dos dois conceitos reside no aspecto semântico do modelo.

Posteriormente, Tryfona (1998) propõe novas construções (Figura 3.6) ao modelo Geo-ER para representar também informações temporais. Desse modo, o modelo passa a ter capacidade de representação para aplicações espaço-temporais.

Figura 3.6. Construções espaço-temporais

Fonte: (TRYFONA, 1998)

Entidade (*entity type*): conjunto de entidades com estrutura e comportamento similares. Em aplicações espaço-temporais, entidades com dimensão temporal são modeladas com a letra T (*Time*) e a dimensão espacial é

modelada com a letra G (*Geometry*). A dimensão temporal pode ser dos tipos tempo de transação (*transaction time*) e tempo de validade (*valid time*), representados por TT e VT respectivamente. Caso seja do tipo bitemporal, deve ser utilizadas TT e VT na área reservada da figura da entidade. A letra G mostra o tipo de dado geométrico da entidade que pode ser Ponto, Linha, Região (Polígono) ou uma combinação destes.

Relacionamento (*relationship type*): representa o conjunto de associações entre entidades. Objetos espaciais são associados a outros no espaço e representam as condições atuais sobre o posicionamento dos objetos espaciais. São divididos em relações topológicas, direcionais e métricas. Essas relações já foram discutidas anteriormente nesta tese (Capítulo 2, seção 2.7).

Atributo (*attribute*): propriedades estáticas de entidades ou relacionamentos.

Campo (*field type*): representa um conjunto, complexo ou não, de atributos espaciais. Um atributo espacial está associado a um objeto situado em alguma posição do espaço. Por exemplo, a “temperatura de uma cidade”. Esses atributos podem receber valores contínuos, como por exemplo, o próprio atributo “temperatura” ou valores discretos, como por exemplo, “classe de temperatura”. Assim como as entidades, um atributo espacial pode possuir a representação espaço-temporal.

Fenômeno (*phenomenon type*): Um fenômeno é o resultado da combinação de campos em uma mesma área, por exemplo, a combinação de “temperatura”, “pressão” e “umidade” podem causar o fenômeno “chuva”.

Restrição (*constraint type*): representa o relacionamento entre entidades e tem o significado de um gatilho (*trigger*). Se verdadeiro, o fenômeno aparece. A Figura 3.7 mostra um exemplo de modelagem de um fenômeno espaço-temporal.

Figura 3.7. Um exemplo de modelagem de um fenômeno espaço-temporal

Fonte: (TRYFONA, 1998)

3.3.3 MODUL-R

MODUL-R (BÉDARD et al., 1996) é um formalismo em nível conceitual adaptado para banco de dados espaço-temporais, derivado do modelo EER. A principal característica do formalismo MODUL-R é sua capacidade em propor, de maneira única, modelagem temporal e espacial, utilizando módulos específicos para cada fim.

Basicamente, os conceitos atômicos do formalismo MODUL-R são os mesmos do modelo EER, tais como entidades, relacionamentos e atributos. O enfoque de Orientação a Objetos (OO) contribui para o formalismo MODUL-R com conceitos como agregação, especialização e generalização.

Para a representação de entidades espaciais são utilizados pictogramas que definem a dimensão da entidade espacial (0-D, 1-D, 2-D, 3-D). O objetivo é diminuir a complexidade dos modelos em nível conceitual para especificar uma estrutura completa de entidades geométricas. A Figura 3.8 apresenta os pictogramas utilizados pelo formalismo MODUL-R.

Figura 3.8. Pictogramas Espaciais

Fonte: (BÉDARD et al., 1996)

Da mesma maneira seguida para referências espaciais, as referências temporais também são representadas por pictogramas, que indicam formas temporais (0-D ou instante, 1-D ou duração). Eles podem representar a existência, presença e função de entidades e a evolução de atributos. A Figura 3.9 mostra os pictogramas temporais do formalismo MODUL-R.

Figura 3.9. Pictogramas Temporais

Fonte: (BÉDARD et al., 1996)

A principal contribuição do formalismo, segundo o autor, é a representação semântica mais sucinta, principalmente pela utilização dos pictogramas que tornam o modelo resultante mais legível e, portanto, mais compreensível.

3.4 Modelos baseados no formalismo Orientado a Objetos

3.4.1 MADS

O MADS (*Modeling of Application Data with Spatio-Temporal features*) (PARENT *et al.*, 1998) é um modelo baseado no formalismo de orientação a objetos. O modelo é definido por uma hierarquia espacial, cuja superclasse é a classe Geo. Essa hierarquia se divide em objetos simples e objetos complexos. Objetos simples são geometrias únicas, tal como um polígono (*simple area*) ou um ponto (*point*). Objetos complexos são conjuntos de geometrias, tal como um conjunto de pontos (*point set*) ou um conjunto de linhas (*line set*). A Figura 3.10 apresenta a hierarquia de objetos do MADS.

Figura 3.10. Hierarquia de tipos espaciais

Fonte: (PARENT *et al.*, 1998)

O modelo utiliza também ícones para representar as geometrias, conforme pode ser observado na hierarquia de objetos da Figura 3.10. Segundo os autores, o modelo conceitual deve ser o mais legível possível, inclusive para não especialistas, e os ícones auxiliam para utilização e entendimento do modelo.

Atributos espaciais são definidos com relacionamentos entre duas geometrias. Por exemplo, na Figura 3.11, uma Cidade (*City*), modelada como um ponto, contém vários Condados (*County*), este modelado como uma área complexa. Ao mesmo tempo, a associação entre *City* e *County* representa um relacionamento topológico entre as duas geometrias (*contains*).

Figura 3.11. Relacionamentos topológicos

Fonte: (PARENT et al., 1998)

O MADS também permite representar outros tipos de relacionamentos topológicos, tais como disjunção, adjacência e sobreposição, sendo que todos são representados por ícones diferenciados. Associações de agregação e generalização entre objetos espaciais são permitidas no MADS, inclusive a construção de herança múltipla é possível.

A característica temporal de dados espaciais também foi contemplada no MADS. Segundo Spaccapietra, Parent e Zimanyi (1998), a capacidade de planejar o futuro baseado em experiências passadas é fundamental, especialmente em aplicações espaciais. Desse modo, os autores criaram mecanismos conceituais no modelo para suportar aspectos temporais. Esses mecanismos complementam o poder semântico do MADS, inserindo ícones que representam o dinamismo temporal de aplicações geográficas. A Figura 3.12 apresenta exemplos de utilização do MADS com características temporais.

Figura 3.12. Modelagem temporal com MADS

Fonte: (SPACCAPIETRA; PARENT; ZIMANYI, 1998)

No primeiro exemplo (a), é mostrada a entidade “Empregado” (*Employee*) e seus atributos temporais, tal como o projeto em que o empregado está trabalhando e trabalhou recentemente, o salário e seus dependentes. Em (b), o empregado é associado a projetos por um período de tempo. Neste caso, o relacionamento é temporal (*WorksOn*), o que mostra que os períodos em que um empregado esteve alocado em determinados projetos são representados.

3.4.2 PVL e UML

O conceito PVL (*Plug-in for Visual Languages*) foi introduzido por Bédard (1999), e corresponde a um conjunto de pictogramas e uma linguagem gráfica utilizada para descrever qualquer possível geometria. A idéia central é que o PVL ofereça um alto nível de abstração, escondendo a complexidade inerente à modelagem de primitivas espaciais e temporais. Esse enfoque é compatível com práticas de extensão de formalismos, tal como os estereótipos UML. Para desenhar os diagramas, a ferramenta CASE *Perceptory* foi desenvolvida (PERCEPTORY, 2006).

PVL é composto basicamente de três pictogramas espaciais e três pictogramas especiais. Esses pictogramas são feitos de formas (0D, 1D, 2D) e representados por . O pictograma ponto pode ser utilizado para modelar classes de objetos como hidrantes, a geometria linear para modelar uma estrada e a geometria de superfície pode ser utilizada para modelar áreas como um parque, por exemplo.

Em classes de objetos, o pictograma deve ser colocado no lado esquerdo do nome da classe. Quando aplicado em atributos, os pictogramas indicam que o valor de um atributo varia dentro da geometria de um objeto. Na Figura 3.13, por exemplo, o pictograma linear do lado direito do atributo “*number of lanes*” (número de pistas) da classe “*Road*” (Estrada) é compatível com a percepção de uma pessoa do número de pistas como uma característica da estrada e que varia espacialmente.

Figura 3.13. Esquema espacial construído com PVL

Fonte: (BÉDARD et al, 2004).

As possíveis formas de pictogramas que representam geometrias e tempo são apresentadas no Quadro 3.1.

Geometria simples	
Ex.	Pictograma com multiplicidade (1,1), implícita e não escrita. Cada instância é representada por uma e somente uma geometria simples. Por exemplo, um hidrante é representado por exatamente uma geometria.
Ex.
Ex.
Geometria composta (AND)	
Ex. _{1,N}	Agregação simples, em que cada instância é representada por um agregado de formas de mesma dimensão. Por exemplo, um pomar pode ser representado por um conjunto de pontos.
Ex. _{1,N}	
Ex.	Agregação complexa, onde uma instância é representada por um conjunto de formas de dimensões diferentes. Por exemplo, grandes rios podem ser representados por linhas e por polígonos, em sua parte mais caudalosa.
Ex.
Ex.
Geometria alternada (XOR)	
Ex.	Uma instância é representada por uma geometria ou por outra, mas não ambas (OR exclusivo). Por exemplo, uma construção pode ser representada por um ponto se é relativamente pequena ou por um polígono, se for relativamente grande.
Ex.
Ex. _{1,N} _{1,N}	
Ex. _{1,N}	

Múltiplas geometrias	
Ex. Ex. _{1,N}	Uma instância é representada pelas geometrias usadas, mas usualmente somente uma é utilizada por vez. Por exemplo, uma cidade possui duas formas para representá-la, um polígono que delimita sua área e um ponto representando o centro da cidade.
Temporais
0D: instante temporal 1D: intervalo temporal

Quadro 3.1. PVL para modelar geometrias de objetos

Fonte: baseado em (BÉDARD *et al.*, 2004)

Foram definidos também pictogramas temporais, utilizados para modelar tempo nas aplicações espaciais. Conforme Quadro 3.1, o PVL espaço-temporal define-se por 0D, ou instante temporal, e 1D, ou intervalo temporal. Para as classes, esses pictogramas devem ser colocados do lado direito do seu nome e ilustram se a vida de cada ocorrência é considerada instantânea ou durável. Por exemplo, o pictograma de instante temporal pode ser utilizado para classes com existência instantânea, tal como a classe Acidente; e o pictograma que representa um intervalo temporal pode ser utilizado em classes com certa duração, tal como uma Construção. Essa interpretação é intimamente ligada aos tipos temporais Tempo de Validade e Tempo de Transação, discutidos já anteriormente neste trabalho.

A Figura 3.14 mostra um exemplo de utilização do PVL espaço-temporal, onde uma construção possui certa duração de existência e estados também periódicos para o atributo “valor comercial”. A área a ser construída apresenta uma evolução geográfica, na qual a mesma é representada por um ponto caso possua área menor do que 250.000 m² e um polígono caso a área ultrapasse esse tamanho.

Figura 3.14. Exemplo de utilização do PVL espaço-temporal

Fonte: (BÉDARD *et al.*, 2004)

3.4.3 GeoOOA

O GeoOOA (*Geometric Object Oriented Analysis*) (KOSTERS; PAGEL, 1996) e (KOSTERS; PAGEL; SIX, 1996) é uma extensão dos conceitos de orientação a objetos, feita principalmente pela introdução de primitivas geoespaciais para tratamento de requisitos específicos de Sistemas de Informações Geográficas (SIG). Essas primitivas suportam abstrações de tipos de classes espaciais, estruturas topológicas de todo-parte, estruturas de rede e classes temporais.

GeoOOA faz distinção entre classes convencionais e quatro tipos de geoclasses. Nesse sentido, objetos sem propriedades geométricas são modelados com a classe Convencional, objetos de ponto por Classes Ponto, objetos de uma dimensão (1-D) por Classes Linha, objetos bidimensionais (2-D) por Classes Região e objetos digitalizados por Classes Raster. A Figura 3.15 apresenta os símbolos utilizados para representar os tipos de classes mencionados.

Figura 3.15. Símbolos de Classes

Fonte: (KOSTERS; PAGEL; SIX, 1996)

As estruturas topológicas de todo-parte são primitivas que servem para modelar relacionamentos topológicos entre o todo e suas partes que ocorrem em geoplicações. Em GeoOOA, elas foram classificadas em três tipos de estruturas:

Coverage (Figura 3.16): o todo e suas partes pertencem ao mesmo tipo de geoclasse e a geometria do todo é formada pela união das geometrias das partes.

Figura 3.16. Relacionamento Coverage

Fonte: (KOSTERS; PAGEL; SIX, 1996)

Containment (Figura 3.17): a geometria do todo contém a geometria de todas suas partes.

Figura 3.17. Relacionamento *Containment*

Fonte: (KOSTERS; PAGEL; SIX, 1996)

Partition (Figura 3.18): é a estrutura de *containment* com duas condições: o todo e suas partes pertencem a um mesmo tipo de geoclasse e as geometrias das partes formam uma parte da geometria do todo.

Figura 3.18. Relacionamento *Partition*

Fonte: (KOSTERS; PAGEL; SIX, 1996)

O modelo GeoOOA distingue classes temporais e classes não-temporais e representa o relacionamento temporal entre essas classes. (Figura 3.19).

Figura 3.19. Classe e relacionamento temporal do modelo GeoOOA

Fonte: (KOSTERS; PAGEL; SIX, 1996)

3.4.4 GMOD

O GMOD (*Geographic Data Model*) (PIRES; MEDEIROS; BARROS, 1993; SALLES *et al.*, 1998) é um modelo orientado a objetos proposto no ambiente de desenvolvimento de aplicações geográficas denominado UAPE (*geo-User Analysis and Project Environment*). Esse ambiente computacional integra modelagem de dados e processos utilizando tecnologias de engenharia de software e bancos de dados, tendo sido desenvolvido na Universidade de Campinas.

GMOD permite a definição de fenômenos geo-referenciados de acordo com as visões de campos e objetos e possui primitivas para a modelagem da dimensão temporal e relacionamentos entre entidades.

GMOD é dividido nos quatro níveis de abstração: nível do mundo real, nível conceitual, nível de representação e nível de implementação. O modelo é baseado em três conceitos básicos: classes, como nos modelos orientados a objetos; relacionamentos, que permitem a conexão das classes; restrições, que são impostas sobre classes, relacionamentos ou instâncias. Se necessário, uma classe ou relacionamento pode ser definido como temporal.

O modelo GMOD permite que as Geoclasses e as classes Convencionais sejam definidas como sendo temporais, através da associação com a classe Tempo. A classe Tempo é especializada em subclasses que distinguem os tipos característicos do tempo como, discreto ou contínuo, variação linear ou intervalar. GMOD representa os relacionamentos de generalização/especialização, agregação, associação, versão e causal. Relacionamentos de versão são usados para associar diferentes versões de uma mesma entidade, podendo ser usado para relacionar diferentes representações espaciais de uma mesma entidade geográfica ou para associar objetos que registram a evolução temporal dos fenômenos geográficos. Um

relacionamento causal estabelece uma relação de causa-efeito entre dois fenômenos modelados.

3.4.5 OMT-G

No modelo OMT-G (BORGES, 1997; BORGES; DAVIS; LAENDER, 2001), a partir das primitivas convencionais do modelo OMT (RUMBAUGH *et al.*, 1991), foram introduzidas primitivas geográficas que aumentam sua capacidade semântica, diminuindo a distância entre o modelo mental do espaço a ser modelado e o modelo de representação normalmente utilizado.

O OMT-G oferece primitivas para modelar a geometria e a topologia dos dados geográficos, suportando estruturas topológicas “todo-parte”, estrutura de rede, múltiplas visões dos objetos e relacionamentos espaciais. Além disso, o modelo permite a diferenciação entre atributos gráficos e alfanuméricos, e a especificação de métodos associados às classes. A Figura 3.20 mostra a hierarquia de classes do modelo.

Figura 3.20. Modelo Geo-OMT

Fonte: (BORGES, 1997)

A estrutura de classes se caracteriza por uma hierarquia que se divide em duas principais classes: Classe Georreferenciada e Classe Convencional.

A classe *Georreferenciada* descreve um conjunto de objetos que possuem representação espacial e estão associados a regiões da superfície da Terra, representando a visão de campos e a visão de objetos.

A classe *Convencional* descreve um conjunto de objetos com propriedades, comportamento, relacionamentos e semântica semelhantes, e que possuem alguma relação com os objetos espaciais, mas que não possuem propriedades geométricas. Um exemplo desse tipo de classe é a que define fazendeiros que possuem relação de propriedade com as fazendas e os talhões presentes no banco de dados geográfico.

Os objetos da classe *GeoCampo* representam objetos distribuídos continuamente pelo espaço, correspondendo a grandezas como tipo de solo, topografia e teor de minerais.

Os objetos da classe *GeoObjeto* representam objetos geográficos individualizáveis, que possuem identificação com elementos do mundo real, como lotes, rios e postes. Esses objetos podem ter ou não atributos não-espaciais, e podem estar associados a mais de uma representação geométrica, dependendo da escala em que é representado, ou de como ele é percebido pelo usuário (abstração).

Abrantes e Gomes (1998) também propõem uma extensão ao OMT, denominada Geo-OMT. Neste caso, esse modelo é voltado exclusivamente para aplicações de redes de telecomunicações. Segundo os autores, esse tipo de aplicação necessita de modelos que suportem características espaciais, pois envolvem componentes de cabos, dutos de cabos e conectores que são projetados sobre mapas de regiões ou construções.

3.4.6 Temporal UML

Svinterikou e Theodoulidis (1999) apresentam uma notação semântica baseada em UML (*Unified Modeling Language*), o TUML (*Temporal Unified Modeling Language*). Esse método considera os aspectos temporais de elementos básicos que compõem a semântica da linguagem, as Classes, os Atributos e as Associações. Tais aspectos temporais são representados pelo mecanismo de extensão da linguagem, o chamado estereótipo (*stereotype*). Os novos estereótipos criados são chamados de *temporal stereotypes*. A seguir são apresentados os

estereótipos criados para a abstração de classes. Similarmente, são definidos estereótipos temporais para atributos e associações.

`<<transaction time class>>`: representa o momento que um objeto sofre algum tipo de operação.

`<<valid time class>>`: representa o período de tempo em que o objeto é válido na realidade modelada.

`<<bitemporal time class>>`: representa as duas informações, *transaction time* e *valid time*.

O método é capaz de suportar os três tipos de estados temporais (tempo de validade, tempo de transação e bitemporal), sendo que a utilização dos tipos depende da necessidade do modelo.

Os tipos de dados definidos para representar os tipos temporais são: *Date*, *Time*, *Timestamp*, *Interval*, *Timepoint* (refinamento do tipo *Timestamp* para suportar granularidade), *Period*, *Temporal_Element* (conjunto de períodos).

A Figura 3.21 mostra um exemplo de como modelar um domínio de uma aplicação em UML com os estereótipos temporais.

Figura 3.21. TUML: estereótipos temporais

A classe `Turma` representa a abstração de classificação - a abstração de classificação implica no agrupamento de entidades com mesmas características (BOOCH; RUMBAUGH; JACOBSON, 2005) - para a realidade turmas de alunos em uma escola. O estereótipo `<<valid time>>` indica que a informação do período em

que uma turma é válida é relevante, portanto representada pelo mecanismo de extensão sem que haja a necessidade de definir demais atributos para a classe. Da mesma maneira, a associação entre as classes `Turma` e `Professor` indica essa característica. Um atributo também pode possuir essa carga semântica, para qualquer um dos tipos temporais. Por exemplo, o atributo salário da classe `Professor` é do tipo bi-temporal, ou seja, possui informações de tempo de validade e de tempo de transação.

3.4.7 *Framework* GeoFrame – UML

O conceito de *framework* é muito utilizado nesse modelo. Um *framework* pode ser definido como um padrão arquitetural que fornece um modelo extensível para aplicações dentro de um domínio (BOOCH; RUMBAUGH; JACOBSON, 2005). Nesse sentido, esse modelo é tido como um *framework* conceitual que fornece uma hierarquia de classes básicas para auxiliar o projetista tanto na modelagem conceitual de dados geográficos como, também, na especificação de padrões de análise em bancos de dados geográficos.

O GeoFrame (LISBOA FILHO, 2000) foi definido de acordo com as regras do formalismo da orientação a objetos, utilizando a notação gráfica do diagrama de classes da linguagem UML. O mesmo autor também define um conjunto de padrões de projeto para modelagem de sistemas geográficos, baseado no GeoFrame. A Figura 3.22 mostra o diagrama de classes do GeoFrame.

Figura 3.22. Modelo GeoFrame

Fonte: (LISBOA FILHO, 2000)

A estrutura hierárquica de classes do modelo GeoFrame é muito parecida com a do modelo OMT-G. A principal diferença reside nas classes Tema e Região Geográfica. Essas classes formam a base de qualquer aplicação geográfica. Cada aplicação geográfica tem como objetivo o gerenciamento e a manipulação de um conjunto de dados para uma determinada região de interesse, constituindo o banco de dados geográfico. Para cada região geográfica pode-se especificar uma coleção de temas. O agrupamento de classes que descrevem os fenômenos geográficos em temas funciona como um mecanismo para redução da complexidade em grandes esquemas de dados. O uso de temas permite ao projetista dividir o esquema de dados em subesquemas coesos, nos quais são agrupadas classes que estão fortemente relacionadas entre si. Conjuntos de temas afins podem ser agrupados em um tema mais genérico, formando uma hierarquia de temas.

Originalmente, esse *framework* ficou desprovido de mecanismos para representar conceitos temporais. No entanto, Rocha (2001) e Rocha e Edelweiss (2001) promoveram evoluções, e assim o modelo passou a ser denominado *framework* GeoFrame-T, sendo adicionado características temporais ao *framework* original. Basicamente, os autores utilizaram a TUML (SVINTERIKOU,

THEODOULIDIS, 1999) para estender o GeoFrame com mecanismos de modelagem conceitual para aplicações temporais.

Para tornar possível a representação dos conceitos temporais no GeoFrame criou-se a classe `ObjetoTemporal`, apresentada na Figura 3.23. A classe `ObjetoTemporal` é um agregado da classe `ObjetoTempo` que pode ter ou não instância da classe `MetadadosTemporais` associada a si.

Figura 3.23. Modelo temporal para o GeoFrame

Fonte: (ROCHA, 2001)

A classe `MetadadosTemporais` descreve os metadados da classe `ObjetoTempo`. A classe `ObjetoTempo` é uma generalização das classes `Estático` e `TipoTemporal`, sendo que esta é especializada em `TempoValidade`, `TempoTransação` e `Bitemporal`. As classes a serem definidas a partir do GeoFrame podem variar em relação ao aspecto tempo, conforme os subtipos da classe `ObjetoTempo` e `TipoTemporal`.

A classe `Estático` foi criada como um artifício a ser utilizado pelas classes que não terão os seus valores mantidos ao longo do tempo, já que era necessário representar no GeoFrame que as classes temporizadas herdam o aspecto temporal.

da classe `ObjetoTemporal`. A solução encontrada foi a criação desta classe para que fosse possível generalizar também as classes atemporais de uma aplicação de SIG.

A classe `TempoValidade` possui o atributo `Ordemtempo` que define de que forma está sendo considerada a variação temporal. Esta pode assumir os valores: `Linear` e `Ramificado`. Somente a representação do tempo de validade permite tempo ramificado, sendo que o tempo de transação é sempre considerado como linear. Observando o diagrama da Figura 3.23, verifica-se que a classe `TipoTemporal` é formada por `PrimitivaTemporal`. Assim, o tempo no `GeoFrame-T` pode ser representado através de:

- `InstanteTemporal` quando é considerado apenas um ponto no tempo;
- `IntervaloTemporal` quando se tem um valor para o início e outro para o fim do intervalo de tempo de validade ou de transação de um elemento;
- `ElementoTemporal`, que é a união de intervalos e instantes temporais de tempo de validade ou de tempo de transação.

Finalmente, o *framework* `GeoFrame-T` é mostrado na Figura 3.24.

Figura 3.24. Framework GeoFrame-T

Fonte: (ROCHA, 2001)

3.5 A Biblioteca TerraLib

A biblioteca TerraLib (CÂMARA *et al.*, 2000) é um projeto que basicamente é constituído por uma biblioteca de classes escritas em C++ para a construção de aplicativos geográficos, com código-fonte aberto e distribuído como um software livre. Destina-se a servir como base para o desenvolvimento cooperativo na comunidade de usuários ou desenvolvedores de SIG.

TerraLib fornece funções para a decodificação de dados geográficos, estruturas de dados espaço-temporais, algoritmos de análise espacial além de propor um modelo para um banco de dados geográficos.

Uma das características mais importantes da TerraLib é a sua capacidade de integração a sistemas de gerenciamento de bancos de dados objetos-relacionais (SGBD-OR) para armazenar os dados geográficos. A TerraLib trabalha em um modelo de arquitetura em camadas (CÂMARA *et al.*, 2000), funcionando como a camada de acesso entre o banco e a aplicação final.

Um banco de dados TerraLib é formado tanto por dados geográficos (suas geometrias e seus atributos) quanto por um conjunto de informações sobre a organização desses dados no banco, ou seja, os metadados da biblioteca. As tabelas de metadados são automaticamente geradas ao se criar um novo banco de dados TerraLib. A Figura 3.25 apresenta as principais classes do modelo conceitual (metadados) da biblioteca.

Figura 3.25. Modelo Conceitual TerraLib

Fonte: (VINHAS; FERREIRA, 2005)

O repositório de informações geográficas e descritivas é representado pela classe *TeDatabase*. Esse repositório pode ser persistido em alguns diferentes SGDBs (Sistemas Gerenciadores de Banco de Dados). Esse repositório é composto de camadas (*layers*) que representam um conjunto de informações espaciais que são localizadas sobre uma mesma região geográfica. Exemplificando, mapas temáticos (mapa de solos, mapa de vegetação) e os mapas cadastrais de objetos geográficos (mapa de municípios do estado de São Paulo). Os *layers* são reconhecidos pela biblioteca por funções de importação de dados nos formatos ASCII-SPRING, MID/MIFF, GeoTiff, JPEG, entre outros. Cada *layer* possui a informação de quais representações geométricas (geometrias) possui. O componente espacial dos dados pode ser do tipo vetorial (pontos, linhas, polígonos) e matricial (grades regulares multi-dimensionais).

Uma projeção cartográfica representa a referência geográfica do componente espacial dos dados geográficos. Uma referência geográfica é a representação da superfície terrestre em uma superfície plana e é utilizada para minimizar as deformações inerentes ao processo de projeção de um elipsóide em um plano. Cada *layer* é associada a uma projeção cartográfica. Finalmente, temas definem uma seleção sobre os dados de um *layer* e uma Vista representa a visão que determinado usuário possui sobre os temas.

Conforme já mencionado, um *layer* é formado por um conjunto de geometrias (representações). As geometrias vetoriais da biblioteca são: Pontos, Linhas, Anéis e Polígonos. Todas essas geometrias são construídas a partir de coordenadas bi-dimensionais. O trecho de código da Figura 3.26 exemplifica a instanciação de geometrias disponíveis na biblioteca.

```
// Cria um conjunto de linhas
TeLine2D reta;
reta.add(TeCoord2D(500,500));
reta.add(TeCoord2D(600,500));
reta.add(TeCoord2D(700,500));
reta.objectId("reta");
TeLine2D ele;
ele.add(TeCoord2D(700,700));
ele.add(TeCoord2D(800,600));
ele.add(TeCoord2D(900,600));
ele.objectId("ele");
TeLineSet ls;
ls.add(reta);
ls.add(ele);
```

Figura 3.26. Instanciação de Geometrias com TerraLib

Todos os objetos criados podem ser persistidos no banco de dados TerraLib, através da classe `TeLayer` ou pela interface `TeDatabase`. Esses elementos possuem métodos com a finalidade de criar e modificar tabelas, inserir, alterar e recuperar dados.

A TerraLib fornece um conjunto de operações espaciais sobre as geometrias vetoriais e matriciais (FERREIRA; PAIVA; CÂMARA, 2003). Essas operações são classificadas em:

- Funções topológicas entre geometrias vetoriais (toca, intercepta, cobre).
- Funções métricas.
- Geração de novas geometrias.
- Operações que combinam geometrias (diferença, união, intersecção).
- Operações zonais e focais sobre geometrias matriciais.

Essas operações são executadas por recursos do próprio SGBD, nos casos nos quais o mesmo implementa extensões espaciais, ou pela própria biblioteca, para SGBDs relacionais. O trecho de código da Figura 3.27 apresenta um exemplo de chamada de operações espaciais sobre geometrias vetoriais.

```

TeLine line1;
TeLine line2;
bool res = TeOverlaps(line1, line2);
TePolygon pol1;
TePolygon pol2;
res = TeOverlaps(pol1, pol2);

```

Figura 3.27. Operações Espaciais com TerraLib

3.6 Deegree: Software Livre para Construção de SDI

O termo *Spatial Data Infrastructure* (SDI) é utilizado para referenciar importantes tecnologias, políticas e arranjos institucionais que disponibilizam e facilitam o acesso a dados espaciais. A SDI fornece a base para busca, avaliação e aplicação de dados espaciais por usuários e provedores em todos os níveis governamentais, setores comerciais, pesquisadores e cidadãos em geral. A palavra infra-estrutura é utilizada para promover o conceito de um ambiente confiável, análogo a uma rede de telecomunicações, que neste caso, facilita o acesso a informações geográficas com o uso de um conjunto mínimo de práticas, protocolos e especificações.

Algumas pesquisas estão sendo desenvolvidas para se construir um SDI. Nölle (2004) propõe em seu trabalho a utilização de um SDI para fornecer serviços geoespaciais. Esse SDI deverá fornecer acesso a várias fontes de dados externas, promovendo o acesso fácil, rápido e seguro dos dados distribuídos. Korduan, Bill e Bölling (2004) acreditam que a informação geográfica pode ser mais bem difundida por meio dessa tecnologia. Neste trabalho, os autores propõem um SDI exclusivo para AP. Segundo os autores, a AP envolve variabilidade espacial e a reboque, envolve dados espaciais sobre solo, cultura, máquinas agrícolas e aplicações. Análises espaciais são realizadas e diferentes formatos de dados e diferentes sistemas de referência espacial são utilizados, gerando enorme esforço de transformação de dados para que os mesmos possam ser utilizados em diferentes SIGs proprietários. Dessa maneira, o SDI proposto tem como principal objetivo melhorar a interoperabilidade dos dados espaciais.

O principal desafio para construção de um SDI é não se apoiar em uma arquitetura dependente de um único fornecedor, o que comprometeria investimentos e não garantiria um padrão de mercado. Desse modo, o *framework* Deegree

(FITZKE *et al.*, 2004; DEEGREE, 2006) vem ao encontro desse desafio, por ser uma tecnologia naturalmente aberta.

O *Deegree* é um software de caráter livre, feito em linguagem JAVA, baseado nas especificações do *Open Geospatial Consortium* (OGC) (OGC, 2006) e Comitê Técnico ISO/TC211. A Figura 3.28 apresenta o modelo arquitetural do *framework*, composto de quatro unidades de processamento. Um processo de seleção (componente Busca) é controlado por restrições de consulta e é responsável em entregar *Features*. *Feature* é o conceito chave do padrão OGC/ISO. O acesso a dados vetoriais via CORBA, SQL ou OLE/COM é feito por uma implementação da especificação *Simple Features* (OPENGIS SPECIFICATIONS, 2006). Uma típica representação de *Features* ou coleções de *Features* é a GML.

O componente *Gerador de Elemento para Display* é responsável em aplicar regras de estilos às *Features* e produzir uma representação gráfica. O resultado é então transformado em uma imagem, com base em algumas restrições tais como profundidade de cores, tamanho da imagem e formato da imagem. Implementações da especificação *Web Map Service* (WMS) (OPENGIS SPECIFICATIONS, 2006) definem uma interface para acessar imagens de mapa para os componentes “*Gerador de Elemento para Display*” e “*Renderizador*”. A imagem do mapa resultante é então mostrada ao usuário, através de um navegador *web* (*browser*).

Figura 3.28. Arquitetura do Deegree

Fonte: (FITZKE et al, 2004)

Do ponto de vista técnico, a adoção de um padrão é determinante para componentes de um SDI, o que permitirá interoperabilidade de informações espaciais entre os diversos componentes. Atualmente, a tecnologia *Web Service* é a tecnologia base para promover colaboração baseada na Web entre todos os domínios de aplicações. A OGC (*Open Geospatial Consortium*), juntamente com o Comitê Técnico ISO/TC211 (*Geographic information/Geomatics*) fornecem as especificações para uma variedade de componentes necessários para a construção de um SDI. O Quadro 3.2 apresenta os serviços OpenGIS que o *framework Deegree* implementa.

Nome	Funcionalidade
<i>Web Map Service</i> (WMS)	Criação de mapas baseada nos formatos raster e vetoriais. Os mapas podem ser visualizados em navegadores para <i>web</i> .
<i>Web Feature Service</i> (WFS)	Acesso a dados vetoriais (geodados), em conformidade com GML 2.1.1.
<i>Web Coverage Service</i> (WCS)	Acesso a dados espaciais no tipo raster, em vários formatos (TIFF, GIF, JPEG, BMP)
<i>Web Catalog Service</i> (CS-W)	Catálogo de serviços para administração e recuperação de metadados que descrevem os dados e serviços espaciais. Permite a recuperação de dados e serviços baseados em critérios de busca espaciais e textuais.
<i>Web Gazetteer Service</i> (WFS-G)	Permite o geo-referenciamento de entidades geográficas, baseado em identificadores textuais.
<i>Web Terrain Service</i> (WTS)	Criação de visões em 3D. Essas visões podem ser visualizadas em um navegador <i>web</i> .
<i>Web Coordinate Transformation Service</i> (WCTS)	Permite a transformação de coordenadas geográficas de um sistema de referência para outro.

Quadro 3.2. Funcionalidades do Deegree.

3.7 Análise e Projeto

O objetivo da disciplina de análise é avaliar / obter o modelo de requisitos de um sistema para alcançar um entendimento mais preciso dos requisitos e elaborar a estrutura de todo o sistema, incluindo a arquitetura. A captura de requisitos deve deixar alguns problemas sem solução, uma vez que a linguagem utilizada deve ser livre de aspectos tecnológicos. Desse modo, o principal propósito da análise é

resolver esses problemas, analisando os requisitos de maneira mais profunda. Segundo Booch, Jacobson e Rumbaugh (1998), a linguagem utilizada na análise é baseada em um modelo de objetos conceitual, chamado de modelo de análise. Os autores fazem a seguinte comparação entre o modelo de requisitos e o modelo de análise (Quadro 3.3).

Modelo de Requisitos	Modelo de Análise
É descrito utilizando a linguagem do Cliente	É descrito utilizando a linguagem do desenvolvedor
Visão externa do sistema: estruturada por casos de uso	Visão interna do sistema: estruturada por classes estereotipadas e pacotes
Utilizado como contrato entre clientes e desenvolvedores sobre o que o sistema deve e não deve fazer.	Utilizado por desenvolvedores para entender como o sistema será projetado e implementado.
Podem conter redundâncias, inconsistências entre os requisitos.	Não pode haver redundâncias e inconsistências entre requisitos.
Captura a funcionalidade do sistema, incluindo a funcionalidade arquiteturalmente significativa.	Resulta em como realizar a funcionalidade dentro do sistema, incluindo a funcionalidade arquiteturalmente significativa.
Define casos de uso que são posteriormente transformados no modelo de análise.	Define realizações de casos de uso, cada um representando a análise de um caso de uso do modelo de requisitos (modelo de casos de uso).

Quadro 3.3. Comparação entre modelo de requisitos e modelo de análise

Por outro lado, a disciplina de projeto (*design*), ao contrário da disciplina de análise, objetiva estabelecer um conhecimento profundo dos requisitos não funcionais e restrições relacionadas a linguagens de programação, reuso de componentes, sistemas operacionais, tecnologias de distribuição e concorrência, banco de dados, interface com usuário, gerenciamento de transações e assim por diante. Uma comparação entre o modelo de análise e o modelo de projeto é apresentada no Quadro 3.4

Modelo de Análise	Modelo de Projeto
Modelo conceitual, pois é uma abstração do sistema.	Modelo físico, pois é o projeto da implementação.
Projeto genérico.	Não é genérico e sim específico de uma implementação.
Menos formal	Mais formal
Menos caro para desenvolver	Mais caro para desenvolver
Menos camadas	Mais camadas
Esboça o projeto do sistema, incluindo sua arquitetura.	Manifesta o projeto do sistema, incluindo sua arquitetura.
Não precisa ser mantido por todo o ciclo de vida do software	Deve ser mantido por todo o ciclo de vida do software.

Quadro 3.4. Comparação entre modelo de análise e modelo de projeto

As duas disciplinas são executadas ao longo do ciclo de vida de desenvolvimento de um software, em mais ou menos intensidade, dependendo do objetivo da fase em que se encontra o desenvolvimento. São disciplinas muito importantes, que significam a transição dos requisitos do software, usualmente escritos em linguagem natural, para modelos prontos para implementar em uma linguagem de programação específica e em uma determinada plataforma (hardware e software).

Segundo estudos publicados por Grady (1992), a fase de projeto, que no modelo de ciclo de vida em cascata engloba as disciplinas de análise e de projeto, consome em torno de 19% do esforço total de um projeto de desenvolvimento. Esse esforço utilizado nessa fase pode ser diminuído com a adoção de padrões, tanto de análise quanto de projeto. Outro ganho significativo pode ser obtido na qualidade do produto, uma vez que os padrões carregam conhecimento adquirido e testado em outras experiências.

Para Gamma, Vlissides e Johnson (1994), o uso de padrões (tradução livre para o termo em inglês *pattern*) está diretamente relacionado com a idéia de reuso em Engenharia de Software. Ainda, Gamma *et al.* (1993) afirmam que a utilização de padrões pode auxiliar no processo de desenvolvimento de software nos seguintes aspectos:

- Padrões fornecem um vocabulário comum para projetistas se comunicarem, documentarem e explorarem alternativas de projeto. Eles reduzem a complexidade do sistema, definindo abstrações que estão acima de classes ou instâncias.
- Padrões constituem uma base reutilizável de experiências para construção de softwares reutilizáveis. Eles fornecem um meio de reutilizar conhecimento adquirido em experiências práticas e agem como blocos de construção, permitindo a evolução para soluções mais complexas; podem ser considerados micro-arquiteturas que contribuem para a arquitetura do sistema como um todo.
- Padrões podem reduzir o tempo de aprendizagem na utilização de uma biblioteca. A experiência adquirida para aprender uma biblioteca de padrões será útil no aprendizado de uma nova biblioteca.

Há dois tipos de padrões, os padrões de análise e os padrões de projeto. O termo padrão de análise foi criado por Fowler (1997) para padrões que capturam modelos conceituais de um domínio da aplicação com o objetivo de promover a reutilização entre aplicações. Segundo Geyer-Schulz e Hahsler (2002), padrões de análise são um importante mecanismo para facilitar a reutilização de conhecimento e contribuem para o processo de desenvolvimento de duas maneiras diferentes. Primeiro, os padrões de análise diminuem o esforço de desenvolvimento de modelos abstratos que capturam os principais requisitos de um problema concreto, possibilitando modelos de análise reutilizáveis com exemplos e uma descrição de suas vantagens e limitações. Segundo, padrões de análise facilitam a transformação do modelo de análise para o modelo de projeto.

Por sua vez, padrões de projeto são soluções elegantes e reutilizáveis para problemas recorrentes que encontramos diariamente no processo de desenvolvimento de aplicativos para o mundo real. Eles tratam da concepção e da interação entre objetos, definindo ao mesmo tempo um padrão de comunicação que é compartilhado por toda a equipe de desenvolvimento. Os padrões definidos por Gamma, Vlissides e Johnson (1994), conhecidos como padrões GoF (*Gang of Four*) são considerados as fundações para todos os outros padrões. Após esses padrões, outros surgiram, mais direcionados em tecnologias distintas, tais como padrões EJB (GRAND, 2001) (MARINESCU, 2002) e padrões J2EE (ALUR; CRUPI; MALKS, 2003).

No contexto de aplicações geográficas, alguns trabalhos de pesquisa foram desenvolvidos no sentido de se desenvolver ou utilizar padrões de análise e projeto para determinados domínios de problemas. Lisboa Filho, Lochpe e Beard (1998) e Lisboa Filho e Lochpe (1999) afirmam que o processo de desenvolvimento de sistemas de informações geográficas pode se beneficiar do enfoque de padrões de análise, uma vez que vários conjuntos de entidades geográficas são recorrentes em muitas aplicações geográficas, e indicam a utilização de catálogos de padrões geográficos como uma maneira eficiente de compartilhar esses padrões com a comunidade de desenvolvedores.

Uma abordagem é a utilização de padrões de projetos definidos na literatura para implementar SIGs. Gordillo, Balaguer e Das Neves (1997) e Gordillo e Balaguer (1998) apresentam uma proposta de arquitetura para SIG utilizando alguns padrões de projeto. Por sua vez, Câmara *et al.* (2001) apresentam uma abordagem baseada em padrões de projeto para o desenvolvimento de aplicações baseadas na biblioteca TerraLib, objetivando reusabilidade de código, legibilidade e clareza. Couto, Casanova e Schwabe (2005) construíram uma aplicação geográfica baseada em uma adaptação do padrão de projeto MVC (*Model-View-Controller*).

Outra abordagem é a definição de padrões específicos para sistemas geográficos. Lisboa Filho, Lochpe e Borges (2002), por exemplo, apresentam uma proposta de padrões de análise para aplicações urbanas e propõem soluções de análise para problemas como modelagem da estrutura das ruas das cidades brasileiras e o tráfego de automóveis nas cidades. Lisboa Filho e Pereira (2002) e Sodr *et al.* (2005) apresentam uma ferramenta CASE que organiza em um catálogo os padrões de análise para aplicações urbanas e argumentam que a abordagem deve proporcionar aumento de qualidade e produtividade na construção das aplicações.

As principais diferenças entre padrões de análise e padrões de projeto são resumidas no Quadro 3.5.

Padrões de Análise	Padrões de Projeto
A forma é descrita através de termos e conceitos pertencentes ao domínio da aplicação.	A forma é descrita através de construtores genéricos de projeto de software
Não servem para propósitos gerais, é importante encontrar o equilíbrio entre obter um padrão muito abstrato e um padrão muito especializado.	Descrevem a estrutura e a dinâmica de seus componentes, esclarecendo suas responsabilidades, podendo ser complementares aos padrões de análise.
Originalmente apresentados de forma mais explicativa, apresentando soluções alternativas, vantagens e desvantagens (FOWLER, 1997).	São apresentados de forma bem estruturada e organizados separadamente em catálogos, como por exemplo, os padrões GoF (GAMMA; VLISSIDES; JOHNSON, 1994)

Quadro 3.5. Comparação entre Padrões de Análise e Padrões de Projeto

3.8 Considerações Finais

Friis-Cristensen, Tryfona e Jensen (2001) realizaram um estudo sobre alguns modelos, entre eles os modelos Perceptory (BÉDARD, 1999), GeoFrame (LISBOA FILHO, 2000) e OMT-G (BORGES, 1997) (BORGES; DAVIS; LAENDER, 2001), detalhados anteriormente nesta seção. A conclusão geral é que os aspectos relacionados a qualidade de dados, restrições e papéis devem ser mais investigados e são requisitos que os modelos avaliados não atendem adequadamente. Nem todos os modelos representam associações espaciais e temporais entre objetos. Quanto às restrições, os modelos herdam as restrições implícitas de associações e atributos, existentes na UML, por exemplo. Nenhum modelo, a exceção do OMT-G (BORGES, 1997; BORGES; DAVIS; LAENDER, 2001), possui restrições geográficas definidas.

Lisboa Filho e Lochpe (1999) apresentam um estudo cronológico sobre alguns dos principais modelos conceituais criados para modelar sistemas de informações geográficas, ilustrado na Figura 3.29. Fazem também um estudo de alguns modelos

e concluem que a maioria deles atende, em menor ou maior grau, os requisitos levantados.

Figura 3.29. Evolução cronológica dos modelos conceituais de dados para SIG

Fonte: (LISBOA FILHO; IOCHPE, 1999)

No entanto, Lisboa Filho e Iochpe (1999) argumentam que a maioria dos projetistas ainda reluta em realizar a fase de modelagem conceitual em seus projetos, não mais devido à ausência de modelos conceituais e sim por deficiências metodológicas.

Os problemas apontados na modelagem conceitual podem ser resolvidos. A representação de papéis por um objeto pode ser implementada com a utilização do padrão de projeto *Decorator* (GAMMA; VLISSIDES; JOHNSON, 1994), por exemplo. Esse padrão permite anexar responsabilidades adicionais a um objeto dinamicamente, permitindo que um mesmo objeto assuma papéis diferentes no sistema.

Quanto às restrições, a UML define uma linguagem OCL (*Object Constraint Language*) (BOOCH, RUMBAUGH; JACOBSON, 2005) que possibilita a construção das restrições de maneira formal. Essa linguagem pode ser estendida para suportar restrições espaciais e temporais, tais como funções topológicas ou sincronização de

tempo entre objetos espaço-temporais. As associações temporais entre objetos também necessitam de definição de padrões.

Em relação à utilização dos modelos conceituais, será mais efetivo se for dado mais um passo no ciclo de vida de desenvolvimento de sistemas, provendo uma abordagem metodológica baseada no reuso, transformando os modelos conceituais em modelos lógicos e componentizados, resolvendo alguns problemas de implementação.

Por fim, o objetivo desse capítulo foi mostrar o estado atual das pesquisas em modelagem conceitual para sistemas de informações geográficas e algumas bibliotecas de apoio à implementação de SIGs. O modelo adotado como referência para este trabalho será o GeoFrame-T (LISBOA FILHO, 2000) (ROCHA, 2001). Esse modelo define conceitualmente a maioria dos requisitos necessários para modelar dados geográficos, tendo se mostrado completo para estender o modelo conceitual MOSAICo com os aspectos espaço-temporais. Outra razão fundamental é que o modelo é baseado em UML, uma linguagem de especificação com propriedades extensíveis e que se tornou padrão tanto na indústria quanto na academia.

4 PLATAFORMA JAVA ENTERPRISE EDITION (JEE)

4.1 Introdução

Segundo Larman (2002), uma arquitetura é um conjunto de decisões significativas sobre a organização do sistema de software, envolvendo a seleção dos elementos estruturais e suas interfaces, o comportamento e como esses elementos interagem e colaboram para alcançar tal comportamento. Nesse contexto, a JEE (JEE, 2006) é mais do que um software para implementação, é uma plataforma baseada em componentes que permite diferentes configurações para alcançar os objetivos ou requisitos do sistema de software. A plataforma JEE é projetada para fornecer suporte do lado servidor e do lado cliente para desenvolvimento de aplicações em camadas e distribuídas para Web. Essas características auxiliam a obter reusabilidade e escalabilidade para as aplicações e no contexto da AP, a tecnologia representa uma plataforma disseminada pela indústria, de tecnologia aberta e baseada na Web, e a portabilidade da linguagem Java.

A arquitetura proposta neste trabalho foi baseada na plataforma JEE, por se tratar de uma tecnologia aberta e padrão Web. Este capítulo apresenta uma breve descrição da plataforma JEE, e apresenta as principais características e componentes da plataforma e alguns estilos arquiteturais que combinam os componentes da plataforma. O objetivo é apresentar alguns conceitos que serão úteis para a compreensão da solução arquitetural proposta.

4.2 Conceitos Básicos

A plataforma *Java Enterprise Edition – JEE* (JEE, 2006) define um padrão para aplicações multicamadas, baseadas em componentes padronizados e modulares. Fornece um conjunto de serviços para esses componentes e implementa detalhes do comportamento da aplicação automaticamente, sem a necessidade de

programação, bastando uma configuração em arquivos descritores, geralmente XML.

A plataforma JEE utiliza recursos tais como *Java Database Connectivity* (JDBC, 2006) para o acesso de base de dados e a tecnologia CORBA para a integração de recursos e um modelo da segurança que protege os dados em aplicações para Internet. Além disso, a plataforma JEE contém suporte a componentes *Enterprise JavaBeans* (EJB, 2006), *Java Servlets* e tecnologia XML.

Cada camada de uma aplicação JEE possui responsabilidades bem definidas e se comunica com as outras camadas por meio de interfaces de serviços bem definidas. A consequência disso é a possibilidade de construir aplicações baseadas em componentes ou subsistemas, facilitando o reuso de partes importantes da aplicação por outros sistemas. A utilização de padrões de projeto, tais como *Factory* e *Adapter* (GAMMA; VLISSIDES; JOHNSON, 1994) e *Front Controller* e *Session Facade* (ALUR; CRUPI; MALKS, 2003), também pode auxiliar na construção de componentes altamente coesos e com baixo acoplamento.

A plataforma JEE possibilita a utilização em escala do produto, pois atualmente a plataforma possui grande aceitação por empresas privadas e públicas, além da existência de inúmeras iniciativas para construção de componentes de código aberto e de utilização geral (JAKARTA, 2006; SOURCEFORGE.net, 2006).

Algumas vantagens são alcançadas com a utilização da plataforma JEE. Uma delas é a portabilidade oferecida pela linguagem Java. A portabilidade e independência de plataforma são normalmente fatores de qualidade essenciais. A portabilidade pode eliminar os custos com pequenas adaptações e recompilação na migração da aplicação de uma plataforma para outra.

Um outro importante recurso da linguagem Java é a capacidade de servir como interface entre Java e o código nativo escrito em C/C++, *assembly* ou outras linguagens, chamado de *Java Native Interface* (JNI). Com ele pode-se acessar a partir de classes do Java, bibliotecas desenvolvidas em linguagens com maior suporte a hardware (baixo nível) ou desenvolver em C/C++ algoritmos críticos onde o desempenho é primordial. Esse recurso pode ser útil para acessar algoritmos de operações topológicas de outras bibliotecas, tal como TerraLib.

Os componentes da plataforma JEE são: *applets*, aplicações clientes, componentes EJB, componentes Web e componentes adaptadores. *Applets* e aplicações clientes são executadas no lado Cliente enquanto os componentes EJB,

Web e adaptadores são executados na plataforma do servidor. As próximas seções apresentam um resumo dos principais componentes JEE.

4.3 Clientes JEE

A plataforma JEE possibilita que sejam utilizados diferentes tipos de clientes para interagir com o servidor de aplicações.

Applets: são clientes baseados em Java que executam dentro de um navegador Web e que têm acesso a todas as características da linguagem de programação Java. São utilizadas quando se deseja que aplicações Web tenham uma interface com usuário mais poderosa.

Cliente de Aplicação: executada sobre a própria plataforma cliente (conjunto de bibliotecas e APIs que suportam o código cliente). Esse tipo de cliente geralmente se comunica com uma camada EJB de uma aplicação JEE utilizando RMI-IIOP (*Remote Method Invocation – Internet Inter-ORB Protocol*).

Java web-start: é um cliente *stand-alone* baseado na API *JFC/Swing* que possibilita a construção de interfaces "ricas" com usuário e ao mesmo tempo a flexibilidade de atualização de sistemas baseados em navegadores Web. A cada execução da aplicação no lado cliente, a tecnologia verifica se existe uma atualização recente do código. Caso exista, a atualização é feita automaticamente para o lado cliente.

4.4 Componentes Web

Servlets: um *Servlet* é um componente que estende a funcionalidade de um servidor Web. Um servidor Web hospeda as classes Java *Servlets* e mantém um conjunto de URLs para um *Servlet*. Uma requisição HTTP para essas URLs vai invocar o *Servlet* mapeado. Ao ser invocado, um *Servlet* gera uma resposta, que possivelmente invocará uma lógica de negócio em um EJB ou uma busca no banco de dados.

JavaServer Pages (JSP): esse componente é responsável por gerar conteúdo dinâmico para um cliente Web. É um documento baseado em texto que descreve como processar uma requisição do cliente para criar uma resposta.

Container Web: os componentes *Web* são hospedados em contêineres JSP e contêineres de *Servlets*, coletivamente denominados como contêineres *Web*. Um *container Web* fornece um conjunto de serviços pelo quais requisições de clientes são recebidas e respostas são enviadas. Esses contêineres suportam protocolos como HTTP e HTTPS.

4.5 Componentes Enterprise JavaBeans (EJB)

EJB é uma tecnologia utilizada no lado servidor para desenvolvimento e implantação de componentes contendo a lógica de negócio de uma aplicação. Há três tipos de EJB: *beans* de sessão (*session beans*), *beans* de entidade (*entity beans*) e *beans* de mensagem (*message-driven beans*). *Session beans* e *entity beans* possuem dois tipos de interfaces, uma interface do componente e a interface *home*. A interface *home* define métodos para criação, recuperação, remoção e acesso aos metadados do componente. As interfaces do componente definem os métodos com a lógica de negócio. As interfaces podem ser dos tipos local ou remota, sendo que interfaces remotas são interfaces baseadas em RMI. Esse tipo de interface permite a localização do *bean* independentemente do servidor em que ele está sendo executado (objetos distribuídos). Por outro lado, um cliente de um EJB que implementa uma interface local deve rodar no mesmo servidor do *bean*.

Um *session bean* é criado para fornecer serviços para a camada cliente de uma aplicação, e representa uma sessão única de um cliente. Pode realizar operações tais como cálculos ou acesso a base de dados. Por sua vez, um *entity bean* é um objeto persistente que representa o dado mantido em um meio persistente (banco de dados). Um *entity bean* pode gerenciar sua própria persistência ou pode delegar esse gerenciamento para o *container*.

Um *message-driven bean* possibilita que clientes assíncronos acessem a lógica de negócio da camada EJB. Esse tipo de *bean* é ativado somente por mensagens assíncronas recebidas de uma fila de mensagens.

Todos os tipos de EJBs citados são hospedados em um container EJB. Um *container EJB* fornece um conjunto de transações e serviços de persistência, acesso aos serviços JEE e APIs de comunicação.

4.6 Arquitetura de Aplicações JEE

As camadas da plataforma JEE, comumente denominadas *tiers*, podem ser combinadas para compor cenários arquiteturais. Segundo Singh, Stearns e Johnson (2002), vários cenários de aplicações podem ser implementados na plataforma JEE. As especificações JEE orientam à diversidade arquitetural, com o objetivo de atender a uma ampla gama de necessidades. A plataforma JEE é flexível o suficiente para permitir que aplicações suportem uma variedade de tipos de Clientes, utilizando o container *Web* e o container EJB opcionalmente.

Os seguintes requisitos podem influenciar na seleção da arquitetura:

- Necessidade de realizar rápidas e freqüentes mudanças na interface do usuário
- Necessidade de particionar a aplicação, separando a lógica de negócio com objetivo de aumentar a modularidade da aplicação.
- Necessidade de simplificar o processo de alocação de recursos humanos no trabalho de desenvolvimento, de maneira que o processo seja independente e ao mesmo tempo permita o trabalho colaborativo.
- Necessidade de gerar um vocabulário para disseminar a lógica de negócio ao time de desenvolvimento e usuários.
- Possibilidade de criar aplicações com uma variedade de fontes, incluindo COTS (*Commercial off-the-shelf*).

Os principais cenários de aplicações JEE são: aplicações multicamadas, cliente *stand-alone*, aplicações centradas na *Web*, aplicações *business-to-business* (b2b).

4.6.1 Aplicações Multicamadas

Em um sistema em multicamadas, cada camada possui responsabilidades e serviços específicos, objetivando reduzir o acoplamento com as demais camadas. A Figura 4.1 ilustra um cenário onde o container *Web* hospeda os componentes *Web* que são exclusivamente dedicados a lógica de apresentação de uma aplicação (páginas JSP e *Servlets* trabalham juntos para gerar conteúdo dinâmico para a *Web* e apresentar a um cliente). O container EJB hospeda componentes da aplicação que

utilizam recursos de dados para responder a requisições da camada *Web*. Essa arquitetura desacopla acesso a dados da interface com o usuário, deixando a lógica de negócio relativamente isolada e melhorando a escalabilidade do sistema.

Figura 4.1. Cenário de aplicações JEE multi-camadas

Fonte: (SINGH, STEARNS; JOHNSON, 2002)

Neste cenário, a XML ocupa um papel importante uma vez que um container *Web* pode produzir ou consumir mensagens XML e viabilizar diversos tipos de clientes. Java e XML são tecnologias complementares, pois Java oferece uma linguagem portátil e XML fornece dados portáteis.

4.6.2 Cliente *Stand-alone*

Clientes *stand-alone* para aplicações JEE podem formar três cenários diferentes:

Clientes EJB: interagem diretamente com EJBs hospedados em um *container*. Esse cenário utiliza RMI-IIOP para comunicação e o servidor EJB acessa os recursos de dados utilizando JDBC ou conectores.

Clientes *Stand-Alone*: implementados em Java ou outra linguagem de programação que consomem conteúdo dinâmico, tipicamente mensagens em XML. Neste cenário, o container *Web* realiza transformações em XML e fornece conectividade para os clientes. A lógica de apresentação ocorre, neste caso, na camada física do cliente e a camada *Web* executa a lógica de negócio e acessa os recursos de dados.

Aplicações Java acessando diretamente recursos de dados: esse tipo de cliente acessa os recursos de dados diretamente através de JDBC ou Conectores. Neste cenário, a lógica de apresentação e a lógica de negócio são implementadas

na camada Cliente, configurando-se em uma arquitetura Cliente-Servidor em duas camadas.

A Figura 4.2 mostra os tipos de clientes *stand-alone*.

Figura 4.2. Cenário de aplicações JEE com clientes *stand-alone*

Fonte: (SINGH, STEARNS; JOHNSON, 2002)

4.6.3 Aplicações centradas na Web

Neste cenário arquitetural, o container *Web* hospeda tanto a lógica de apresentação quanto a lógica de negócio e acessa os recursos de dados com a tecnologia JDBC ou conectores. A principal desvantagem é o acoplamento entre o código de apresentação e negócio, o que desfavorece a escalabilidade do sistema. Porém, a plataforma JEE fornece flexibilidade para adequar a configuração da aplicação a requisitos específicos e esse cenário pode ser o ponto de partida para muitas aplicações JEE por sua simplicidade de implementação. A Figura 4.3 ilustra esse cenário.

Figura 4.3. Cenário de aplicações JEE *Web-centric*

Fonte: (SINGH, STEARNS; JOHNSON, 2002)

4.6.4 Aplicações *Business-to-Business*

Esse cenário foca as interações entre containeres *Web* e containeres *EJB* (Figura 4.4). Mensagens XML sobre o protocolo HTTP é o principal meio de comunicação entre containeres *Web*. Já entre containeres *EJB*, a comunicação pode ser feita com *Java Message Service* (JMS) ou *message-driven beans*.

Figura 4.4. Cenário de aplicações JEE B2B

Fonte: (SINGH, STEARNS; JOHNSON, 2002)

4.7 Considerações Finais

Os possíveis cenários arquiteturais da plataforma JEE mostram que a tecnologia pode ser adequada às necessidades de cada sistema de software. Aplicações podem ser distribuídas para ganhar em escalabilidade, podem ser baseadas na *Web* com interface por navegadores ou interfaces *stand-alone* e podem ser organizadas em camadas, permitindo maior reusabilidade e componentização e obtendo ganhos de produtividade. Para tanto, o arquiteto de software deve adequar essa plataforma aos requisitos da aplicação, mas corre um grande risco de manter uma arquitetura complexa demais e perder principalmente em desempenho. Essa questão pode ser direcionada em duas linhas de atuação. A customização da plataforma em relação aos componentes físicos de software e a utilização de boas práticas de projeto, que envolve a organização lógica do software.

Conforme visto neste capítulo, a plataforma é composta de camadas físicas ou *tiers* (cliente, containeres *Web* e *EJB*) e respectivos componentes que podem ser

combinados para obter o máximo de vantagens da arquitetura e ao mesmo tempo evitar construir uma arquitetura muito complexa inutilmente. Em relação à organização lógica do sistema, existem muitos padrões de projetos específicos para utilização da plataforma JEE (ALUR; CRUPI; MALKS, 2003; GRAND, 2001; MARINESCU, 2002) e também padrões genéricos, tais como os definidos por Gamma *et al.* (1993) e Gamma, Vlissides e Johnson (1994), conhecidos por padrões *GoF* (*Gang of Four*) em alusão aos quatro pesquisadores que definiram os padrões. Esses padrões de projeto são como boas práticas de implementação, que resolvem problemas conhecidos de desempenho, reusabilidade e escalabilidade, embutindo assim conhecimento adquirido em outras experiências. O próximo capítulo apresenta visões arquiteturais abordando as duas questões. Por um lado apresenta os cenários arquiteturais com os componentes JEE e suas vantagens e desvantagens, e por outro, foca na organização lógica de sistemas e propõe alguns padrões de análise e de projeto espaço-temporais de desenvolvimento de aplicações *Web* para AP.

5 UMA PROPOSTA DE ARQUITETURA DE SISTEMAS ESPAÇO-TEMPORAIS PARA AGRICULTURA DE PRECISÃO

5.1 Introdução

Um projeto arquitetural adequado é crucial para suportar aplicações espaço-temporais. Em relação ao banco de dados, elemento essencial nos sistemas para AP, Breunig *et al.* (2003) apontam três principais estilos arquiteturais: a arquitetura em camadas, na qual são utilizados sistemas de banco de dados comerciais, e as extensões espaciais e temporais são feitas na aplicação; a arquitetura monolítica onde o fabricante do sistema gerenciador de banco de dados embute as extensões no próprio banco de dados; e por fim, as arquiteturas extensíveis onde é possível conectar as extensões no próprio sistema gerenciador de banco de dados.

De maneira geral, os sistemas comerciais se encaixam nesses três tipos de arquitetura. Rigaux *et al.* (2003), por exemplo, apresentam um protótipo de banco de dados com arquitetura extensível, que permite que clientes Java acessem os dados com SQL. Alguns fabricantes de gerenciadores de banco de dados relacionais (SGBDR) estenderam seus produtos para suportar dados espaciais. São exemplos o Informix, IBM DB2, Oracle, Postgres e MySQL. Em Queiroz e Ferreira (2005) é possível encontrar uma descrição das principais extensões para os bancos de dados Postgres e Oracle.

Os SIGs são bastante poderosos em relação ao tratamento de dados espaciais, mas normalmente é difícil estender o banco de dados para combinar a manipulação de dados espaciais e não espaciais, principalmente os dados temporais. Segundo Dias, Câmara e Davis Jr. (2005), a maioria das aplicações de tecnologia de geoinformação utiliza representações estáticas de fenômenos espaciais. Porém, conforme os autores afirmam, um conjunto significativo de fenômenos espaciais são inerentemente dinâmicos e as representações estáticas comumente utilizadas não os capturam de forma adequada.

Ferreira *et al.* (2002) argumentam que os novos bancos de dados com extensões espaciais abrem uma nova perspectiva para desenvolvimento de SIGs e propõem uma arquitetura que fornece uma interface única de programação de aplicações. Ainda, Engenhofer (1999) argumenta que novas tecnologias permitem a transição dos sistemas monolíticos para aplicativos geográficos ou sistemas dedicados para necessidades específicas.

Uma dessas novas tecnologias é a *Web*. Dragicevic (2004) aponta algumas direções na utilização de SIG baseado na *Web*, que podem se configurar como tendências para desenvolvimento de sistemas geográficos baseados na *Web*: a disseminação e acesso a dados espaciais; a exploração e visualização de dados espaciais e a modelagem, análise e processamento de dados espaciais. Roddick *et al.* (2004) apontam como uma das mais promissoras linhas de pesquisa, na área de sistemas espaço-temporais, aplicações para dispositivos móveis e sem fio (*wireless*).

Em razão desses comentários, pode-se assumir que a arquitetura para sistemas que envolvem aspectos espaço-temporais deve possuir elementos que permitam a extensibilidade para novas funcionalidades e tecnologias de hardware e software, representação do espaço e tempo de maneira integrada e construção de software baseado na *Web*.

Neste capítulo, uma proposta de arquitetura de software baseada na plataforma JEE é apresentada. Essa arquitetura pode ser configurada em três cenários arquiteturais, a fim de se adequar a diferentes necessidades de desempenho, escalabilidade e outros requisitos. Os três cenários atendem aos requisitos de sistemas para AP e as necessidades espaço-temporais e, ao mesmo tempo, vêm ao encontro das necessidades do mercado com utilização de padrões largamente disseminados pela indústria de software.

Como parte da arquitetura de software, apresenta-se também um catálogo de padrões de análise e de projeto que permeiam todas as camadas lógicas da aplicação e serão úteis no desenvolvimento de aplicações para AP. Esses padrões podem ser utilizados independentemente do cenário adotado, conforme apresentado mais adiante. Por fim, as especificações de cada camada são apresentadas, com o objetivo de detalhar as principais tecnologias envolvidas em cada uma delas.

5.2 Propostas de Cenários Arquiteturais

Para gerenciar a complexidade e a crescente evolução das aplicações para AP é necessário definir uma arquitetura capaz de suportar essas características. Uma arquitetura de software deve definir a aplicação em termos de componentes e interações entre esses componentes. Componentes são pacotes autocontidos de funcionalidades que têm interfaces abertas e claramente definidas que podem oferecer recursos como "*plug and play*". Componentes reduzem a complexidade porque os programadores não precisam saber como eles trabalham internamente, só precisam saber o que os componentes fazem e quais os serviços que eles fornecem (FINGAR, 2000).

Uma arquitetura organizada em camadas lógicas é uma premissa para permitir componentização do código. Isso porque em uma arquitetura em camadas é possível dividir melhor as responsabilidades entre os componentes.

Independentemente das camadas lógicas que compõem a arquitetura proposta, a plataforma JEE permite que vários cenários sejam criados. A configuração dos cenários depende exclusivamente dos requisitos (principalmente não funcionais) da aplicação alvo. Desse modo, são propostos neste trabalho três cenários arquiteturais baseados em camadas físicas (*tiers*) e lógicas (*layers*). Todas as propostas são aderentes a uma arquitetura de referência, explicada a seguir. A escolha do cenário arquitetural mais adequado pode ser feita com base em recomendações de uso e com uma análise de critérios de qualidade, que é apresentada no decorrer e ao final desta seção, respectivamente.

5.2.1 Arquitetura de Referência

Uma arquitetura de referência serve como base para elaboração de arquiteturas de software. Conforme definiram Bass; Clements e Kazman (2003), uma arquitetura de referência consiste em componentes de software e os relacionamentos entre eles que implementam funcionalidades relativas às partes definidas no modelo de referência. Uma arquitetura de referência é originada a partir de um modelo de referência e um estilo arquitetural (Figura 5.1).

Figura 5.1. Relacionamento entre modelo de referência, estilo de arquitetura e arquitetura de referência implementando uma arquitetura de software

Fonte: (BASS; CLEMENTS; KAZMAN, 2003)

Murakami (2006) e Murakami *et al.* (2006) definiram uma arquitetura de referência (Figura 5.2) para sistemas para AP, utilizando como modelo de referência o MOSAICo e um estilo arquitetural em camadas e orientado a serviços. Essa arquitetura de referência define um barramento de serviços que podem ser acessados de maneira síncrona ou assíncrona. A camada de serviços agrícolas é fornecedora de funcionalidades que podem ser acessados por portais, sendo que esses serviços podem estar geograficamente distribuídos.

Figura 5.2. Visão geral da arquitetura de referência.

Fonte: (MURAKAMI, 2006)

A seguir são apresentadas três configurações da arquitetura, demonstradas como cenários arquiteturais, cada um com seu desenho específico, porém derivados da mesma arquitetura de referência. Todas elas atendem a construção de sistemas em camadas e orientadas a serviços, mas se diferenciam pela capacidade de se adequarem ou não a determinados requisitos como de desempenho e escalabilidade.

5.2.2 Cenário Centrado na *Web*

Neste cenário, um servidor *Web* é responsável por gerenciar as interações entre clientes *Web* e a lógica de negócio da aplicação, gerar o conteúdo dinamicamente (HTML, imagens, sons e vídeos), apresentar dados e coletar entradas dos usuários, controlar o fluxo de telas a ser apresentado para os usuários, manter estado da sessão *Web* dos usuários conectados, suportar múltiplos tipos de clientes e finalmente, implementar a lógica de negócio.

Uma aplicação *Web* roda dentro de um container *Web*, que gerencia o ciclo de vida de cada componente *Web*, despacha as requisições da aplicação para os componentes e fornece uma interface padrão para contextualizar dados, tais como estado da sessão e informações sobre a requisição corrente. Como essas interfaces são baseadas em especificações (JEE), os componentes *Web* são portáteis entre os diversos servidores *Web* ou servidores de aplicações JEE.

Os principais componentes da camada *Web* são os *Servlets* e as páginas JSP. Em uma aplicação típica, *Servlets* são utilizados para implementar a camada Controle do padrão MVC, JSP implementam a camada Visão e a camada Modelo é implementada por classes planas Java, utilizando um padrão de projeto JEE tal como *Business Object* (ALUR; CRUPI; MALKS, 2003). A integração pode ser implementada segundo o padrão de projeto *Data Access Object* (ALUR; CRUPI; MALKS, 2003)

Essa arquitetura é recomendada para aplicações de menor porte, que são baseadas na *Web*, porém não necessitam de soluções distribuídas e que podem perder em escalabilidade sem comprometer a qualidade da aplicação. Apresenta maior simplicidade de empacotamento e instalação e não exige servidores de aplicações, apenas um servidor *Web*. A Figura 5.3 (em UML) apresenta o desenho arquitetural para este cenário de aplicação. Em cinza estão as camadas físicas de

software, preconizadas pela plataforma JEE e em branco, as camadas lógicas de software, criadas para distribuir responsabilidades, minimizar o acoplamento e promover a componentização e reuso da aplicação.

Figura 5.3. Arquitetura de aplicação JEE baseado na Web

Neste cenário, a camada Cliente recebe as requisições dos usuários, feitas em navegadores *Web* (botões, *hiperlinks*) e envia para o controlador, geralmente um *Servlet*. Esse controlador é responsável em chamar a lógica de negócio da camada inferior e despachar o resultado em uma página JSP. Os serviços de persistência são executados na camada de Integração. A Figura 5.4 mostra um diagrama com a

seqüência de operações e os objetos envolvidos em uma aplicação para cadastrar um talhão de uma fazenda.

Figura 5.4. Diagrama de seqüência para inserir um Talhão

5.2.3 Cenário Multicamadas

Aplicações implementadas em uma arquitetura multicamadas possuem, além da camada *Web*, a camada EJB. Nesta arquitetura, a camada *Web* continua com o mesmo papel do cenário centrado na *Web*, com a diferença que as camadas Negócio e Integração passam a ser de responsabilidade do container EJB. O container EJB fornece serviços tais como de gerenciamento de transação, controle de concorrência e segurança. Além disso, componentes EJB representam uma solução organizada de maneira distribuída, conferindo maior escalabilidade para as aplicações.

A lógica de negócio é um conjunto de métodos utilizados para gerenciar uma função de negócio específica. O paradigma de orientação a objetos possibilita que desenvolvedores decomponham a lógica de negócio em um conjunto de componentes ou elementos chamados de objetos de negócio. Em outras palavras, objetos de negócio devem implementar a solução computacional do domínio do problema específico, tal como os requisitos da AP.

Entretanto, outros serviços de caráter genérico devem ser fornecidos pelos objetos de negócio. São serviços complexos por natureza e fogem do domínio

específico de uma aplicação. Esses serviços são automaticamente implementados em um container EJB. São eles:

Manter o estado: objetos de negócio devem manter seu estado durante a conversação entre cliente e aplicação e também de maneira persistente, em banco de dados ou outro meio.

Operar sobre dados compartilhados: devem operar sobre dados compartilhados, em que múltiplos usuários podem tentar atualizar a mesma informação. Um objeto de negócio deve manter a consistência desses dados.

Resolver transações: uma transação pode ser descrita como um conjunto de tarefas que necessita ser completada como uma unidade somente. Se uma tarefa falhar, as outras devem ser desfeitas ou canceladas e se todas forem realizadas com sucesso, a transação é completada.

Servir um grande número de clientes: objetos de negócio devem fornecer serviços a um grande número de clientes simultaneamente, dando a impressão a cada cliente que objetos de negócio dedicados estão disponíveis para servir suas requisições.

Prover disponibilidade: prover disponibilidade dos objetos, escondendo os erros do servidor dos clientes.

Prover acesso remoto: objetos de negócio devem disponibilizar serviços remotamente, sendo que eles necessitam de um ambiente de computação distribuída.

Controlar acesso: fornecer serviços de autorização e autenticação.

Facilitar reusabilidade: objetos de negócio devem ser reutilizados por diferentes componentes da mesma aplicação e até mesmo por outras aplicações.

Essa arquitetura é recomendada para aplicações que necessitam de maior escalabilidade e de soluções distribuídas, rodam em servidores de aplicação JEE e possibilitam inúmeras estratégias de implantação, conferindo maior desempenho e disponibilidade à aplicação. A Figura 5.5 mostra o desenho arquitetural para este cenário. Pode-se notar que, diferentemente do cenário anterior, as camadas físicas de software são três: a camada Cliente, container *Web* e por último, o container EJB.

Figura 5.5. Arquitetura de aplicação JEE multicamadas

5.2.4 Cenário com Cliente Stand-alone

Nessa arquitetura, os clientes são implementados em Java em três cenários diferentes: os clientes de serviços *Web*, clientes de serviços EJB, e clientes das camadas de dados.

Em qualquer um dos casos, a implementação de clientes Java exige muito mais esforço do que uma interface *Web*, porém fornecem outros benefícios. A utilização de componentes *Graphical User Interface* (GUI) possibilita a criação de

interfaces mais poderosas e clientes Java consomem menos largura de banda do que os navegadores *Web*. Por exemplo, se um cliente de serviço *Web* e um cliente Java requisitam uma mesma lista de talhões de uma fazenda, a resposta é maior para o navegador *Web*, pois contém também a lógica de apresentação, enquanto para o cliente Java apenas os dados já bastam. Além disso, clientes Java podem ser programados para menos conexões com o servidor, uma vez que o dado recuperado é processado localmente, enquanto que o cliente *Web* necessita de uma nova conexão a cada apresentação diferente do mesmo dado.

No caso de clientes de serviços *Web*, os programas Java devem simular um navegador *Web* para traduzir os comandos do usuário em requisições HTTP e interpretar as respostas HTTP do servidor *Web*. Isso deve ser programado e é uma questão chave na decisão da melhor arquitetura, pois envolve um esforço extra. A vantagem é que muitos servidores possuem *firewalls* e o protocolo HTTP é um dos únicos que não são barrados por essa barreira de segurança.

Em clientes de serviços EJB, a conectividade é realizada diretamente na camada EJB com Java RMI, eliminando a necessidade de simular um navegador *Web* na aplicação Cliente. Nesse cenário, a lógica de negócio é implementada no servidor, possibilitando o reuso da aplicação por outros tipos de clientes. Possibilita também que o processamento seja distribuído, aumentando a escalabilidade da aplicação. A desvantagem é que se a conexão for feita pela Internet e o servidor for protegido por *firewalls*, essa arquitetura deixa de ser uma opção. A utilização mais comum é em ambientes de Intranet, onde a segurança é menos restritiva.

Por último, o cenário de clientes de camadas de dados requer interfaces para acesso aos dados diretamente, tal como API JDBC. As desvantagens dessa arquitetura são inúmeras: por ser implementada na camada Cliente, fica impossibilitada a reutilização da lógica de negócio por outros clientes; não é possível computação distribuída, pois a aplicação executa *stand-alone* e a segurança dos dados fica comprometida, pois as interfaces de acesso aos dados são implementadas no cliente e podem ser violadas. Entretanto, essa arquitetura é recomendada para pequenas aplicações, que sirvam para tarefas de gerenciamento ou administração. Por exemplo, rotinas em lote que são executadas no período da noite para manutenção em banco de dados. A Figura 5.6 apresenta a arquitetura de software para este cenário. Fica evidente que a perda principal nesse tipo de arquitetura é a possibilidade de reutilização de componentes, uma vez que toda a

lógica está implementada na camada Cliente. Outros sistemas só acessariam a camada de dados, não sendo possível o acesso a serviços.

Figura 5.6. Arquitetura de aplicação JEE *stand-alone*

5.2.5 Avaliação dos Cenários Arquiteturais

Há na literatura alguns métodos de avaliação arquitetural, entre eles métodos baseados nos atributos de qualidade arquitetuais, tais como os métodos SAAM (*Software Architecture Analysis Method*) (KAZMAN *et al.* 1994) e ATAM (*Architecture Trade-off Analysis Method*) (KAZMAN; KLEN; CLEMENTS, 2000). Esses dois métodos avaliam na arquitetura de um sistema determinados requisitos arquitetuais, executando cenários que representam o comportamento esperado do software em relação a uma determinada característica de qualidade. Murakami (2006) indicou algumas propriedades arquitetuais importantes para definição de arquiteturas de software em sistemas para AP. O Quadro 5.1 apresenta cada uma dessas propriedades e qual a percepção de atendimento do cenário arquitetural no contexto de cada uma das propriedades. Adicionalmente, é definido um atributo que indica a aderência da arquitetura de software à arquitetura de referência.

Arquitetura			<i>Stand-alone</i>
Atributo de qualidade	Multicamadas	Web	(Acesso à camada de dados)
Escalabilidade	Fornece alto grau de escalabilidade, pois permite diferentes configurações de servidores e permite distribuição para suportar maior número de usuários, por exemplo. A divisão em camadas proporciona baixo grau de acoplamento, permitindo o crescimento da aplicação minimizando efeitos colaterais indesejados.	Perde um pouco da escalabilidade, fica dependente do poder computacional do hardware. Não é possível distribuir os componentes.	Perde toda a escalabilidade, pois fica totalmente dependente do poder computacional da camada Cliente. Possui acoplamento entre componentes e não permite distribuição.
Simplicidade	Alto grau de simplicidade, à medida que a descentralização dos serviços pode garantir componentes menos complexos e de fácil entendimento.	Alto grau de simplicidade, pois a descentralização dos componentes depende somente das camadas lógicas do software.	É possível construir componentes de fácil entendimento na camada Cliente.
Extensibilidade	Alto grau de extensibilidade, pois é possível implantar funcionalidades em servidores diferentes sem ao menos impactar a execução de uma aplicação.	A componentização garantida pelas camadas lógicas garante um bom grau de extensibilidade, uma vez que novos componentes podem ser acoplados sem efeitos colaterais nos demais. Diferentemente do cenário	A componentização garantida pelas camadas lógicas garante um bom grau de extensibilidade, uma vez que novos componentes podem ser acoplados sem efeitos colaterais nos demais. Diferentemente do cenário

Arquitetura			Stand-alone
Atributo de qualidade	Multicamadas	Web	(Acesso à camada de dados)
		multicamadas, não é possível estender a aplicação sem afetar a disponibilidade da mesma.	multicamadas, não é possível estender a aplicação sem afetar a disponibilidade da mesma.
Configurabilidade	Altamente configurável, podendo aceitar níveis de segurança diferenciados nas camadas, formas de persistência e tratamento de requisições sem alterações em código.	Altamente configurável, podendo aceitar níveis de segurança diferenciados nas camadas, formas de persistência e tratamento de requisições sem alterações em código.	A arquitetura <i>stand-alone</i> perde as características de customização de servidor da plataforma, pois roda no lado Cliente.
Customizabilidade	O padrão de projeto Template Method (GAMMA; VLISSIDES; JOHNSON, 1994) permite que a implementação de certos métodos seja feita pelos Clientes. A utilização das interfaces Java também permite reimplementações.	Idem	Idem
Reusabilidade	Alto grau de reusabilidade, tanto pela divisão em camadas lógicas quanto pelas camadas físicas de software.	Alto grau de reusabilidade, porém o acesso aos serviços somente poderá ser realizado pela <i>Web</i> .	Não permite reusabilidade de serviços.
Portabilidade	Característica inerente da plataforma J2EE e a linguagem Java	Idem	Idem

Arquitetura			Stand-alone
Atributo de qualidade	Multicamadas	Web	(Acesso à camada de dados)
Confiabilidade	A capacidade de distribuição pode permitir que o tratamento a uma falha de hardware direcione a chamada do cliente para outro servidor.	A confiabilidade depende do servidor de aplicações centralizado.	Não é garantido nenhum nível de confiabilidade além da qualidade da aplicação cliente.
Aderência à arquitetura de referência	Total. Permite a construção de sistemas em camadas lógicas e também a disponibilização de serviços com <i>web services</i> , interfaces remotas de EJB e serviços de mensageria (<i>message-driven beans</i>)	Total. Permite a construção de sistemas em camadas lógicas e também a disponibilização de serviços com <i>web services</i> .	Parcial Permite a construção de sistemas em camadas lógicas, porém não permite a disponibilização de serviços a outros clientes.

Quadro 5.1. Avaliação das propriedades de qualidade dos cenários de arquitetura.

5.3 Padrões de Análise e Projeto

Como parte da arquitetura de software, foi criado um catálogo de padrões de análise e padrões de projeto que se propõe a auxiliar a construção de componentes para sistemas voltados para AP. Esses padrões foram organizados segundo as camadas lógicas adotadas para essa proposta arquitetural. Desse modo, não importa qual seja o cenário arquitetural selecionado, os padrões podem sempre ser utilizados na construção dos componentes.

Para modelar os padrões de análise, foram definidos alguns estereótipos da linguagem UML, conforme apresentado a seguir.

5.3.1 Extensão da UML

O MOSAICo, definido por Saraiva (1998) e Saraiva, Massola e Cugnasca (1998), é um modelo que possui um conjunto de pacotes de subsistemas que capturam os principais requisitos de sistemas de informações para AP. Os subsistemas Dados de Campo e Geração de Mapas, por conterem informações espaço-temporais foram estendidos para suportar tais características.

A UML foi utilizada para o modelamento dessas extensões. Apesar do metamodelo da linguagem não incluir representação semântica necessária para modelar requisitos espaciais e temporais, a linguagem prevê um mecanismo de extensão chamado de estereótipo (BOOCH, RUMBAUGH; JACOBSON, 2005). Esse mecanismo permite definir a essência da semântica da UML e fornece ao projetista a liberdade para ajustar a linguagem às suas necessidades. Dessa maneira, definimos alguns estereótipos para a UML. São eles:

Temporais: definem conceitos temporais.

<<tt>>: Tempo de Transação

<<vt>>: Tempo de Validade

Espaciais: definem elementos espacialmente modelados.

<<point>>: Ponto

<<polygon>>: Polígonos

<<line>>: Linhas

<<cells>>: Grade de células

Espaço-Temporais: definem elementos espacialmente modelados e com conceitos temporais, ou seja, todas as combinações possíveis entre os dois tipos (Espacial e Temporal).

<<point - tt>>: Ponto e Tempo de Transação

<<point - vt>>: Ponto e Tempo de Validade

<<polygon - tt>>: Polígono e Tempo de Transação

<<polygon - vt>>: Polígono e Tempo de Validade

<<line - tt>>: Linha e Tempo de Transação

<<line - vt>>: Linha e Tempo de Validade

<<cells - tt>>: Grade de células e Tempo de Transação

<<cells - vt>>: Grade de células e Tempo de Validade

Esses estereótipos podem ser utilizados para estender as características dos conceitos de classe, atributos e relacionamentos. A Figura 5.7 apresenta um exemplo de utilização dessa semântica.

Figura 5.7. Estereótipos espaço-temporais

Uma classe é criada para representar uma classificação de entidades do mundo real e uma entidade do mundo real pode possuir uma forma e localização. Essas entidades são modeladas com estereótipos de geometrias, tais como polígonos, linhas ou pontos. Vale lembrar que o modelo resultante de uma realidade depende da abstração aplicada. No mesmo exemplo dado pela Figura 5.7, na qual uma cidade é modelada como um polígono, poderia ser considerada uma geometria do tipo ponto, caso o contexto geral fosse mapa do Brasil.

Entidades geométricas que podem ter suas propriedades geométricas alteradas no tempo podem ser modeladas como temporais. É o caso da classe *CameraObservação*. Uma câmera de observação é considerada um ponto em um segmento de via, que não pode variar em sua forma, mas sim em sua localização. Dessa maneira, para a realidade modelada, as câmeras de observação são móveis e posicionadas de acordo com alguma conveniência. O tipo temporal dessa classe é tempo de validade, o que indica que o operador do sistema poderá cadastrar o posicionamento da câmera para o tempo presente e também para o futuro, além de manter o histórico das posições de uma câmera na via.

As propriedades de uma classe também podem ser consideradas temporais. Por exemplo, na classe *Via*, a propriedade *nome* é modelada com o tipo temporal

Tempo de Transação. Isso significa que o sistema deverá manter um histórico de alterações para o nome das vias da cidade.

5.3.2 O Catálogo de Padrões de Análise e Padrões de Projeto

Os padrões de análise para aplicações para AP foram construídos com auxílio dos estereótipos definidos. Tanto os padrões de análise como os padrões de projeto foram divididos nas camadas lógicas do sistema (apresentação, negócio e integração) para organização do catálogo. A Figura 5.8 mostra o catálogo de padrões de análise e projeto para desenvolvimento de aplicações espaço-temporais.

Os padrões com estereótipos <<j2ee>> fazem parte do catálogo J2EE (ALUR; CRUPI; MALKS, 2003). Os padrões com estereótipos <<espaço-temporal>>, <<espacial>> e <<temporal>> são partes deste trabalho e serão detalhados mais à frente.

O MOSAICo pode ser dividido em vários padrões de análise, que seriam utilizados para compor outras aplicações e estendidos conforme suas necessidades. Entretanto, o foco deste trabalho foi o desenvolvimento dos padrões que envolvem aspectos espaço-temporais. Sendo assim, os padrões de análise são extensões do modelo MOSAICo para permitir suporte aos dados e operações espaciais e temporais.

Figura 5.8. Catálogo de padrões de análise e projeto espaço-temporais

O modelo para descrever os padrões de análise será o mesmo definido por Meszaros e Doble (2006):

- **Problema:** fornece uma declaração sucinta do problema que necessita ser resolvido, normalmente é uma pergunta.
- **Contexto:** descreve o contexto no qual o problema foi identificado e para o qual a solução foi proposta.
- **Forças:** conjunto de restrições que foram consideradas na elaboração da solução do problema.

- **Solução:** descreve a solução dada e ilustra com diagramas de classes e diagramas de seqüência.

O modelo para descrever os padrões de projeto será o mesmo utilizado para documentação dos padrões de análise, porém com adição do campo “exemplo”, que apresenta exemplos para sistemas reais através de diagramas de classes e diagramas de seqüência e o campo “padrões relacionados”, que lista os padrões de projeto relacionados com a solução apresentada. Esses campos foram utilizados, respectivamente, na definição dos padrões *GoF* (GAMMA; VLISSIDES; JOHNSON, 1994) (GAMMA *et al.*, 1993) e J2EE (ALUR; CRUPI; MALKS, 2003).

5.3.2.1 Padrão de Análise: Talhão e Amostras

Problema: Como modelar os talhões de uma fazenda e as amostras que estão contidas nesses talhões?

Contexto: Um talhão é a menor unidade de manejo de uma fazenda, sendo que estes devem ter seus limites geográficos estabelecidos. Um novo mapeamento das áreas de cultivo pode ser necessário, causando a reconfiguração das áreas da fazenda que formam os talhões. Desse modo, é necessário que um histórico da área dos talhões esteja disponível. As amostragens realizadas nos talhões podem ser de vários tipos, tais como de solo, doenças, culturas, ervas daninhas e outros (SARAIVA, 1998). Cada amostragem possui um período de validade, determinada pelo usuário. Para uma amostra pode haver mais de um atributo associado, como por exemplo, em uma amostra de solo teremos o teor de potássio e o teor de calcário.

Forças:

- Ao longo do tempo, várias amostragens podem ser realizadas para um mesmo talhão.
- Uma amostragem é um conjunto de amostras

Solução: A Figura 5.9 mostra o diagrama de classes para a solução. O talhão é um fenômeno espacial, modelado como um polígono (área). Além disso, o histórico das áreas de um talhão é representado. Cada talhão pode ter amostragens, sendo que as mesmas são constituídas por um conjunto de amostras (pontos). O período de validade foi resolvido com a restrição temporal “Tempo de Validade” para a classe

Amostragem. Dessa maneira, será possível saber as amostras válidas no presente e futuro e também qual foi validade passada de uma amostragem.

Figura 5.9. Padrão de análise Talhão e Amostras

5.3.2.2 Padrão de Análise: Mapas de Talhões

Problema: Como modelar os diversos mapas, com uma camada primária e várias camadas secundárias, permitindo que os atributos das amostras e os dados derivados das análises sejam apresentados visualmente?

Contexto: Os dados coletados pelas amostragens devem ser armazenados em grades de células que representa a área do talhão. Cada célula deve conter um valor para um determinado atributo coletado. A primeira camada de dados obtida do processo de coleta é chamada de mapa primário. Os dados primários são obtidos da correção dos erros e distorções originadas no processo de coleta. Murakami (2006) apresenta em seu trabalho a implementação de um algoritmo de correção de dados coletados em campo. Os dados primários podem conter número insuficiente de amostras para permitir as análises necessárias. Esses dados são então corrigidos com métodos como interpolação e então uma segunda camada de dados é gerada, composta pelos mapas básicos. A partir de um mapa básico, camadas de dados adicionais são geradas através da aplicação de métodos de análise, obtendo-se, por

exemplo, os mapas de aplicação. Alternativamente, o operador pode simular os modelos existentes, gerando os mapas de simulação.

Forças:

- Um talhão pode conter zero ou mais mapas
- Um mapa não contém restrição temporal própria, mas é parte de um talhão (relacionamento de agregação) que possui a restrição temporal “tempo de transação”.

Solução: A Figura 5.10 mostra a solução para o problema descrito. Um talhão pode conter vários mapas, permitindo que várias camadas de dados sejam associadas ao talhão. Além disso, os mapas podem ser dos tipos primário, básico, secundário e simulado. Os mapas são modelados como uma matriz ou grade de células. Desse modo, é possível associar valores a cada célula da grade, sendo que cada valor representa um atributo de uma amostra.

Figura 5.10. Padrão de análise Mapas de talhões

5.3.2.3 Padrão de Projeto: Apresentar Objetos Espaciais

Problema: Como apresentar dados espaciais em interface baseada em navegadores *Web* (*browser*)?

Contexto: A apresentação de dados convencionais em navegadores *Web* é bastante explorada e solucionada por *frameworks* de apresentação, tais como Struts (STRUTS, 2006) e Java Server Faces (JSF, 2006). Por sua vez, dados espaciais podem ser representados com gráficos matriciais, nos formatos GIF ou JPEG. Os navegadores *Web* já são preparados para essa representação da informação espacial, onde para cada ponto (*pixel*) da imagem uma cor é associada. Já para gráficos vetoriais, tanto os *frameworks* de apresentação quanto os navegadores ainda não estão preparados para apresentar tais gráficos. Gráficos vetoriais são muito úteis na representação geográfica, pois permitem, por meio de instruções, a determinação de um valor para cada ponto (*pixel*) no espaço. Gráficos vetoriais permitem maior dinamismo (por exemplo, cores e escalas), pois podem ser mais facilmente manipulados.

Forças:

- A camada de visão deve ser baseada em *templates*
- Evitar lógica de apresentação na camada de visão
- Permitir regras de apresentação para formatar a visão conforme regras de negócio da aplicação

Solução: Adotar a abordagem *ViewHelper*, na qual são construídas classes que embutem a lógica da apresentação, desacoplando da camada visão. Devem ser criados modelos (*templates*) para permitir a transformação dos dados espaciais em um formato compreensível pelo navegador, tal como *Scalable Vector Graphics* (SVG, 2006). Os dados espaciais (PAML ou objetos MOSAICo) serão transformados em formatos SVG com base nos *templates*. O padrão *Strategy* (GAMMA; VLISSIDES; JOHNSON, 1994) é utilizado para representar as diferentes regras que podem ser aplicadas a uma transformação, como por exemplo padrões de cores e escala. A solução é ilustrada pela Figura 5.11 e a Figura 5.12.

Figura 5.11. Diagrama de classes: Apresentar Objetos Espaciais

Figura 5.12. Diagrama de seqüência: Apresentar Objetos Espaciais

Exemplo:

O exemplo de implementação deste padrão é apresentado no Capítulo 6. Estudo de Caso.

Padrões Relacionados:

- **<<j2ee>> View Helper** (ALUR; CRUPI; MALKS, 2003)
- **<<GoF>> Strategy** (GAMMA; VLISSIDES; JOHNSON, 1994)

5.3.2.4 Padrão de Projeto: Atributos Temporais

Problema: Como implementar atributos de classes que devem possuir aspectos temporais?

Contexto: Classes convencionais ou espaço-temporais podem conter atributos que necessitam de dados históricos, tanto do tipo tempo de transação como tempo de validade. O valor vigente na data corrente deve ser obtido de forma transparente pelo programador, sem que o mesmo tenha que conhecer a estrutura de dados do atributo para obter o valor.

Forças:

- O período de validade deve ser informado pelo usuário do sistema e ser visível para o Cliente.
- Um objeto que representa um atributo temporal possui somente um estado do atributo, que por padrão é o último.
- O cliente da classe que possui o atributo temporal deve obter o valor de maneira transparente, sem preocupações com datas.

Solução: Criar uma superclasse representando o tipo temporal (Tempo de Transação e Tempo de Validade). Nessas superclasses estarão implementados os atributos e serviços específicos para o tipo temporal. Criar subclasses que representam os tipos primitivos de dados (*float*, *int*, *string*), que conterão os valores primitivos associados ao tempo da superclasse. A classe base deve conter uma referência para a subclasse que representa o tipo primitivo e deve utilizar o método de recuperação para obter o valor do atributo. A solução é ilustrada pela Figura 5.13 e a Figura 5.14.

Figura 5.13. Diagrama de classe: Atributos Temporais

Figura 5.14. Diagrama de seqüência: Atributos Temporais

Exemplo: A classe *Fazenda* definida no modelo MOSAICo possui o atributo *nome*. Naturalmente, o nome de uma fazenda pode ser alterado, e nesse caso pode ser interessante ter acesso a esses nomes passados. O tipo temporal *Tempo de Transação*, por definição, armazena dados históricos, porém o valor válido é o determinado na última alteração, sendo possível recuperar os valores de transações anteriores. O atributo *nome* da classe *Fazenda* é transformado em uma associação com a classe *StringTempoTransação*, que possui o nome da fazenda atual ou um nome anterior, dependendo da consulta que foi realizada. A Figura 5.15 e a Figura 5.16 ilustram o exemplo.

Figura 5.15. Diagrama de classes: exemplo de atributos temporal

Figura 5.16. Diagrama de seqüência: exemplo de atributos temporal

Padrões relacionados:

- `<<espaço-temporal>>` Objetos de Negócio Espaço-Temporais

5.3.2.5 Padrão de Projeto: Relacionamentos Temporais

Problema: Como implementar relacionamentos entre classes que devem possuir aspectos temporais?

Contexto: Classes convencionais ou espaço-temporais podem conter relacionamentos com outras classes convencionais ou espaço-temporais que necessitem de dados históricos, tanto do tipo tempo de transação como tempo de validade.

Forças:

- O período de validade deve ser informado pelo usuário do sistema e ser visível para o Cliente.
- A estrutura representa a informação para um período recuperado, podendo ser passado, presente ou futuro, no caso de tempo de validade.

Solução: Criar uma superclasse representando o tipo temporal (Tempo de Transação e Tempo de Validade). Nessas superclasses estarão implementados os atributos e serviços específicos para o tipo temporal. Criar subclasses para os relacionamentos temporais que conterà o relacionamento de um objeto com outro objeto, podendo ser de multiplicidade 1 ou muitos, associados ao tempo da superclasse. A classe base deve conter uma referência para a subclasse que representa o relacionamento temporal e deve utilizar o método de recuperação para obter o(s) objeto(s) relacionado(s). A Figura 5.17 e a Figura 5.18 ilustram a solução.

Figura 5.17. Diagrama de classe: Relacionamentos Temporais

Figura 5.18. Diagrama de seqüência: Relacionamentos Temporais

Exemplo: A classe Fazenda definida no modelo MOSAICo possui um relacionamento de propriedade com a classe Proprietário. A fazenda é de um proprietário por um período de tempo, podendo esse período abranger tempos passados, presente e até mesmo futuro. O tipo temporal Tempo de Validade é o mais indicado para esse caso, pois representa o período válido da informação, neste caso o relacionamento de propriedade. A Figura 5.19 e a Figura 5.20 ilustram o exemplo.

Figura 5.19. Diagrama de classe: exemplo de relacionamento temporal.

Figura 5.20. Diagrama de sequência: exemplo de relacionamento temporal.

Padrões relacionados:

- <<espaço-temporal>> Objetos de Negócio Espaço-Temporais

5.3.2.6 Padrão de Projeto: Objetos de Negócio Espaço-Temporais

Problema: Como adicionar funcionalidade espaço-temporal a objetos de negócio?

Contexto: Objetos de negócio considerados espaciais, temporais ou espaço-temporais no modelo de análise devem incorporar a funcionalidade espaço-temporal sem que o projetista necessite implementar esse comportamento na classe de negócio. O mecanismo de herança é o mais apropriado para estender o estado e o comportamento de classes. No entanto, para classes que são espaço-temporais, a classe filho deverá herdar a funcionalidade espacial e temporal, caracterizando uma herança múltipla. Como a herança múltipla não é possível de ser implementada na linguagem Java, não é possível definir uma classe espaço-temporal somente com herança.

Forças:

- A funcionalidade espacial e temporal deve ser transparente para a classe de negócio.

Solução: Criar uma classe que agrega a funcionalidade espacial e a funcionalidade temporal. Essa classe é uma subclasse de uma Geometria, herdando assim toda a funcionalidade espacial. Ao mesmo tempo, essa classe possui uma agregação com o tipo temporal correspondente. A classe deverá encapsular a funcionalidade temporal, realizando chamadas aos métodos da classe temporal para que a classe de negócio tenha disponíveis as operações espaciais e temporais. A Figura 5.21 apresenta o diagrama de classes com a solução proposta.

Figura 5.21. Diagrama de classes: Objetos de negócio espaço-temporais

Exemplo: A classe de negócio Talhão pode ser modelada como um polígono com histórico de tempo de transação

Figura 5.22. Diagrama de classes: Exemplo Objetos de negócio espaço-temporais

Padrões Relacionados:

- <<espaço-temporal>> Tipos Temporais
- <<espaço-temporal>> Relacionamentos Temporais

5.3.2.7 Padrão de Projeto: Persistir Objetos Espaço-Temporais

Problema: Como persistir objetos espaço-temporais de maneira que a persistência e a recuperação dos dados espaciais e temporais dos objetos sejam transparentes para desenvolvedores?

Contexto: Os *frameworks* tradicionais de persistência não são preparados para manipular dados espaciais e temporais separadamente, de maneira que essas operações sejam transparentes para o cliente do *framework*. A estrutura de armazenamento das geometrias deve ser única, seja em banco de dados relacionais ou em outros meios persistentes, e a lógica de manipulação desses dados deve ser embutida em classes específicas para tal. A camada de persistência deve ser preparada para várias tecnologias de persistência, diferentes gerenciadores de banco de dados ou GML.

Forças:

- Evitar lógica de persistência nos objetos de negócio
- Não utilizar EJB (*entity bean*).
- Tornar transparente para a aplicação a persistência dos dados espaciais e temporais
- Prover interfaces para implementações em diferentes tecnologias de persistência. As particularidades de cada tecnologia devem ser indiferentes para a aplicação.

Solução: Criar um gerenciador de persistência que centralizará todas as operações CRUD para os objetos de negócio, convencionais ou espaço-temporais, projetado conforme o padrão de projeto *Façade* (GAMMA; VLISSIDES; JOHNSON, 1994). As configurações para cada tecnologia de persistência, as regras de mapeamento objeto-relacional bem como regras de gerenciamento de transação para servidores de diferentes aplicações devem estar definidas previamente. Os estados dos objetos

persistentes (novo, antigo, não modificado, modificado e removido) devem ser controlados pela camada, possibilitando que as operações CRUD sejam realizadas. O padrão de projeto *State* (GAMMA; VLISSIDES; JOHNSON, 1994) foi utilizado para implementar esse controle. Por final, a lógica de persistência para cada tecnologia deve ser implementada separadamente, de maneira a tornar a camada facilmente extensível a outras tecnologias. Os dados espaciais e temporais devem ser tratados de maneira genérica, sendo que as operações CRUD são responsáveis em manipular esses dados. O padrão DAO (ALUR; CRUPI; MALKS, 2003) foi utilizado para implementar os objetos de persistência. A Figura 5.23 e a Figura 5.24 ilustram a solução dada.

Figura 5.23. Diagrama de classes: Persistir objetos espaço-temporais

Figura 5.24. Diagrama de sequência: Persistir objetos espaço-temporais

Exemplo: Mecanismo de persistência para o sistema gerenciador de banco de dados MySQL e para arquivos GML do modelo de objetos MOSAICo. A Figura 5.25 apresenta o diagrama de classe com o exemplo dado.

Figura 5.25. Diagrama de classe: Exemplo de persistir objetos espaço-temporais

Padrões Relacionados:

- <<GoF>> *Factory method* (GAMMA; VLISSIDES; JOHNSON, 1994)

- <<GoF>> *Facade* (GAMMA; VLISSIDES; JOHNSON, 1994)
- <<GoF>> *State* (GAMMA; VLISSIDES; JOHNSON, 1994)
- <<j2ee>> DAO (ALUR; CRUPI; MALKS, 2003)
- <<espaço-temporal>> Consultar Objetos Espaço-Temporais

5.3.2.8 Padrão de Projeto: Consultar Objetos Espaço-Temporais

Problema: Como realizar consultas com critérios ou funções espaço-temporais?

Contexto: Alguns sistemas gerenciadores de banco de dados da indústria já possuem suporte a dados espaciais. Esses sistemas gerenciadores são capazes de armazenar os dados espaciais e realizar consultas onde critérios podem possuir funções espaciais, tais como o *PostGIS*, *Oracle Spatial* (QUEIROZ; FERREIRA, 2005) e a versão 5.0 do SGBD *MySQL*. Contudo, muitos sistemas não possuem tais características. Desse modo, passa a ser responsabilidade das aplicações implementar os operadores espaciais.

Forças:

- A execução dos operadores espaciais e temporais deve ser realizada pelo componente ou delegada para o SGBD de maneira transparente para a aplicação.

Solução: Criar um comando para executar consultas ao sistema gerenciador de banco de dados e uma hierarquia de classes para implementar o processamento das consultas para cada SGBD alvo. As funções temporais são traduzidas para restrições de data pela classe *TemporalFunction*. Enquanto os operadores temporais são sempre traduzidos para cláusulas SQL e resolvidos pelo SGBD, os operadores espaciais podem ser resolvidos pela própria aplicação, tendo como entrada o resultado da consulta ao banco de dados. O SQL padrão ANSI é implementado na superclasse *ANSIStatement* e as subclasses reimplementam, caso necessário, as especificidades do SGBD alvo. A Figura 5.26 e a Figura 5.27 mostram os diagramas de classe e seqüência da solução.

Figura 5.26. Diagrama de classe: Consultar Objetos Espaço-Temporais

Figura 5.27. Diagrama de sequência: Consultar Objetos Espaço-Temporais

Padrões Relacionados:

- <<GoF>> *Command* (GAMMA; VLISSIDES; JOHNSON, 1994)

5.3.2.9 Padrão de Projeto: Montar Objetos Espaço-Temporais Complexos

Problema: Como montar objetos espaciais complexos, formados por geometrias heterogêneas tais como polígonos, linhas e pontos?

Contexto: Objetos complexos, que envolvem informações espaciais e temporais, podem ser difíceis de serem construídos, envolvendo algoritmos também complexos. Se todo o algoritmo for implementado em um mesmo lugar, a reusabilidade e manutenibilidade da operação estarão comprometidas. Além disso, podem existir diferentes fontes e / ou formatos de informações para montagem dos objetos, mas é desejável que isso seja transparente para o Cliente do montador.

Forças:

- Dividir a responsabilidade de montagem dos objetos espaço-temporais em partes
- Considerar que pode haver mais de uma implementação para montar o mesmo objeto espaço-temporal complexo.

Solução: Separar a construção do objeto complexo de sua representação, delegando a complexidade de construção para outras classes e separando em partes menores, facilitando o entendimento, reuso e manutenibilidade do componente. Criar uma superclasse abstrata para o construtor, permitindo que exista diferentes implementações do construtor para um mesmo objeto. O resultado (objeto construído) deve ser armazenado no próprio construtor para que o Cliente acesse diretamente. A Figura 5.28 e a Figura 5.29 ilustram a solução.

Figura 5.28. Diagrama de classe: Montar objetos espaço-temporais complexos

Figura 5.29. Diagrama de sequência: Montar objetos espaço-temporais complexos

Exemplo: Um talhão é um objeto que pode ser considerado complexo, pois ele próprio é uma geometria, composta por outras geometrias (amostras) e possui aspectos temporais. A montagem do talhão pode ser feita a partir de fontes diversas, tais como arquivos GML, arquivos texto provenientes de máquinas agrícolas e também de interfaces de banco de dados. A implementação deve prover uma maneira única de construir esses objetos, seja qual for a fonte da informação ou a camada da aplicação do cliente do montador. A Figura 5.30 e a 5.31 apresentam a solução.

Figura 5.30. Diagrama de classe: Exemplo de montar talhão

Figura 5.31. Diagrama de seqüência: Exemplo de montar talhão

Padrões Relacionados:

- <<GoF>> *Builder* (GAMMA; VLISSIDES; JOHNSON, 1994)

5.4 Especificação das Camadas Lógicas

As camadas lógicas dos cenários arquiteturais propostos anteriormente são baseadas sempre em três camadas lógicas: camada de apresentação, camada de negócio e camada de integração / persistência. Cada camada fornece serviços para a camada imediatamente acima e utiliza serviços da camada imediatamente abaixo.

Essa abordagem proporciona sistemas mais modulares e flexíveis e divide a funcionalidade de manutenção e apresentação dos dados para minimizar o grau de acoplamento entre os objetos. Sendo assim, a principal característica de um sistema em camadas é a separação da lógica de negócio, o processamento do servidor e a interface com o usuário em componentes distintos. Essa separação permite que cada componente seja facilmente reutilizado e intercambiado com outros componentes.

A grande vantagem de sistemas em camadas é sua organização, que permite um código extremamente modular. A modularização permite melhor reutilização do código e torna a manutenção do sistema mais fácil e localizada, que implica no aumento da produtividade de desenvolvimento. Além disso, um componente pode ser alterado ou substituído sem causar maiores impactos no projeto, devido à

independência de cada camada. A seguir, são apresentados alguns detalhes de quais tecnologias podem ser utilizadas e quais funcionalidades cada camada deve implementar.

5.4.1 Camada de Apresentação

A Camada de Apresentação apresenta os dados dos objetos de negócio na interface com os usuários do sistema. Ela acessa dados da camada negócio e contém a lógica de apresentação, que especifica como os dados serão apresentados. A camada define a lógica relacionada à interface do sistema com o usuário. A plataforma JEE permite dois tipos de interfaces: a interface baseada em HTML, que tem o processamento suportado pelo servidor de aplicações, e a interface gráfica (GUI), cujo processamento é realizado no cliente da aplicação.

Há vários padrões de projeto (ALUR; CRUPI; MALKS, 2003) disponíveis para se construir uma camada de apresentação versátil e totalmente desacoplada das regras de negócio do sistema. Algumas implementações desses padrões de projetos, os chamados *frameworks* de apresentação, estão disponíveis para desenvolvedores de aplicações JEE. Alguns deles, tal como *Struts* (STRUTS, 2006) possuem grande aceitação no mercado e contam com um sem-número de conhecedores. Tais *frameworks* são implementações da tecnologia *Servlets* e JSP e baseados no padrão de projeto *Model-View-Controller* – MVC (MVC, 2006). *Servlets* e JSP são processadas pelo servidor de aplicações e geram, como saída, o código HTML para a interface com o usuário.

Com a plataforma JEE, é possível também construir aplicações com maior poder gráfico, utilizando a API *Swing* do Java. Essa API fornece componentes de visualização de dados mais poderosos e, neste caso, o processamento da lógica de apresentação é realizado pela máquina Cliente.

No caso de apresentação *Web*, uma tecnologia emergente para apresentação de dados geográficos é o *Scalable Vector Graphics* (SVG, 2006). SVG é uma linguagem para descrever gráficos vetoriais bidimensionais em XML e permite três tipos de objetos gráficos: formas gráficas vetoriais (por exemplo, polígonos), imagens e texto. Mais detalhes sobre essa tecnologia e como utilizá-la para apresentar objetos espaciais em navegadores *Web* serão apresentados no Capítulo 6. Estudo de Caso.

5.4.2 Camada de Negócio

A Camada de Negócio representa os dados de negócio e a lógica de negócio que acessa e modificam os dados de negócio. Essa camada representa a funcionalidade do mundo real, traduzindo um problema concreto para uma implementação computacional. A separação da lógica de negócio da lógica de apresentação minimiza os impactos causados por mudanças, aumenta a manutenibilidade do sistema, possibilita a independência de tecnologia utilizada na camada Cliente e permite a separação de papéis de desenvolvedores (especialização por camadas).

Essa camada é acessada pela camada de apresentação. A separação dos requisitos de negócio em uma camada independente da apresentação proporciona flexibilidade na utilização de tecnologias distintas na visualização dos dados e comunicação entre sistemas. A transferência de dados para a camada de apresentação é feita por objetos próprios. Esses objetos podem implementar padrões de projeto tais como *Transfer Objects* (ALUR; CRUPI; MALKS, 2003).

Se por um lado, os padrões de análise mostraram as necessidades espaciais e temporais para o MOSAICo em termos de análise, para o modelo de projeto é necessário definir uma estrutura de objetos que suporte as informações e serviços espaço-temporais para utilização no MOSAICo. A Figura 5.32 apresenta a hierarquia de objetos espaciais projetada para estender o MOSAICo. Essa hierarquia foi baseada no modelo conceitual GeoFrame-T (LISBOA FILHO, 2000; Rocha, 2001).

Figura 5.32. Modelo de objetos espaciais

As representações conceituais de temas e regiões geográficas definidas no GeoFrame não foram consideradas no projeto das classes espaciais do MOSAICo, pois essas classes têm caráter genérico e servem para representar quaisquer temas relacionados a sistemas de informações geográficas, o que foge ao objetivo de nosso trabalho. A classe `ObjNaoGeografico` do GeoFrame é uma representação conceitual, não sendo necessário o mapeamento dessa classe no modelo de projeto.

Para representação de campo, a grade de células será suficiente para representação de atributos contínuos no MOSAICo. A representação vetorial (visão de objetos) é feita pela classe abstrata *Geometry*. A infra-estrutura representa somente as geometrias vetoriais, que são todas elas construídas a partir de coordenadas bi-dimensionais (coordenada).

A superclasse `Geometry` contém as operações sobre os dados vetoriais, tais como consultas topológicas, métricas e geração de novas geometrias a partir de operações de conjunto. Essas operações são automaticamente herdadas por todas

as geometrias (pontos, linhas e polígonos) e reimplementadas para suportar as especificidades de cada geometria. O Quadro 5.2 apresenta uma descrição das funções.

Relações espaciais	
Contém	Retorna verdadeiro se uma geometria contém uma outra geometria dentro de sua área, sem a geometria menor toque os limites da geometria maior. Caso contrário, retorna falso. Exemplo: <code>talhao.contem(amostraPraga)</code>
Dentro	Retorna verdadeiro se uma geometria está totalmente dentro de uma outra geometria sem tocar em seus limites. Caso contrário, retorna falso. Exemplo: <code>amostraSolo.dentro(fazendaOuroVermelho)</code>
Cobre	Retorna verdadeiro se uma geometria envolve uma outra geometria, sendo que o limite externo da geometria menor deve tocar o limite interno da geometria maior. Caso contrário, retorna falso. Exemplo: <code>fazenda.cobre(talhao1)</code>
Sobrepoe	Retorna verdadeiro se uma geometria sobrepõe a área de outra geometria. Caso contrário, retorna falso. Exemplo: <code>talhao1.sobrepoe(talhao2)</code>
Toca	Retorna verdadeiro se o limite externo de uma geometria toca o limite externo de outra geometria. Caso contrário, retorna falso. Exemplo: <code>talhao1.toca(talhao2)</code>
Desloca	Retorna verdadeiro se uma geometria desloca outra geometria. Exemplo: <code>talhao1.desloca(talhao2)</code>
Igual	Retorna verdadeiro se uma geometria possui o mesmo formato e área de uma outra geometria. Exemplo: <code>talhao1.igual(talhao2)</code>
Métricas	
Área	Calcula e retorna a área de um polígono. Exemplo: <code>area = fazenda1.area()</code>
Distancia	Calcula e retorna a distância entre duas geometrias. Exemplo: <code>distancia = amostra1.distancia(amostra2)</code>

Conjunto	
Intersecção	Retorna o polígono resultante da operação de intersecção entre dois polígonos. Exemplo: <code>parteAlagada = talhao.interseccao(lagoa)</code>
União	Retorna o polígono resultante da operação de união entre dois polígonos. Exemplo: <code>talhao3 = talhao1.uniao(talhao2)</code>
Diferença	Retorna o polígono resultante da operação de diferença entre dois polígonos. Exemplo: <code>parteNaoAlagada = talhao.diferenca(lagoa)</code>

Quadro 5.2. Operações geográficas

A hierarquia de objetos espaciais foi então utilizada para estender o MOSAICo com aspectos espaço-temporais, tal como mostrado na Figura 5.33.

Figura 5.33. Extensão do modelo MOSAICo

O recurso da herança é o principal mecanismo para incorporar a funcionalidade espaço-temporal aos objetos do MOSAICo. Porém, outras estratégias podem ser utilizadas, tal como o padrão de projeto *Decorator* (GAMMA; VLISSIDES; JOHNSON, 1994), que agrega dinamicamente funcionalidade a um determinado objeto. A Figura 5.34 apresenta o mapeamento do modelo da análise (a) para o modelo de projeto (b) seguindo a estratégia de herança.

Figura 5.34. Mapeamento do modelo da análise para modelo de projeto

Assim como o modelo de objetos espaciais, o modelo de objetos temporais também foi derivado do GeoFrame-T (LISBOA FILHO, 2000; ROCHA, 2001). A hierarquia representa os tipos temporais tempo de validade e tempo de transação. Esses dois tipos possuem uma associação para a primitiva temporal, que pode ser um instante ou um intervalo. Mesmo sendo um histórico de transação, a mesma pode ser feita dentro de um intervalo de tempo (Figura 5.35).

Figura 5.35. Modelo de objetos de tipos temporais

Ao contrário do GeoFrame-T, a hierarquia de objetos temporais não é associada à hierarquia de objetos espaciais. O objetivo é dar mais flexibilidade aos projetistas para incorporar as funcionalidades temporais, independentemente se são classes espaciais ou classes convencionais. Conceitualmente, o GeoFrame-T representa essa flexibilidade com a classe `Estatico`.

Os padrões de projeto Atributos Temporais (Padrão de Projeto: Atributos Temporais) e Relacionamentos Temporais (Padrão de Projeto: Relacionamentos Temporais) são exemplos de utilização dessa hierarquia de objetos. Quanto aos objetos espaciais, a estratégia de herança não parece a mais adequada, uma vez que a linguagem Java não permite a construção de heranças múltiplas. Para esses casos foi criado o padrão de projeto espaço-temporal Objetos de Negócio Espaço-Temporais (Padrão de Projeto: Objetos de Negócio Espaço-Temporais).

O Quadro 5.3 descreve as funções temporais disponíveis:

Tempo de Validade	
Igual	Retorna verdadeiro se um objeto possui o período de validade igual ao período de validade de um outro objeto. Caso contrário, retorna falso. Exemplo: <code>amostraPragas1.igual(amostraPragas2)</code>
Sobrepõe	Retorna verdadeiro se um objeto possui o período de validade que sobrepõe o tempo de validade de um outro objeto. Caso contrário, retorna falso. Exemplo: <code>amostraPragas1.sobrepoe(amostraPragas2)</code>
Contém	Retorna verdadeiro se um objeto possui o período de validade que contém o período de validade de um outro objeto. Caso contrário, retorna falso. Exemplo: <code>amostraPragas1.contem(amostraPragas2)</code>
Encontra	Retorna verdadeiro se o início do período de um objeto encontra o final do período período de validade de outro objeto. Exemplo: <code>evento1.encontra(evento2)</code>
Duração	Calcula e retorna a duração de tempo ocorrida entre dois eventos temporais, onde a duração calculada se refere ao período do objeto, podendo ser passado, presente ou futuro. Exemplo: <code>duracaoValidade = amostraPragas.duracao()</code>
Inicia	Retorna verdadeiro se o período de validade de um objeto se inicia na mesma data de início do período de validade de outro objeto. Exemplo: <code>amostraPragas.inicia(amostraCultura)</code>

Tempo de Validade	
Termina	Retorna verdadeiro se o período de validade de um objeto termina na mesma data do término do período de validade de outro objeto. Exemplo: <code>amostraPragas.termina(amostraCultura)</code>
Tempo de Validade e Tempo de Transação	
Precede	Retorna verdadeiro se o objeto possui o período de validade ou o tempo da transação anterior ao do objeto comparado. Caso contrário, retorna falso. Exemplo: <code>talhao1.precede(talhao2)</code>
Sucedee	Retorna verdadeiro se o objeto possui o período de validade ou o tempo da transação posterior ao do objeto comparado. Caso contrário, retorna falso. Exemplo: <code>talhao1.precede(talhao2)</code>

Quadro 5.3. Operações temporais

A Figura 5.36 apresenta o modelo de projeto parcial para o subsistema Dados de Campo do MOSAICo, derivado do padrão de análise Talhão e Amostras (Padrão de Análise: Talhão e Amostras). O padrão de projeto Objetos de Negócio Espaço-Temporais (Padrão de Projeto: Objetos de Negócio Espaço-Temporais) foi utilizado para modelar a classe espaço-temporal Talhão. Essa classe é um objeto espacial (polígono) que possui um histórico temporal do tipo tempo de transação. O padrão de projeto Relacionamentos Temporais (Padrão de Projeto: Relacionamentos Temporais) foi utilizado para modelar o relacionamento do tipo tempo de validade entre as classes Fazenda e Proprietário.

Figura 5.36. Modelo de Projeto Dados de Campo

5.4.3 Camada de Integração e Persistência

A camada de persistência deve realizar o mecanismo de persistência e recuperação de objetos de negócio dos meios de persistência utilizados. Sistemas Gerenciadores de Bancos de Dados Relacionais (SGBDR) são os meios mais utilizados, porém vem sendo crescente a utilização de mecanismos de persistência baseados em XML.

A camada de persistência deve garantir a flexibilidade de utilização de diferentes tecnologias, além de realizar a transformação de objetos para tuplas e vice-versa. Nesse sentido, a plataforma J2EE fornece a API JDBC. Essa API disponibiliza um conjunto de interfaces que possibilita o acesso aos dados, independente da tecnologia utilizada. Para mudar o SGBD, basta mudar a implementação da API, normalmente oferecida pelo fornecedor do SGBD. Há também alguns *frameworks* de persistência disponíveis, tal como *Hibernate* (SOURCEFORGE.net, 2006), que implementam o mapeamento Objeto-Relacional.

A especificação dessa camada apresenta um banco de dados relacional genérico, que é gerado pela infra-estrutura assim que a aplicação se conecta pela

primeira vez ao SGBD. Tanto as relações de dados espaço-temporais quanto as relações para armazenar os dados descritivos da aplicação são automaticamente criadas. As relações que armazenam os dados espaço-temporais possuem uma estrutura padrão, serão sempre as mesmas independentemente do domínio da aplicação. Já as relações de dados descritivos referentes ao domínio da aplicação são geradas a partir de um mapeamento objeto-relacional, em XML, que deve ser feito previamente pelo desenvolvedor da aplicação. A estrutura desse XML será apresentada mais adiante. A Figura 5.37 mostra o esquema relacional para suportar os dados espaço-temporais.

Figura 5.37. Modelo de Dados Espaço-Temporal

Todo objeto com propriedades espaço-temporais criado pela aplicação é armazenado na relação TB_OBJECTS, garantindo, assim, que cada objeto tenha um identificador único no banco de dados. Assim, o esquema do banco de dados pode ser único e independente da aplicação.

As relações de geometrias, por exemplo, `TB_POLYGONS_VT` e `TB_POINTS_VT`, possuem um campo identificador (`GEOM_ID`). Essa abordagem também garante a unicidade da geometria em sua classe e foi também a mesma estratégia utilizada pela API TerraLib (TERRALIB, 2006). Esses identificadores são sequenciais e gerados pela infra-estrutura.

Uma geometria da classe Ponto (`TB_POINTS_VT` e `TB_POINTS_TT`) é formada por uma coordenada (`TB_COORDINATES`). O relacionamento entre as duas relações é feito pelo campo `GEOM_ID`, com restrição de integridade do SGBD, e pelo campo `OBJECT_ID`, cujo valor deve ser o mesmo para as duas tuplas. Da mesma maneira é feito o relacionamento entre uma geometria da classe Linha (`TB_LINES_VT` e `TB_LINES_TT`) com as coordenadas. Uma geometria da classe Linha é formada por várias coordenadas, sendo que quando a última coordenada coincide com a primeira coordenada, a geometria se configura na forma de um anel. Um polígono (`TB_POLYGONS_VT` e `TB_POLYGONS_TT`), por sua vez, é formado por linhas do tipo anel.

As propriedades temporais das geometrias são armazenadas nas próprias tabelas das geometrias. As extensões do nome das tabelas, VT e TT, indicam Tempo de Validade e Tempo de Transação, respectivamente.

As tabelas `TB_STRING_VT`, `TB_STRING_TT`, `TB_NUMBER_VT` e `TB_NUMBER_TT` armazenam as propriedades temporais e o valor (cadeia de caracteres ou numérico) para dados descritivos do modelo.

5.4.3.1 Mapeamento Objeto-Relacional

O mapeamento objeto-relacional é um mecanismo para traduzir o modelo de objetos para o modelo relacional e vice-versa, utilizado pela infra-estrutura para a criação do esquema relacional do banco de dados e para a inclusão e recuperação de objetos neste banco de dados. Esse mapeamento é feito com XML e apoiada por um *XML Schema* (FALLSIDE; WALMSLEY, 2004) que define as regras e sintaxe desse XML. Atualmente, há vários produtos disponíveis que realizam a persistência de objetos em bancos de dados relacionais baseados em mapeamento XML, mas não atendem às necessidades da infra-estrutura, principalmente em relação aos dados espaço-temporais. Há também persistência de objetos baseada em XML, que não utiliza SGBDs convencionais, porém ainda não é largamente utilizada,

particularmente por aplicações comerciais. Os trechos de código das Figuras 5.38, 5.39, 5.40 e 5.41 mostram a sintaxe do XML de mapeamento.

```
<class
  name = "className"           1
  geom-type = "point|line|polygon" 2
  temporal-type = "tt|vt"      3
  table-name = "tableName"     4
/>
```

Figura 5.38. Mapeamento O/R: Tag class

A *tag class* é utilizada para representar uma classe do modelo de negócio da aplicação

- (1) Nome completo da classe, inclusive com a hierarquia completa do empacotamento. Obrigatório.
- (2) Tipo da geometria (ponto, linha, polígono). Opcional.
- (3) Tipo temporal (tempo de validade, tempo de transação). Opcional.
- (4) Nome da tabela. Obrigatório.

```
<subclass
  name = "subclassName"       1
  table-name = "tableName"    2
</subclass>
```

Figura 5.39. Mapeamento O/R: Tag subclass

A *tag subclass* representa uma subclasse, e deve estar subordinada a uma classe (*tag class*). Uma subclasse assume os tipos geométrico e temporal da superclasse.

- (1) Nome da subclasse. Obrigatório.
- (2) Nome da tabela. Sempre será criada uma tabela para cada subclasse de uma superclasse. Obrigatório.

```
<property
  name = "propertyName"      1
  type = "typeName"          2
  null = "yes|no"            3
  temporal-type = "tt|vt"     4
/>
```

Figura 5.40. Mapeamento O/R: Tag property

A *tag property* representa uma propriedade de uma classe.

- (1) Nome da propriedade de uma classe. Obrigatório.
- (2) Tipo de dados da propriedade. Pode assumir os seguintes valores: Integer, String, Character, Date, Timestamp, Float, Boolean, Blob, Collection. Obrigatório.
- (3) Indica se a propriedade pode conter valor nulo ou não. Obrigatório.
- (4) Indica o tipo temporal da propriedade, quando necessário. Opcional.

```
<association
  type = "one-to-many|many-to-one|many-to-many" 1
  class = "className"                           2
/>
```

Figura 5.41. Mapeamento O/R: Tag *association*

- A *tag association* representa uma associação de duas classes. Como exemplo, temos as ClasseA e ClasseB, e uma propriedade da ClasseA que representa uma associação entre as duas classes.

- (1) Tipo da associação que define o mapeamento para o modelo relacional. Obrigatório.

one-to-many: indica que a propriedade da ClasseA possui vários objetos da ClasseB. Essa propriedade deve ser, necessariamente, do tipo *Collection*. Como consequência para geração do modelo relacional, significa que a chave primária da tabela que representa a ClasseA será chave estrangeira na tabela que representa a ClasseB.

many-to-one: indica que a propriedade da ClasseA possui um objeto da ClasseB. No modelo relacional, a chave primária da tabela que representa a ClasseB deverá ser chave estrangeira na tabela que representa a ClasseA.

many-to-many: indica que a propriedade da ClasseA possui vários objetos da ClasseB. Essa propriedade deve ser, necessariamente, do tipo *Collection*. No modelo relacional, será criada uma nova tabela que conterá as chaves primárias das tabelas que representam as ClasseA e ClasseB.

- (2) Nome completo da classe associada, inclusive com o caminho do empacotamento. Obrigatório.

O trecho de XML da Figura 5.42 apresenta o mapeamento do modelo de projeto mostrado na Figura 5.36 para o esquema relacional.

```
<mapping>
  <class name="package.Amostra" geomType="ponto" table-name="TBAMOSTRA">
    <property name="responsavel" type="string" null="no"/>
    <property name="anotacoes" type="string" null="yes"/>
    <subclass name="package.AmostraPragas" table-name="TBAMOSTRAPRAGAS">
      <property name="nomePopular" type="String" null="no"/>
      <property name="nomeCientifica" type="String" null="no"/>
      <property name="metodoColeta" type="String" null="yes"/>
    </subclass>
  </class>
  ...
</mapping>
```

Figura 5.42. Exemplo de Mapeamento O/R

5.4.3.2 Processador de Consultas

Com a crescente utilização de tecnologias de Sistemas de Informações Geográficas, surgiu a necessidade de se prover linguagens de consulta com extensões espaciais e temporais. Frank (1982) apresenta as evidências das necessidades de extensões em linguagens de consultas convencionais. Basicamente, essas linguagens são derivadas do SQL, “SQL-like”, e introduzem tipos de dados espaciais e operadores espaciais, permitindo que usuários realizem consultas utilizando critérios espaciais, principalmente relações topológicas (*A CONTAINS B*) e métricas (*DISTANCE(A,B)<1000*).

Egenhofer (1994) apresenta uma linguagem baseada em SQL e na construção SELECT-FROM-WHERE, que corresponde às operações da álgebra relacional: projeção, produto Cartesiano e seleção, respectivamente. Essa linguagem de consulta realiza a recuperação de geometrias armazenadas em banco de dados relacionais com utilização de funções especiais, permitindo assim, a recuperação de dados convencionais utilizando critérios geográficos. Lin e Huang (2001) apresentam um *framework* Cliente/Servidor, o SQL/SDA (*Spatial Data Analysis*). Esse *framework* fornece, do lado Cliente, uma interface com estilo de linguagem visual que auxilia a usuários formularem suas consultas e, do lado Servidor, um processador de consultas em uma linguagem baseada em SQL.

Alguns trabalhos apresentam linguagens de consulta visuais. Mainguenaud, Portier e Cigales (1990) propõem a linguagem Cigales, que permite a construção de

consultas visuais por meio de ícones. A consulta é representada por uma expressão funcional interna que é depois traduzida para uma consulta SQL estendida.

Meyer (1993), apresenta a linguagem *Sketch!*, cuja tradução livre seria *Esboço!*. Essa linguagem oferece maior liberdade ao usuário, justamente por permitir que ele desenhe um esboço na tela que representa a consulta espacial. Esse esboço é interpretado pelo sistema e traduzido para uma linguagem de consulta.

Na mesma linha, Egenhofer (1996) propõe a linguagem *Spatial-Query-By-Sketch*, que utiliza uma tela sensível ao toque como dispositivo de entrada para que o usuário desenhe suas consultas. Essa mesma linguagem é estendida para aplicativos em dispositivos móveis (CADUFF e EGENHOFER, 2005). Em Bonhomme *et al.* (1999) é apresentada a linguagem *Lvis*, que realiza também consultas a dados temporais e possui o mesmo princípio da linguagem *Cigales*, em que o usuário realiza suas consultas com a ajuda de ícones. Um conjunto de ícones representa os tipos de objetos armazenados no banco de dados espaço-temporal e também os operadores lógicos, espaciais e temporais.

A biblioteca TerraLib, por sua vez, (TERRALIB, 2006) implementa consultas baseadas em funções pré-definidas escritas em linguagem C. Os critérios de busca são parâmetros “não espaciais” e “espaciais”, tais como restrições topológicas, e são passados para uma função de busca que recupera os objetos do banco de dados.

Há também iniciativas de linguagens de consulta para dados temporais. Fegaras e Elmasri (1998) apresentam uma linguagem baseada em OQL (*Object Query Language*) com extensões para realização de consultas temporais. Essa linguagem permite indicar intervalos de tempo na cláusula FROM e dessa forma, recuperar objetos válidos no período informado. Norvag (2002) define operadores que suportam consultas temporais em banco de dados XML e permite consultas em dados temporais de ambos os tipos, Tempo de Transação e Tempo de Validade.

Fabricantes de gerenciadores de banco de dados relacionais (SGBD) estenderam seus produtos para suportar dados espaciais. São exemplos o Informix, IBM DB2, Oracle, Postgres e MySQL. Em Queiroz e Ferreira, (2005) é possível encontrar uma descrição das principais extensões para os banco de dados Postgres e Oracle. Nesses gerenciadores, é possível realizar uma consulta SQL com restrições espaciais, implementadas em funções tais como *equals*, *intersects*, *overlaps* e *contains*. No entanto, funções temporais não são contempladas nesses sistemas.

A literatura mostra que linguagens de consulta baseadas na estrutura SQL possuem maior aceitação, até mesmo por se tratar de um padrão de mercado. Desse modo, é importante que a camada de persistência da aplicação forneça um mecanismo de consulta baseado na construção SELECT-FROM-WHERE.

O Padrão de Projeto: Consultar Objetos Espaço-Temporais pode ser utilizado como ponto de partida para implementação de um processador de consultas. A solução mostra como aproveitar as funções espaciais de um SGBD ou como interpretar as funções espaço-temporais na própria aplicação.

A seguir são apresentadas as cláusulas que compõe uma consulta.

Cláusula SELECT

Recebe um vetor de objetos do tipo `Class`. Cada `Class` indica o tipo de objeto cuja coleção deverá retornar em uma consulta.

```
Query query = new Query();
query.SELECT({[Class1],[Class2]})
```

Cláusula FROM

O parâmetro é um vetor com os nomes dos repositórios onde a consulta deverá pesquisar. Deve ser utilizado o nome do mapeamento objeto-relacional feito em XML.

```
query.FROM({[Nome Repositório1, Alias1],[Nome Repositório2, Alias2]})
```

Cláusula WHERE

Na cláusula `where` são passados os critérios de consulta. Pode receber expressões relacionais e também funções geométricas e temporais.

```
query.WHERE(new EQUAL(objeto.atributo,value))
```

Os critérios podem ser expressões booleanas, tais como:

```
query.WHERE(new EQUAL(objeto.atributo,value)).AND(new
EQUAL(objeto.atributo,value))
```

As funções geométricas e temporais também podem ser utilizadas como critérios de consulta. Exemplos:

```
query.WHERE(new CONTAINS(geometria1,geometria2)).AND(new
BEFORE(geometria1,geometria2))
```

Exemplo

1) Recuperar as amostras de solo provenientes do Talhão de número 1.

```
Query q = new Query();
q.SELECT({package.Amostra}).FROM({[Talhao,t],[Amostra,a]}).WHERE(new
CONTAINS(t,a)).AND(new EQUAL(t.numero,1));
Collection c = q.executeQuery();
```

5.4.3.3 Interoperabilidade com GML

A utilização de XML para prover interoperabilidade entre sistema tem sido uma estratégia amplamente utilizada na área de TI, tais como *Web Services* e SOAP, e também em trabalhos científicos. Murakami (2006) propõe uma linguagem padrão para troca de dados entre aplicações para AP denominada *Precision Agriculture Markup Language* – PAML, que estende a *Geographic Markup Language* - GML. Fidalgo *et al.* (2003) apresentam uma linguagem baseada em GML (COX *et al.*, 2002) para promover a integração entre tecnologias DW (*Data Warehouse*), OLTP (*On-Line Analytical Processing*) e Sistemas de Informações Geográficas, objetivando a construção de sistemas para apoio a decisão. Peng e Zhang (2004) também utilizam GML como mecanismo de codificação e transporte de dados geográficos pela Internet e seu trabalho.

A camada de integração utiliza como meio de comunicação de dados geográficos e temporais a PAML, uma extensão da GML. GML é uma codificação em XML *Schema* que permite modelar, transportar e armazenar informações geográficas, incluindo informações não espaciais de fenômenos geográficos, definidos pelo *Open Geospatial Consortium* (OGC, 2006). Os principais objetivos da GML são:

- Fornecer uma estrutura aberta para a definição de esquemas de aplicações e objetos geográficos;
- Suportar a descrição de esquemas de aplicações geográficas para domínios específicos e comunidades de informação;
- Permitir a criação e manutenção de esquemas e dados geográficos distribuídos e relacionados;
- Aumentar a habilidade das organizações para compartilhar dados geográficos;
- Servir como ferramenta para a interoperabilidade de Sistemas de Informação Geográfica - SIG.

A especificação de GML utiliza a linguagem XML Schema (FALLSIDE; WALMSLEY, 2004) para descrever a gramática das instâncias de dados. Essa linguagem representa um avanço significativo em relação a *Document Type Definition* (DTD), também utilizada para definir gramáticas em XML. Tais avanços podem ser percebidos em características do XML Schema, em detrimento ao DTD: possui 41 tipos de dados, é extensível, permite herança e possui o conceito de conjuntos.

A GML possui dois modelos de objetos: *Geometry* e a *Feature*. O modelo *Geometry* define geometrias como *Point*, *LineString*, *Polygon* e *LinearRing*, e também os conjuntos de geometrias, heterogêneas e homogêneas. As propriedades geográficas de entidades do mundo real devem ser representadas por classes desse modelo. O modelo *Feature* é utilizado para representar os fenômenos geográficos, que são fenômenos do mundo real e que possuem uma localização relativa à Terra. Uma *Feature* pode ou não possuir propriedades geométricas, o que é expresso através de um relacionamento com uma geometria. A Figura 5.43 mostra o esquema da PAML derivado do esquema da GML.

Figura 5.43. Intercâmbio de Dados Espaço-Temporais

O XML Schema `FieldData.xsd` é uma extensão das *features* e geometrias da PAML. Essa implementação representa o modelo conceitual MOSAICo definido por Saraiva (1998) e Saraiva, Massola e Cugnasca (1998), estendido com os aspectos espaciais e temporais, e faz parte de uma pesquisa de Doutorado desenvolvida no Laboratório de Automação Agrícola (LAA) da Escola Politécnica (EPUSP) (MURAKAMI, 2006). Entretanto, é importante ressaltar que, para qualquer outro modelo pode haver um XML Schema. Dessa maneira, o sistema poderá traduzir os objetos deste outro modelo para GML ou PAML e também entenderá os dados baseados no *XML Schema*. No exemplo dado para o Padrão de Projeto: Montar Objetos Espaço-Temporais Complexos é mostrada uma solução para montar objetos espaciais a partir de um arquivo GML.

5.5 Considerações Finais

Esse capítulo apresentou uma proposta de arquitetura de software baseada na plataforma JEE e em uma arquitetura de referência proposta por Murakami (2006) e Murakami *et al.* (2006). Essa arquitetura pode ser configurada em três cenários arquiteturais. Esses cenários mostram como as camadas físicas da plataforma (camada cliente, camada *web* e camada EJB) podem ser utilizadas para compor uma arquitetura em combinação com as camadas lógicas de um sistema (apresentação, negócio e persistência). As camadas físicas são comumente denominadas *tiers*, enquanto as camadas lógicas são denominadas *layers*.

As propostas foram então confrontadas segundo algumas propriedades arquiteturais (atributos de qualidade). Esse estudo pode ser utilizado por desenvolvedores para adotar o cenário mais adequado aos requisitos (principalmente não funcionais) do sistema a ser desenvolvido.

Os padrões de análise e projeto são o passo seguinte da escolha do cenário arquitetural. Os padrões de análise apontam como modelar os requisitos espaço-temporais do MOSAICo e os padrões de projeto mostram como resolver alguns problemas de projeto referentes a manipulação de objetos espaço-temporais. Esses padrões foram organizados em um catálogo e classificados de acordo com as camadas lógicas do sistema. Nos dois casos, o foco principal é prover reutilização, tanto de lógica de negócio quanto de soluções técnicas.

Em seguida, foram apresentadas as especificações de cada camada lógica. Para a camada de apresentação, recomendamos a utilização de *frameworks* MVC (MVC, 2006), tal como o *Struts* (STRUTS, 2006). Para a camada de negócio, especificamos as estruturas de classes para representar dados e operações espaço-temporais. Essas estruturas foram baseadas no modelo conceitual Geo-Frame (LISBOA FILHO, 2000) (ROCHA, 2001) e em conjunto com o MOSAICo, representam a lógica de negócio para aplicações para AP. Por fim, para a camada de persistência e integração, são especificados os mecanismos de persistência dinâmica, o processador de consultas e o padrão de interoperabilidade, o PAML (MURAKAMI, 2006).

A seguir, no próximo capítulo, um estudo de caso foi desenvolvido com implementação do Padrão de Projeto: Apresentar Objetos Espaciais segundo uma arquitetura centrada na *web*.

6 ESTUDO DE CASO

6.1 Introdução

Esse capítulo apresenta e discute um estudo de caso utilizado para validar um ponto da arquitetura de software definida neste trabalho. O ponto em questão foi selecionado a partir da camada de apresentação, mais especificamente o Padrão de Projeto: Apresentar Objetos Espaciais definido no capítulo 5 deste trabalho. O componente implementado chama-se `ViewTalhao`. O cenário arquitetural adotado para essa implementação está definido na Seção 5.2.2. Cenário Centrado na *Web*.

O problema se concentra na apresentação, em ambiente *Web*, de um talhão e suas amostras coletadas a partir da colheita de milho. Além das informações de posicionamento (espaciais) do talhão e das amostras, duas outras camadas de informações são consideradas. A taxa de produtividade e a taxa de umidade do grão de milho.

A camada de apresentação foi selecionada para esse estudo de caso pois se trata de um componente ainda crítico no desenvolvimento de sistemas de informações espaciais. Acreditamos que um conjunto de componentes *Web*, aliados a um *framework* de apresentação (JSF ou STRUTS), pode auxiliar bastante a construção de sistemas, conferindo produtividade ao processo, principalmente pela possibilidade de reutilização que tais componentes podem oferecer.

6.2 Conversão do PAML para o SVG

O PAML, definido por Murakami (2006), define uma linguagem para armazenamento e troca de informações espaciais entre aplicações. Essa linguagem é uma herança do GML, conforme mostrado no capítulo anterior. A apresentação dessa informação, entretanto, não é automática em navegadores *Web*, sendo então

necessária a conversão para um formato em que o navegador interprete e apresente adequadamente. Esse formato é o *Scalable Vector Graphics* (SVG).

A Figura 6.1 apresenta o esquema da conversão e a seguir uma breve explicação das tecnologias envolvidas na construção do componente.

Figura 6.1. Esquema de conversão do PAML → SVG

6.2.1 Scalable Vector Graphics (SVG)

O SVG é uma linguagem XML para descrição de gráficos bi-dimensionais. A especificação completa do SVG pode ser encontrada em SVG (2006). O SVG permite três tipos de objetos gráficos: elementos geométricos vetoriais (linhas, polígonos, pontos, etc); elementos de texto e imagens. Suporta também animações e interatividade com inclusão de código (*scripts*).

Adicionalmente, o SVG possui uma linguagem de *script* que possibilita o acesso a todos os elementos, atributos e propriedades, o SVG Document Object Model. Aliado aos manipuladores de eventos (*onmouseover* e *onclick*), é possível manipular um SVG mediante requisições de usuários pelo navegador *Web* (*browser*) e dessa maneira, construir interatividade com os elementos da tela.

O SVG é um formato não proprietário, desenvolvido no âmbito do consórcio W3C, com a participação de grandes fabricantes de software. Uma lista de

implementações pode ser encontrada em (<http://www.svgi.org/>), desde visualizadores, editores, visualizadores para dispositivos móveis, etc.

O termo “*Scalable*” significa aumentar e diminuir uniformemente. Em relação aos gráficos, significa que o SVG não é limitado a um tamanho de ponto (*pixel*) único e fixo. Desse modo, um mesmo gráfico em SVG pode ser apresentado em diferentes resoluções, por exemplo, ser apresentado com o mesmo tamanho em monitores com resoluções diferentes.

Gráficos vetoriais são objetos geométricos tais como linhas e curvas. Comparativamente ao formato matricial (GIF, JPEG), o gráfico vetorial oferece maior flexibilidade de manipulação. Por exemplo, caso seja necessário funções de “*zoom*” em imagens matriciais é necessário manter esses arquivos com grande resolução e mesmo assim pode-se perder em qualidade na ampliação da imagem.

6.2.1.1 O Formato SVG

O trecho de código da Figura 6.2 mostra a estrutura geral de um documento SVG. Ela é composta de duas partes principais: a primeira contém um conjunto de definições e a segunda é composta pelos elementos gráficos.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.0//EN"
"http://www.w3.org/TR/2001/REC-SVG-20010904/DTD/svg10.dtd">
<svg width="1200" height="550" viewBox="-22.701048 -49.992294 0.00921688
0.009466553">
<desc>Talhão com amostras filtradas</desc>
<defs>
...
</defs>
...
</svg>
```

Figura 6.2. Estrutura básica do documento SVG

Os atributos *width* e *height* especificam as dimensões da imagem. O exemplo mostra que o documento SVG ocupará uma área de 1200 x 550 pontos na tela do monitor. Já o atributo *viewBox* permite o ajuste do desenho em um espaço de coordenadas com outras dimensões. No exemplo da Figura 6.2, a imagem se desenvolverá em um espaço de -22.701048 a -22,71026488 sobre o eixo X e de -49.992294 a -50,001760553 sobre o eixo Y em um espaço com medidas 1200 x 550.

6.2.1.2 Formas Básicas, Caminhos e Texto

SVG define 6 formas básicas que, juntamente com os caminhos (*paths*), podem ser combinados para formar qualquer possível imagem. Cada uma dessas formas possui propriedades que especificam sua posição e tamanho. Suas cores e contornos podem ser definidos pelas propriedades *fill* e *stroke*, respectivamente. São elas:

- *circle* (Círculo)
- *ellipse* (Elipse)
- *rect* (Retângulo)
- *line* (Linha)
- *polyline*
- *polygon* (Polígono)

A Figura 6.3 mostra o resultado no navegador Internet Explorer das 6 formas geométricas básicas possíveis no SVG.

Figura 6.3. Formas geométricas do SVG

Quando as formas básicas não forem suficientes para a imagem que se deseja montar, como no caso da necessidade de se desenhar curvas, o recurso de “caminho” (*paths*) do SVG pode ser útil. Esse recurso compreende uma série de comandos que são usados para criar uma forma como parte de uma imagem. Essa forma pode ser aberta (tal como *polyline*) ou fechada (tal como *polygon*) e pode conter linhas, curvas e segmentos.

Textos também podem ser adicionados em arquivos SVG. Os textos no SVG podem conter efeitos gráficos e também podem ser formatados com os efeitos disponíveis nas folhas de estilo tradicionais (CSS) do formato HTML, conforme exemplo da Figura 6.4.

Figura 6.4. Texto em SVG

Originalmente, o SVG não foi criado para visualização de informações geográficas. Entretanto, as características do SVG que foram apresentadas permitem a adequação da linguagem para apresentar esse tipo de informação. A capacidade de adequação do SVG a diferentes espaços pode ser utilizada para simular um espaço geográfico, permitindo a conversão de um ponto na tela para uma coordenada geográfica (latitude e longitude). O atributo *viewBox* permite

mostrar diferentes escalas de uma mesma imagem, possibilitando o recurso de zoom.

As seis formas geométricas disponíveis e o recurso de caminho (*path*) do SVG permitem desenhar qualquer imagem referente a mapas, talhões, amostras, rios, etc. Todas as formas podem ser posicionadas. Os recursos de texto permitem personalizar as imagens, tais como inserção de legendas. Finalmente, a possibilidade de interatividade permite que o usuário interaja com a imagem através de um clique do mouse, por exemplo. Isso significa que o sistema pode tomar ações em resposta a requisições do usuário feitas diretamente na imagem da tela.

6.2.2 Transformações PAML para SVG

O *eXtensible Stylesheet Language* (XSL) foi desenvolvido pelo consórcio W3C. O XSL está para o XML tal como o CSS está para o HTML, ou seja, funciona como uma folha de estilos do XML, embora o XSL vá além de que uma linguagem de estilos. O XSL consiste de três partes:

- XSLT: uma linguagem para transformar documentos.
- *Xpath*: uma linguagem para navegar em documentos XML.
- XSL-FO: uma linguagem para formatar documentos XML.

O XSL *Transformations* (XSLT, 2006) é utilizado para transformar um documento XML em outro documento XML ou em outro documento que é reconhecido por um navegador, tal como HTML e SVG.

No contexto deste trabalho, o XSLT foi utilizado para transformar o PAML, que é um XML derivado do GML, em formato SVG, que também é um XML. Essa abordagem de transformação vem sendo recentemente utilizada (PENG e ZHANG, 2004) (ROCHA, 2005) e (CARDOSO; ROCHA; LOPES, 2004), especialmente o formato GML para SVG na camada de apresentação de aplicações *Web*. O esquema apresentado na Figura 6.1 mostra a necessidade de se escrever o *Stylesheet*. Esse arquivo deve conter as regras de conversão de um formato (PAML) para outro formato (SVG). Um trecho do arquivo `layout.xsl` (Figura 6.5), utilizado pelo componente para realizar a conversão é apresentado a seguir. Esse trecho mostra as regras de conversão de um polígono do PAML para um elemento

polygon do SVG. O elemento *polygon* foi utilizado para desenhar a forma de um talhão.

```
?xml version="1.0" encoding="UTF-8" ?>
<xsl:stylesheet xmlns:xsl=
"http://www.w3.org/1999/XSL/Transform" version="1.0">
<xsl:output method="xml"/>
<xsl:template match="metadata">
<svg>
...
<xsl:apply-templates/>
</svg>
</xsl:template>

<xsl:template match="grupo">
<g>
  <xsl:template match="poligono">
    <polygon>
      <xsl:attribute name="points">
        <xsl:value-of select="@pontos"/>
      </xsl:attribute>
      <xsl:attribute name="stroke">
        <xsl:value-of select="@corLinha"/>
      </xsl:attribute>
      <xsl:attribute name="stroke-width">
        <xsl:value-of select="@linha"/>
      </xsl:attribute>
      <xsl:attribute name="fill">
        <xsl:value-of select="@cor"/>
      </xsl:attribute>
    </polygon>
  </xsl:template>
...
</g>
</xsl:stylesheet>
```

Figura 6.5. Trecho do arquivo layout.xsl

6.3 Implementação do Componente ViewTalhao

O componente *ViewTalhao* é baseado no Padrão de Projeto: Apresentar Objetos Espaciais. O objetivo desse componente é retirar do componente de Visão (JSP) a lógica de conversão de PAML para SVG.

Utilizamos a estratégia *Tag Lib* para implementar a principal classe auxiliar. *Tags* customizadas é um recurso da API Java que possibilita a construção de componentes *Web* que podem ser chamados a partir de páginas JSP. As consequências principais dessa estratégia são a possibilidade de reuso e a simplificação do código das páginas JSP.

O diagrama de classe apresentado na Figura 6.6 apresenta a solução do componente `ViewTalhao`. Esse componente recebe como parâmetro, por uma página JSP, o arquivo PAML, a largura e altura da imagem e a classe de regras.

Figura 6.6. Diagrama de classe: componente `ViewTalhao`

As classes de regras, representadas no diagrama da Figura 6.4 pelas classes `Moisture` e `DryYield`, são responsáveis por regras de apresentação específicas de cada uma das camadas de informação. Para esse exemplo, a regra implementada para cada camada foi a determinação da cor da amostra.

O arquivo `amostrasFiltradas.paml` corresponde a informações geográficas de um talhão e suas amostras. Para cada amostra, há as informações de taxa de produtividade (`DryYield`) e a taxa de umidade do grão de milho (`Moisture`). A necessidade de regras para cada informação deriva do fato de que cada atributo representa um domínio diferente e conseqüentemente, possui faixas de valores diferentes. A vantagem do padrão de projeto *<<GoF>> Strategy* (GAMMA; VLISSIDES; JOHNSON, 1994) é a flexibilidade de criar novas regras para outras medições, sem a necessidade de criar outro componente.

6.3.1 Infraestrutura de Software

O Quadro 6.1 apresenta a infra-estrutura de software utilizada para a implantação do estudo de caso. O *plugin* SVGViewer é necessário para capacitar o Internet Explorer a interpretar o código SVG.

Software	Versão
Apache Tomcat	5.5
JSP	2.0
Servlet	2.3
JDK (Java)	1.5.0_10
ADOBE SVGViewer	3.0
Microsoft Internet Explorer	7.0

Quadro 6.1. Infra-estrutura de Software do componente ViewTalhao

A Figura 6.7 apresenta a arquitetura de software utilizada para implementação e teste do componente de software. Essa arquitetura corresponde ao cenário arquitetural definido na Seção 5.2.2. Cenário Centrado na Web.

Figura 6.7. Cenário arquitetural utilizado para implementação e teste do componente ViewTalhão.

6.4 Resultados Alcançados

A Figura 6.8 apresenta o resultado da utilização do componente em uma página JSP. A imagem mostra os dados brutos, sem filtragem. A camada de dados representada é a de umidade do grão de milho. Quanto mais escuro o tom de azul, maior a umidade do grão. Por sua vez, a Figura 6.9 apresenta o talhão com as amostras já filtradas. Nessa imagem, muitos pontos referentes a amostras foram retirados por conterem alguns defeitos que vão desde erros de posicionamento até valores nulos para produtividade e umidade. O software de filtragem foi desenvolvido por Murakami (2006), como estudo de caso em sua tese de Doutorado. Os detalhes sobre o algoritmo de filtragem também podem ser encontrados em Murakami (2006).

Figura 6.8. Mapa do talhão de amostras brutas (umidade do grão de milho)

Figura 6.9. Mapa do talhão de amostras filtradas (umidade do grão de milho)

O mesmo arquivo de dados PAML foi submetido à transformação pelo componente, mas desta vez com as regras referentes à camada de dados Produtividade. A Figura 6.10 a seguir apresenta o resultado para o arquivo de dados filtrados. A legenda da figura explica qual foi a regra utilizada.

Realizando uma comparação visual entre a Figura 6.9 e a Figura 6.10, é possível distinguir claramente uma relação entre a taxa de umidade do grão de milho com a taxa de produtividade. Percebe-se que nas áreas de maior produtividade (os pontos verdes na Figura 6.10), a umidade também é maior (pontos azuis escuros na Figura 6.9). Esse padrão é bem visível nas extremidades superior e inferior das figuras. O significado dessa eventual correlação, todavia, escapa ao escopo deste trabalho.

Figura 6.10. Mapa do talhão de amostras filtradas (taxa de produtividade)

6.5 Considerações Finais

O componente de apresentação `ViewTalhão` converte um arquivo de dados de um Talhão e Amostras em formato PAML para SVG. O SVG é uma linguagem de imagens vetoriais que é interpretada por navegadores *Web*. Esse componente implementa o Padrão de Projeto: Apresentar Objetos Espaciais do catálogo de padrões criados como parte da arquitetura proposta. A implementação conforme o padrão definido possibilitou um componente de baixo acoplamento e reutilizável.

A utilização do componente também é bastante simples. A estratégia de *Tag Lib* é bastante difundida. Ainda, o componente pode facilmente ser integrado a uma biblioteca de componentes *Web* de um *framework* de mercado, tal como o Struts. A Figura 6.11 mostra o trecho de código de uma página JSP com duas chamadas para

o componente, uma com as regras da camada de dados de Produtividade e outra com as regras da camada de dados de Umidade.

```
<%@taglib uri="/geotags" prefix="geotags" %>

<geotags:viewTalhao link="Amostras Filtradas - Humidade"
gml="/amostrasFiltrada.gml"
regraClass="poli.usp.taglib.transferengine.rules.Moisture"
altura="550" largura="800" />

<geotags:viewTalhao link="Amostras Filtradas - Produtividade"
gml="/amostrasFiltrada.gml"
regraClass="poli.usp.taglib.transferengine.rules.DryYield"
altura="550"
largura="800" />
```

Figura 6.11. Trecho de página JSP com chamada para o componente ViewTalhao

7 CONSIDERAÇÕES FINAIS

Esse capítulo contém os comentários finais deste trabalho de Doutorado. Nele, são apresentadas as conclusões de todo o projeto de pesquisa desenvolvido, a interligação com outros trabalhos do LAA e as principais dificuldades encontradas. Por fim, apresentamos as principais contribuições deste projeto e também os próximos passos propostos para continuidade da pesquisa.

7.1 Conclusões

A construção de sistemas de informação para AP envolve vários aspectos que podem ser considerados não-triviais para a área de desenvolvimento de sistemas. Um desses aspectos diz respeito ao tipo da informação que esses sistemas devem manipular, as informações espaciais e temporais. A representação das informações geográficas em sistemas computacionais envolve um grande esforço de modelagem e de construção. Neste trabalho apresentamos um conjunto de aspectos que devem ser considerados em sistemas para AP: os requisitos gerais para AP e características espaciais e temporais que os sistemas devem conter.

Esse estudo mostrou que, para que seja possível resolver a etapa de gerenciamento de informação no processo da AP, seria necessária uma abordagem única para integrar todos os aspectos envolvidos. A partir dessa conclusão, foi realizado um levantamento sobre alguns modelos de dados geográficos que podem ser utilizados para construção de sistemas espaço-temporais para AP.

Em maior ou menor grau, todos os modelos atendem satisfatoriamente aos requisitos de sistemas geográficos. Entre esses requisitos está a capacidade de modelar a localização de eventos, extensão temporal, eventos espaciais complexos, generalizações, relacionamentos métricos e topológicos e restrições espaço-temporais. À luz desses comentários, pode-se concluir que os modelos já existentes atendem às necessidades espaço-temporais para sistemas voltados a AP. Desse modo, no âmbito desse trabalho, foi adotado o modelo GeoFrame-T (LISBOA

FILHO, 2000) (ROCHA, 2001) (ROCHA e EDELWEISS, 2001). Essa escolha se mostrou adequada, pois o modelo possui estruturas suficientes para representação espacial e temporal, além de ser especificado em UML, uma linguagem que se tornou padrão de mercado para especificação de sistemas e também foi adotada no contexto desta tese. A proposta para extensão do MOSAICo com aspectos espaço-temporais foi apresentada nos trabalhos (RIBEIRO; SARAIVA; MURAKAMI, 2003; 2004; RIBEIRO *et al.*, 2002; 2003).

Foi adotada como tecnologia a plataforma JEE, que por ser baseada em camadas físicas (*tiers*), possibilitou a configuração de cenários arquiteturais, cada qual com suas características. Em resumo, cada cenário arquitetural proposto organiza as camadas lógicas (apresentação, negócio e persistência / integração) por entre as camadas físicas da plataforma JEE (Cliente, container *Web* e container EJB). Essa abordagem se mostrou bastante adequada por dois motivos: em primeiro lugar, foi possível associar as vantagens e desvantagens da plataforma com os requisitos não funcionais de uma aplicação para AP, tornando a arquitetura flexível suficiente para atender a um conjunto diverso de necessidades; em segundo lugar, a tecnologia JEE aborda soluções para questões relacionadas a interoperabilidade, persistência, interface *Web*, escalabilidade e reusabilidade.

Por sua vez, o catálogo de padrões de análise e de padrões de projeto criado representa o modelo de análise e o modelo de projeto de sistemas espaço-temporais para AP, respectivamente. Esses padrões foram agrupados de acordo com cada camada lógica e têm como principal objetivo a possibilidade de reuso. A abordagem baseada em padrões é bastante interessante, à medida que permite a prática intensiva de reutilização em dois sentidos:

- Reutilização de conhecimento, considerando que os padrões de análise e projeto possuem soluções prontas. Especificamente, os padrões de análise embutem o conhecimento adquirido nas fases de requisitos e análise de um processo de desenvolvimento e os padrões de projeto possuem conhecimento técnico elaborado em uma fase de projeto de sistemas.
- Reutilização de código, já que os padrões de projetos são criados para gerar componentes altamente coesos. A especialização desses padrões por camadas lógicas facilita a construção de componentes com baixo acoplamento.

No entanto, é necessário que testes sejam feitos para avaliar aspectos referentes à aplicabilidade e principalmente desempenho de componentes implementados a partir dos padrões de projeto propostos. Apesar do fato de que os padrões de projeto definidos neste trabalho foram concebidos a partir de padrões já conhecidos, o principal obstáculo deste trabalho foi realizar a validação dos aspectos citados. Nesse sentido, o estudo de caso implementado como parte desta tese mostrou que é possível criar um componente de apresentação coeso e desacoplado, mas que outros problemas precisam ser tratados, como tráfego de grandes volumes de dados pela *Web* e capacidade de processamento do cliente.

7.2 Principais Contribuições

Este trabalho de tese apresentou uma proposta de arquitetura de software que procura solucionar algumas questões importantes que estão envolvidas no processo de desenvolvimento de aplicações, principalmente para ambientes tão heterogêneos como o composto pelo domínio da AP.

Essa proposta foi uma evolução dos trabalhos desenvolvidos por Saraiva (1998) e Murakami (2006), no sentido de se disponibilizar aplicações ou serviços para AP na *Web*. Primeiro, Saraiva (1998) definiu um modelo conceitual, o MOSAICo, que representa os requisitos para AP. Na seqüência, Murakami (2006) definiu uma arquitetura de referência que adotou como modelo de referência o próprio MOSAICo e um estilo arquitetural baseado em camadas e orientado a serviços. Essa arquitetura de referência serviu como ponto de partida para a definição da arquitetura de software deste trabalho de pesquisa.

Os cenários arquiteturais definidos contribuem para atender à diversidade de necessidades em aplicações espaço-temporais para *Web*. Neste trabalho de tese foram feitas recomendações de uso e uma análise qualitativa dos cenários arquiteturais segundo alguns atributos arquiteturais. Entre os atributos avaliados, estão escalabilidade, portabilidade, simplicidade e extensibilidade.

A interoperabilidade de informações foi resolvida pela arquitetura de software proposta a partir da linguagem PAML (baseada em GML), definida por Murakami (2006) e Murakami *et al.* (2006). Essa característica confere às aplicações a

possibilidade de troca de dados de uma maneira padronizada e reconhecida internacionalmente.

A arquitetura de software proposta também possibilita que um alto grau de extensibilidade possa ser obtido pelas aplicações. A divisão em camadas lógicas possibilita que as responsabilidades de um sistema sejam bem distribuídas, permitindo a adição de novas funcionalidades sem maiores impactos de manutenção. Um exemplo clássico seria a criação de uma interface com usuário do tipo “*stand-alone*” em um sistema *Web* já construído: toda a lógica de negócio e persistência estariam prontas, bastando construir a nova camada de apresentação.

Contudo, os pontos mais importantes e, portanto, as principais contribuições, foram as soluções representadas pelos padrões de análise e padrões de projeto. Organizados em um catálogo, os padrões definidos conferem alto grau de reusabilidade e, conforme citado na seção anterior, reusabilidade de conhecimento e reusabilidade de código.

Segundo estudos publicados por Grady (1992), a fase de projeto, que no modelo de ciclo de vida em cascata engloba as disciplinas de análise e de projeto, consome em torno de 19% do esforço total de um projeto de desenvolvimento. O esforço utilizado nessa fase pode ser diminuído com a adoção de padrões, tanto de análise quanto de projeto. Esse fato pode se tornar mais evidente em sistemas com informações espaço-temporais, que envolvem conhecimento especializado. Outro ganho significativo pode ser obtido na qualidade do produto, uma vez que os padrões carregam conhecimento adquirido e testado em outras experiências.

O estudo de caso implementado neste trabalho de tese e apresentado no Capítulo 6 configura-se como uma boa evidência da capacidade de reutilização de um componente baseado em um padrão de projeto. A reutilização de código é mostrada com a disponibilização de *Tag Libs* para apresentar objetos geográficos em navegadores *Web*, e que podem ser utilizados a partir de páginas JSP. Por sua vez, a reutilização de conhecimento pode ser encontrada nas classes de regras implementadas neste mesmo componente, que embutem um conhecimento agrônomo relacionado a faixas de valores para determinadas propriedades de uma amostra, tal como produtividade e umidade.

7.3 Trabalhos Futuros

Nessa seção, procuramos apresentar alguns itens que consideramos propostas de projetos relevantes para a continuidade deste trabalho de pesquisa.

A principal proposta seria a implementação do jSTF (*Spatio-Temporal Framework*) (RIBEIRO; SARAIVA; MURAKAMI, 2006), que viria ao encontro da principal dificuldade encontrada nessa pesquisa e já discutida neste capítulo: a validação, em termos de desempenho e aplicabilidade dos padrões de projeto definidos.

Apesar de ser concebido para sistemas voltados para a AP, o *framework* seria genérico, de forma que partes dele poderiam atender a outros projetos existentes no LAA. Um desses projetos é o *openModeller* (SANTANA *et al.*, 2006), um projeto de software voltado a pesquisas em biodiversidade e que possui necessidades de arquiteturas que permitam processamento paralelo e interface baseada na *Web*.

O enfoque de código aberto (*open-source*) é muito importante para essas pesquisas futuras. A exemplo de outras iniciativas, tal como TerraLib, essa abordagem deve, no futuro, proporcionar a divulgação e crescimento dos trabalhos de pesquisa e dos produtos resultantes.

Um ponto de partida importante para esse trabalho seria o desenvolvimento de pesquisas no sentido de se projetar e implementar um conjunto de componentes *Web* para apresentação e manipulação de dados espaciais, tal como o exemplo feito nesta tese, e integrá-los a um *framework* de apresentação conhecido, por exemplo o *Struts*. Funções de *zoom-in* e *zoom-out*, funções para interatividade do usuário, funções de rotação e outras podem ser parte de tais componentes.

Problemas relacionados ao desempenho das aplicações também devem ser tratados nas pesquisas futuras. O estudo de caso desenvolvido mostrou que o volume de dados envolvidos em sistemas para AP tende a ser significativamente grande. Os dados experimentais de um talhão apresentaram 12.000 amostras brutas, o que gerou um arquivo, no formato PAML, de 7 Mbytes de informação. Essa desvantagem é inerente a arquivos XML, que tendem a aumentar de tamanho pois contêm informações além do dado puro.

A manipulação do PAML não se mostrou lenta, porém, o SVG gerado para a apresentação, apesar de menor (1 Mbyte), é considerado grande para transitar como

uma página *Web*. Essas questões podem criar três problemas em sistemas baseados na *Web*.

1. O processamento do lado servidor tende a ser custoso. Grandes arquivos podem significar alta carga de processamento. Além disso, um sistema com um número razoável de usuários pode exigir uma grande capacidade de processamento de consultas ao servidor de banco de dados.
2. Limitações na conexão e largura de banda na *Web* podem inviabilizar uma solução que tenha que mostrar mapas com muitas informações, pois o volume de dados a trafegar tende a ser grande.
3. Por último, o processamento local realizado pelo navegador *Web* (*browser*) também requer atenção. Ainda que testes nesse sentido não foram realizados, é perceptível que o navegador *Web* consome um tempo para desenhar as imagens vetoriais do SVG. Isso implica que usuários com computadores de capacidade de processamento menor podem ser prejudicados, dependendo do tamanho da página retornada em uma consulta.

Nesse sentido, algumas opções precisam ser pesquisadas e embutidas como solução nos componentes do jSTF:

1. O tamanho do SVG está exclusivamente ligado à quantidade de caracteres do documento. Nesse sentido, a tecnologia SVG oferece um recurso de definições prévias, onde figuras e outros itens podem ser definidos em uma seção do documento e reutilizados ao longo do documento. O SVG permite também a utilização de folhas de estilo (CSS), tal como HTML. Isso deve retirar do SVG informações de formatação de fontes e cores, por exemplo.
2. O protocolo HTTP permite a transmissão de arquivos compactados, o que minimizaria o tamanho dos documentos SVG e conseqüentemente o tráfego de informações na rede.
3. Há uma tecnologia recente para *Web*, o AJAX (*Asynchronous JavaScript and XML*) (CRANE; PASCARELLO, 2005), que procura minimizar os efeitos indesejáveis de uma página muito grande. Em resumo, essa tecnologia permite que requisições ao servidor *Web* sejam feitas de maneira assíncrona, em processamento *background*, e possibilita que a

interface mostre o resultado aos poucos para o usuário. Um exemplo típico da utilização dessa tecnologia é o *Google Maps*. Essa tecnologia também pode possibilitar que a consulta aos dados seja paginada, minimizando o processamento no servidor de banco de dados.

4. Por fim, a capacidade de processamento pode ser aumentada com a configuração de um servidor de aplicações em *clusters*. Desse modo, vários servidores podem trabalhar em conjunto e dividir a responsabilidade de processar uma tarefa considerada pesada.

Finalmente, outra opção de pesquisa futura que propomos está relacionada a componentes de software para dispositivos móveis. Conforme informações disponibilizadas pela Anatel³, está aberta desde Julho de 2006 uma licitação pública para a concessão da banda larga de telefonia móvel, pela qual o poder público concederá por um período de 15 anos os direitos sobre o serviço a empresas privadas. A promessa é que essas redes cheguem a lugares onde o cabeamento da solução atual para banda larga não esteja disponível.

A despeito dos prazos referentes aos trâmites legais e a operacionalização do serviço, esse fato abre uma boa perspectiva para efetiva utilização dos recursos de celulares de 3ª geração e computadores de mão. A tecnologia adotada como parte da solução desse projeto de pesquisa, a plataforma JEE, disponibiliza a API J2ME (*Java 2 Micro Edition*), que possibilita a implementação de aplicações para dispositivos móveis. Nesse sentido, a pesquisa e implementação de componentes baseados nos padrões definidos podem ser estendidas para essa área de atuação.

³ http://www.anatel.gov.br/Tools/frame.asp?link=/biblioteca/releases/2006/release17_07_2006r11.pdf

8 REFERÊNCIAS

- ABRANTES, M.G.; GOMES, M.R. GEO-OMT: An Object-Oriented Method Supporting the Development of Facilities Management Systems. In: GIS PLANET'98 ANNUAL CONFERENCE, 1998, Lisboa, Portugal. **Proceedings**. Lisboa, Portugal. Setembro 1998. (CD-ROM).
- ALUR, D.; CRUPI, J.; MALKS, D. **Core J2EE Patterns: Best Practices and Design Strategies**. 2nd Edition. New York: Prentice Hall / Sun Microsystems Press, June, 2003.
- ASSAD, E.,D.; SANO, E.,E. **Sistema de Informações Geográficas**: Aplicações na Agricultura. Brasília: Embrapa-SPI / Embrapa-CPAC, 1998, 434p
- BASS, L.; CLEMENTS, P.; KAZMAN, R. **Software Architecture in Practice**, 2nd Edition. Addison-Wesley Professional, 2003.
- BÉDARD, Y. Visual Modelling of Spatial Databases: Towards Spatial PVL and UML. *Geomática*, 53(2), p. 169-185. 1999.
- BÉDARD, Y.; CARON, C.; MAAMAR, Z.; MOULIN, B; VALLIÈRE, D. A. Adapting data models for the design of spatio-temporal databases. **Computer, Environment, and Urban Systems**, v.20, n.1, p. 19-41. January, 1996.
- BÉDARD, Y.; LARRIVÉE, S.; PROULX M-J.; NADEAU M. Modeling Geospatial Databases with Plug-Ins for Visual Languages: A Pragmatic Approach and the Impacts of 16 Years of Research and Experimentations on Perceptory, In: INTERNATIONAL WORKSHOP ON CONCEPTUAL MODELING FOR GIS, 1st, 2004, Shanghai. China. **Proceedings**. Shanghai. China. 2004. p. 17–30.
- BONHOMME, C.; TRÉPIED, C.; AUFAURE, M-A.; LAURINI, R. A visual language for querying spatio-temporal databases. In: ACM INTERNATIONAL SYMPOSIUM ON ADVANCES IN GEOGRAPHIC INFORMATION SYSTEMS, 7, 1999. Kansas City, Missouri, USA, **Proceedings**. Missouri, USA, 1999. p.34 – 39.

BOOCH, G.; JACOBSON, I.; RUMBAUGH, J. UML: Guia do Usuário. 2 edição. São Paulo: Editora Campus. 2005.

BORGES, K.A.V. **Modelagem de Dados Geográficos: uma extensão do modelo OMT para aplicações geográficas**. 1997. Dissertação (Mestrado) - Fundação João Pinheiro. Belo Horizonte, 1997.

BORGES, K.A.V.; DAVIS JR., C.A.; LAENDER, A.H.F. OMT-G: An Object-Oriented Data Model for Geographic Applications. In: **Geoinformatica** v. 5, n(3), 221-260, 2001.

BREUNIG, M.; TÜRKER, C.; BÖHLEN, M.H.; DIEKER, S.; GÜTING, R.H.; JENSEN, C.S.; RELLY, L.; RIGAUX, P.; SCHEK, H-J.; SCHOLL, M. Architectures and Implementations of Spatio-temporal Database Management Systems. In: KOUBARAKIS *et al.* **Spatio-Temporal Databases: The Chorochronos Approach**. Berlin, Springer, 2003. p.264-318.

CADUFF, D.; EGENHOFER, M. Geo-Mobile Queries: Sketch-Based Queries in Mobile GIS Environments. **Lecture Notes in Computer Science**, Springer, December, 2005. p.143-154. Presented in International Workshop on *Web and Wireless Geographical Information Systems*, 5, Lausanne, Switzerland. 2005.

CÂMARA, G., CASANOVA, M., HEMERLY, A., MAGALHÃES, G., MEDEIROS, C. **Anatomia de Sistemas de Informação Geográfica**. Campinas: Instituto de Computação, UNICAMP., 1996. 197p.

CÂMARA, G.; SOUZA, B.; PEDROSA, L.; VINHAS, A.; MONTEIRO, J.; PAIVA, M.; CARVALHO M. GATTASS, M. TerraLib: Technology in Support of GIS Innovation. In: SIMPÓSIO BRASILEIRO DE GEOINFORMÁTICA, 2, 2000, **Proceedings**. Brasil. 2000.

CÂMARA, G.; VINHAS, L.; SOUZA, J.R. PAIVA, A. M.; CARVALHO, M.; RAOULT B. Design Patterns in GIS Development: The Terralib Experience. In SIMPÓSIO BRASILEIRO DE GEOINFORMÁTICA, 3, 2001, Rio de Janeiro. **Proceedings**. Rio de Janeiro, 2001.

CARDOSO, J; ROCHA, A. J.; LOPES, C. M-GIS - Sistema Móvel Interoperável de Informação Geográfica. In: XML: APLICAÇÕES E TECNOLOGIAS ASSOCIADAS – XATA. 2004, Porto, Portugal. **Proceedings**. Porto, Portugal, 12-13 Fevereiro, 2004. p71-87.

- COUTO, L.; CASANOVA, C.; SCHWABE, D. Desenvolvimento de Sistemas de Informação Geográfica Cooperativos seguindo o Padrão Modelo-Visão-Controle. In: SIMPÓSIO BRASILEIRO DE GEOINFORMÁTICA, 7, 2005. Campos do Jordão. **Proceedings**. Campos do Jordão. SP. Novembro 2005. Disponível em: <http://www.geoinfo.info/portuguese/geoinfo2005/programa.html>. Acesso em Dezembro 2005.
- COX, S., CUTHBERT, A., LAKE, R., MARTELL, R. GML Implementation Specification, version. 2.1.2. OpenGIS Implementation Specification. 2002. Disponível em <<http://www.opengeospatial.org/docs/02-069.pdf>> Acesso em Junho 2005.
- CRANE, D; PASCARELLO, E. **AJAX in Action**. Manning Publications, 2005. 680p.
- D'ALGE, J. C. L. Cartografia para Geoprocessamento. In: CÂMARA, G.; DAVIS.C.; MONTEIRO, A.M. **Introdução à Ciência da Geoinformação**. São José dos Campos, INPE, 2001. Disponível em <<http://www.dpi.inpe.br/gilberto/livro/introd/>>.
- DAMJANOVIC-ZIVIC, D; MAJDANDZIC, N. Information System Isvip Used In Planning And Monitoring Of Grape Growing Processes. In: EUROPEAN FEDERATION FOR INFORMATION TECHNOLOGY IN AGRICULTURE CONFERENCE, 3, 2001. Montpellier, France. **Proceedings**. Montpellier, France. 2001. pp 143-146.
- DEEGREE. Free Software for Spatial Data Infrastructures. Disponível em <<http://www.deegree.org/>>. Acesso em Novembro 2006.
- DIAS, L. T.; CÂMARA, G.; DAVIS JR., C. A. Modelos Espaço-Temporais. In Marco Casanova, Gilberto Câmara, Clodoveu Davis, Lúbia Vinhas e Gilberto Ribeiro de Queiroz. **Bancos de Dados Geográficos**. Curitiba: Editora MundoGEO, 2005. p. 137-169.
- DRAGICEVIC, S. The potential of Web-based GIS. **Journal of Geographical Systems**. v. 6, n. 2. Springer-Verlag Heidelberg, p 95-116, junho 2004.
- EDELWEISS, N. Bancos de Dados Temporais: Teoria e Prática. In: JORNADA DE ATUALIZAÇÃO EM INFORMÁTICA, 17, vol. II, 1998, Recife. **Anais**, Recife: Sociedade Brasileira de Computação, 1998. p.225-282.
- EGENHOFER, M. J. Spatial-Query-by-Sketch. IN IEEE SYMPOSIUM ON VISUAL LANGUAGES, 1996, Boulder, USA. **Proceedings**, Boulder, USA, 1996. p 60-67.

EGENHOFER, M. J. Spatial SQL: A Query and Presentation Language. **IEEE Transactions on Knowledge and Data Engineering**. v 6, Issue 1. p:86-95. 1994.

EJB. Enterprise Java Beans. Disponível em <<http://java.sun.com/products/ejb>>. Acesso em Janeiro 2006.

FALLSIDE, D.C., WALMSLEY, P. XML Schema Part 0: Primer Second Edition. W3C Recommendation. 2004.
Disponível em <<http://www.w3.org/TR/xmlschema-0>>. Acesso em Janeiro 2006.

FEGARAS L.; ELMASRI, R. A temporal object query language. In: INTERNATIONAL WORKSHOP ON TEMPORAL REPRESENTATION AND REASONING, 5, 1998, Florida, USA. **Proceedings**. Florida, USA. May 1998.

FERREIRA, K. R.; PAIVA, J.A.C.; CÂMARA, G. Interface para Operações Espaciais em Banco de Dados Geográficos. In: SIMPÓSIO BRASILEIRO DE GEOINFOMÁTICA, 5, 2003, Campos do Jordão. Anais. Campos do Jordão - SP. Novembro 2003.
Disponível em:
<http://www.geoinfo.info/portuguese/geoinfo2003/programa.html>>. Acesso em Junho 2005.

FERREIRA, K. R.; QUEIROZ, G. R.; PAIVA, J. A. C.; SOUZA, R. C. M.; CÂMARA, G. Arquitetura de Software para Construção de Bancos de Dados Geográficos com SGBD Objeto-Relacionais. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, 17, 2002, Gramado. **Anais**. Gramado, Rio Grande do Sul, 2002. p.57-67.

FIDALGO, R.N.; SILVA, J.; TIMES, V.C.; SOUSA, F.F.; BARROS, R.S.M. GMLA: A XML Schema for Integration and Exchange of Multidimensional-Geographical Data. In: SIMPÓSIO BRASILEIRO DE GEOINFOMÁTICA, 5, 2003, Campos do Jordão. **Anais**. Campos do Jordão, SP. Novembro 2003. Disponível em:
<<http://www.geoinfo.info/portuguese/geoinfo2003/programa.html>>. Acesso em Junho 2005.

FINGAR, P. Component-Based *Frameworks* for E-Commerce. **Communications of the ACM**. v. 43, n. 10. October 2000.

FITZKE, J; GREVE, K; MÜLLER, M.; POTH A. Building SDIs with Free Software - The Deegree Project. In: CONF. GLOBAL SPATIAL DATA INFRASTRUCTURE, 7, 2004, Bangalore. **Proceedings**. Bangalore, India. 2004.

FOWLER, M. **Analysis Patterns: Reusable Object Models**, Object Technology Series. Addison-Wesley Publishing Company, Reading, 1997.

FRANK, A. MAPQUERY: Data Base Query Language for Retrieval of Geometric Data and their Graphical Representation. In ACM INTERNATIONAL CONFERENCE ON COMPUTER GRAPHICS AND INTERACTIVE TECHNIQUES, 1982, Boston. **Proceedings**. Boston, Massachusetts, USA. July 1982.

FRIIS-CHRISTENSEN, A.; TRYFONA, N.; JENSEN, C.S. Requirements and research issues in geographic data modeling. In ACM INTERNATIONAL SYMPOSIUM ON ADVANCES IN GEOGRAPHIC INFORMATION SYSTEMS, 9, 2001, Atlanta. **Proceedings**. Atlanta, Georgia, USA., 2001. p 2-8.

GAMMA, E. HELM, R. JOHNSON, J. VLISSIDES. Design patterns: Abstraction and reuse in object-oriented designs. In: EUROPEAN CONFERENCE ON OBJECT ORIENTED PROGRAMMING, 1993, Berlin. **Proceedings** Berlin, Springer-Verlag, 1993.

GAMMA, E.; VLISSIDES, J.; JOHNSON, R. **Design Patterns**. Addison Wesley Pub, 1994.

GEYER-SCHULZ, A.; HAHSLER, M. Software Reuse with Analysis Patterns. In: AMERICAS CONFERENCE ON INFORMATIN SYSTEMS. 2002, Dallas. **Proceedings** . Dallas, August, 2002.

GOODCHILD, M.F. Geographical data modeling. **Computers & Geosciences**, London, v.18, n.4. p. 401-408, May 1992.

GORDILLO, S.; BALAGUER, F. Refining an object-oriented GIS design model: topologies and field data. In: ACM INTERNATIONAL SYMPOSIUM ON ADVANCES IN GEOGRAPHIC INFORMATION SYSTEMS, 6, 1998, Washington, D.C., USA. **Proceedings**. Washington, D.C., United States. 1998. pp. 76-81

GORDILLO, S.; BALAGUER, F.; DAS NEVES, F. Generating the architecture of GIS applications with design patterns. In: ACM INTERNATIONAL WORKSHOP N ADVANCES IN GEOGRAPHIC INFORMATION SYSTEMS, 5, 1997, Las Vegas. **Proceedings**. Las Vegas, Nevada, United States, 1997. pp. 30-34.

GRADY R. B. **Pratical Software Metrics for Project Management and Process Improvement**, Prentice-Hall, 1992. Falta local

GRAND, M. **Java Enterprise Design Patterns**: Patterns in Java. Vol 3, John Wiley & Sons. 2001.

GUTING, R. H. An Introduction to Spatial Database Systems. **VLDB Journal**, v.3, n.4, 1994.

HADZILACOS, T.; TRYFONA, N. An Extended Entity-Relationship Model for Geographic Applications. **ACM SIGMOD**. Volume 26 (3), ACM Press, New York, USA. September 1997.

JAKARTA. The Apache Jakarta Project. Disponível em <<http://jakarta.apache.org/>>. Acesso em Janeiro 2006.

JEE. Java Platform, Enterprise Edition (JEE). Sun Microsystems. Disponível em <<http://java.sun.com/javaee/>>. Acesso em Janeiro 2006.

JDBC. Java Database Connectivity. Disponível em <<http://java.sun.com/products/jdbc>>. Acesso em Janeiro 2006.

JENSEN, A.L.; BOLL, P. S.; THYSEN, I.; PATHAK B. K. PI@nteInfo - a *web*-based system for personalised decision support in crop management. **Computers and Electronics in Agriculture**. v.25, p.271-293. 2000.

JSF. Java Server Faces Technology. Disponível em <<http://java.sun.com/javaee/jaserverfaces/>>. Acesso em Janeiro 2006.

KAZMAN, R.; BASS, L. J.; WEBB, M.; ABOWD, G. D. SAAM: A Method for Analyzing the Properties of Software Architectures. In: INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, 16, 1994, Sorrento, Italy. **Proceedings**. Sorrento Italy, 1994, p 81-90.

KAZMAN, R.; KLEIN, M.; CLEMENTS, P. ATAM: Method for Architecture Evaluation. Technical Report CMU/SEI-2000-TR-004, CMU/SEI. 2000.

KORDUAN, P.; BILL, R.; BÖLLING, S. An Interoperable Geodata Infrastructure for Precision Agriculture. In: AGILE CONFERENCE ON GEOGRAPHIC INFORMATION SCIENCE, 7., 2004, Berlin. **Proceedings**. Greece, 2004. p.747-751.

- KOSTERS, G.; PAGEL, B.U. The GEOOA-Tool and Its Interface to Open GIS-Software Development Environments In:ACM WORKSHOP ON ADVANCES IN GEOGRAPHIC INFORMATION SYSTEMS, 4, 1996, Rockville, USA. **Proceedings**. Rockville, USA, November 1996, p. 163-171. Disponível em: <http://www.informatik.fernuni-hagen.de/import/pi3/publikationen/> Acesso em Junho de 2003.
- KOSTERS, G.; PAGEL, B.U.; SIX, H.W. GeoOOA: Object-oriented analysis for geographic information systems. In: INTERNATIONAL CONFERENCE ON REQUIREMENTS ENGINEERING, 2, 1996, Colorado – USA. **Proceedings**. Colorado Springs, USA, April 1996. p. 245-253. Disponível em: <http://www.informatik.fernuni-hagen.de/import/pi3/publikationen/> Acesso em Junho de 2003.
- LAMPARELLI, R. A. C.; ROCHA, J. V.; BORGHI, E. **Geoprocessamento e Agricultura de Precisão**: Fundamentos e Aplicações. Guaíba-RS: Livraria e Editora Agropecuária, 2001.
- LARMAN, C. **Applying UML and Patterns**. An Introduction to Object-Oriented Analysis and Design and the Unified Process. USA. 2002.
- LASSEN, P., BØJER, O.Q., HANSEN, J.G., JENSEN, A.L.; THYSEN, I. Implementation of Internet based Decision Support Systems in Estonia, Latvia, Lithuania and Poland. In: EUROPEAN FEDERATION FOR INFORMATION TECHNOLOGY IN AGRICULTURE CONFERENCE, 4, 2003, Budapeste, Hungria. **Proceedings**. Budapeste, Hungria, 5-9 July. 2003. p. 59-66.
- LIN, H.; HUANG, B. SQL/SDA: a query language for spatial analysis and its *web*-based implementation. **IEEE Transactions on Knowledge and Data Engineering**, volume13 (4), p. 671-682, 2001.
- LISBOA FILHO, J. **Projeto Conceitual de Banco de Dados Geográficos através de Reutilização de Esquemas, utilizando Padrões de Análise e um Framework Conceitual**. 2000. 212 p. Tese (Doutorado) – Instituto de Informática, Universidade Federal do Rio Grande do Sul. Porto Alegre/RS. 2000.
- LISBOA FILHO, J.; IOCHPE, C.; BORGES, K. A. Analysis patterns for GIS data schema reuse on urban management applications. **CLEI Electronic Journal**, v.5, n.2, p.1-15, 2002.

- LISBOA FILHO, J.; IOCHPE C., Specifying Analysis Patterns for Geographic Databases on the basis of a Conceptual *Framework*. In: ACM SYMPOSIUM ON ADVANCES IN GEOGRAPHIC INFORMATION SYSTEMS, 7, 1999, Kansas City, USA. **Proceedings**. Kansas City, USA, 1999.
- LISBOA FILHO, J.; IOCHPE, C.; BEARD, K. Applying analysis patterns in the GIS domain. In: ANNUAL COLLOQUIUM OF THE SPATIAL INFORMATION RESEARCH CENTRE, 10, 1998, Dunedin, New Zealand. **Proceedings**. Dunedin, New Zealand, 16-19 November 1998.
- LISBOA FILHO, J.; PEREIRA, M.A. Desenvolvimento de uma Ferramenta CASE para o Modelo UML-GeoFrame com Suporte para Padrões de Análise. In: SIMPÓSIO BRASILEIRO DE GEOINFORMÁTICA, 4, 2002, Caxambu, Minas Gerais. **Anais**. Caxambu, Minas Gerais. Dezembro 2002. Disponível em: <<http://www.geoinfo.info/portuguese/geoinfo2002/programa.html>>. Acesso em Junho 2005.
- MAINGUENAUD, M.; PORTIER, M.A. CIGALES: a Graphical Query Language for Geographical Information Systems. In: INTERNATIONAL SYMPOSIUM ON SPATIAL DATA HANDLING, 4, 1990, Zurich, Switzerland. **Proceedings**. Zurich, Switzerland, 1990. p 393-404.
- MARINESCU, F. EJB Design Patterns: Advanced patterns, processes, and idioms. John Wiley. 2002.
- MASCARIN, L. S.; MOLIN, J. P. Geração de Mapas de Produtividade para Citrus. In: CONGRESSO BRASILEIRO DE AGRICULTURA DE PRECISÃO, 2, 2006, São Pedro, SP. **Anais**. São Pedro. SP. 04-07 Junho, 2006. (CD-ROM)
- MESZAROS, G.; DOBLE, J. A patterns language for pattern writing. Disponível em <<http://hillside.net/patterns/writing/patterns.htm>>. Acesso em Julho 2006.
- MEYER B. Beyond Icons: Towards New Metaphors for Visual Query Languages for Spatial Information Systems. In: INTERNATIONAL WORKSHOP ON INTERFACES TO DATABASE SYSTEMS, 1, 1993, Glasgow, UK. **Proceedings**. Glasgow, UK, 1993. p 113-135.
- MIGUEL, F. R. M.; VIEIRA, S. R.; GREGO, C. R.; MELO, T. H. Z. Variabilidade Espacial e Temporal das Propriedades Físico-Hídricas do Solo em Função do Tempo de Pastejo. In: CONGRESSO BRASILEIRO DE AGRICULTURA DE PRECISÃO, 2, 2006. São Pedro, SP. **Anais**. São Pedro, SP. 04-07 Junho, 2006. (CD-ROM)

MIRANDA, R.A.V.; BAPTISTA, C.S.; ALMEIDA, R.R.; CATÃO, B.; PAZINATTO, E. IGIS: um *Framework* para Sistemas de Informações Geográficas em N-Camadas usando um SGBD Objeto-Relacional. In: SIMPÓSIO BRASILEIRO DE GEOINFORMÁTICA, 4, 2002, Caxambu- MG. **Anais**. Caxambu, Minas Gerais. Dezembro 2002. Disponível em: <<http://www.geoinfo.info/portuguese/geoinfo2002/programa.html>>. Acesso em Junho 2005.

MURAKAMI, E. **Uma infra-estrutura de desenvolvimento de sistemas de informação orientados a serviços distribuídos para agricultura de precisão**. 2006. 186p. Tese (Doutorado) - Escola Politécnica, Universidade de São Paulo. São Paulo, 2006.

MURAKAMI, E., SARAIVA, A.M., RIBEIRO, L.C.M., CUGNASCA, C.E., HIRAKAWA, A.R., CORREA, P.L.P. An infrastructure for the development of distributed service-oriented information systems for precision agriculture. **Computers and Electronics in Agriculture Journal**. 2006. No prelo.

MURAKAMI, E., RIBEIRO, L.C.M., SARAIVA, A.M. Desenvolvimento De Software Para Agricultura De Precisão Baseado Em Componentes Interoperáveis. In: CONGRESSO BRASILEIRO DA SOCIEDADE BRASILEIRA DE INFORMÁTICA APLICADA À AGRICULTURA E À AGROINDÚSTRIA, 4, 2003, Porto Seguro – BA, Brasil. **Anais**. Porto Seguro-BA, Brasil. 2003. 1 CD-ROM.

MURAKAMI, E., SARAIVA, A.M., CUGNASCA, C.E., RIBEIRO, L.C.M. A Solution for Interoperability of Precision Agriculture Applications using *Web Services*. In: EUROPEAN CONFERENCE ON PRECISION AGRICULTURE, 4, 2003, Berlin, Germany. **Proceedings**. Berlin, Germany, 2003. p.511-512. 1 CD-ROM.

MURAKAMI, E., RIBEIRO, L.C.M., CUGNASCA, C.E., SARAIVA, A.M. An Infrastructure for Development of Information System for Precision Agriculture. In: INTERNATIONAL CONFERENCE ON PRECISION AGRICULTURE AND OTHERS PRECISION RESOURCES MANAGEMENT, 6, 2002, Minneapolis, USA. **Proceedings**. Minneapolis, MN, USA, 2002. p.1712-1722. 1 CD-ROM.

MVC. Model-View-Controller Architecture. Disponível em <http://java.sun.com/blueprints/patterns/j2ee_patterns/model_view_controller/index.html>. Acesso em Novembro 2006.

NÖLLE, O. Geo-Services: Optimizing Agricultural Business Processes. In: EC GI & GIS WORKSHOP, ESDI STATE OF THE ART, 10., 2004, Poland. **Proceedings**. Warsaw, 2004. 8p.

NORVAG, K. Temporal query operators in XML databases. In: SYMPOSIUM ON APPLIED COMPUTING, 2002, Madrid, Spain. **Proceedings**. Madrid, Spain, ACM Press. 2002. p.402 – 406.

OGC. The Open Geospatial Consortium. Disponível em <<http://www.opengeospatial.org/>>. Acesso em Janeiro 2006.

OPENGIS SPECIFICATIONS. OpenGIS Specifications (Standards). Disponível em <<http://www.opengeospatial.org/standards/>>. Acesso em Julho 2006.

PARENT, C.; SPACCAPIETRA, S.; ZIMANYI, E. Spatio-Temporal Conceptual Models: Data Structures + Space + Time. In: SYMPOSIUM ON ADVANCES IN GEOGRAPHIC INFORMATION SYSTEMS, 7, 1999, Kansas City, Missouri, USA. **Proceedings**. Kansas City, Missouri, USA, 1999. p. 26-33.

PARENT, C.; SPACCAPIETRA, S.; ZIMANYI, E.; DOMINI, P.; PLAZANET, C.; VANGENOT, C. Modeling spatial data in the MADS conceptual model. In: INTERNATIONAL SYMPOSIUM ON SPATIAL DATA HANDLING, 1998, Vancouver, Canadá. **Proceedings**. Vancouver, Canada. 1998.

PENG, Z-R; ZHANG C. The roles of geography markup language (GML), Scalable Vector Graphics (SVG), and *Web Feature Service* (WFS) specifications in the development of Internet geographic information systems (GIS). In **Journal of Geographical Systems**. Volume 6, Number 2, Springer-Verlag Heidelberg, p 133-153, Junho 2004.

PERCEPTORY. A conceptual modeling tool for geospatial databases. Disponível em <http://sirs.scg.ulaval.ca/Perceptory/english/download2006_e.asp>. Último acesso em Julho, 2006.

PIRES, F.; MEDEIROS, C.B.; BARROS, A. Modeling Geographic Information Systems using an Object Oriented *Framework*. In: INTERNATIONAL CONFERENCE OF THE CHILEAN COMPUTER SCIENCE SOCIETY, 13, 1993, La Serena, Chile. **Proceedings**. La Serena, Chile, 1993. p217-232.

QUEIROZ, G. R.; FERREIRA, K. R. SGBD com extensões espaciais. In: Marco Casanova, Gilberto Câmara, Clodoveu Davis, Lúbia Vinhas e Gilberto Ribeiro de Queiroz. **Bancos de Dados Geográficos**. Curitiba: Editora MundoGEO, 2005.

- RIBEIRO, L.C.M., SARAIVA, A.M., MURAKAMI, E. jSTF: A J2EE-based Spatio-Temporal *Framework* for Precision Agriculture Information Systems. In: INTERNATIONAL CONFERENCE ON PRECISION AGRICULTURE, 8, 2006, Minneapolis, Minnesota, USA. Proceedings. Minneapolis, Minnesota, USA. 2006. No prelo.
- RIBEIRO, L.C.M., SARAIVA, A.M., MURAKAMI, E. Componentes Espaço-Temporais para Agricultura de Precisão. In: CONGRESSO BRASILEIRO DE AGRICULTURA DE PRECISÃO, 2004, Piracicaba-SP. **Anais**. Piracicaba-SP, 2004. 1 CD-ROM.
- RIBEIRO, L.C.M., SARAIVA, A.M., MURAKAMI, E. Modelagem de Banco de Dados Espaciais para Agricultura de Precisão. In: CONGRESSO BRASILEIRO DA SOCIEDADE BRASILEIRA DE INFORMÁTICA APLICADA À AGRICULTURA E À AGROINDÚSTRIA, 4, 2003, Porto Seguro-BA, Brasil. **Anais**. Porto Seguro-BA, Brasil. 2003. 1 CD-ROM.
- RIBEIRO, L.C.M., SARAIVA, A.M., MURAKAMI, E., CUGNASCA, C.E. A geo-spatial Database for Precision Agriculture Information Systems. In: EUROPEAN CONFERENCE ON PRECISION AGRICULTURE, 4, 2003, Berlin, Germany. **Proceedings**. Berlin, Germany, 2003. p. 543-544. 1 CD-ROM.
- RIBEIRO, L.C.M., MURAKAMI, E., SARAIVA, A.M., CUGNASCA, C.E. Um *Framework* Baseado em Componentes para Sistemas de Informações para Agricultura de Precisão. In: SIMPÓSIO INTERNACIONAL DE AGRICULTURA DE PRECISÃO, 2, 2002, Viçosa, MG. **Anais**. Viçosa, MG, 2002. 1 CD-ROM.
- RIGAUX, P.; SCHOLL, M.; SEGOUFIN, L.; GRUMBACH, S. Building a constraint-based spatial database system: model, languages, and implementation. **Information Systems**, Volume 28 (6), p.563-595, September 2003.
- ROCHA, J. G. P. B. **Informação Geográfica: Meta-Informação, Codificação e Visualização**. 2005. 142p. Tese (Doutorado) -. Departamento de Informática. Escola de Engenharia. Universidade do Minho, Portugal,. pp. 142. 2005. Disponível em: <<https://repositorium.sdum.uminho.pt/bitstream/1822/1243/1/tese.pdf>>. Acesso em Junho 2006.
- ROCHA, L. V. **GeoFrame-T: um Framework Conceitual Temporal para Aplicações de Sistemas de Informações Geográficas**. 2001. 138 p. Dissertação (Mestrado) – Instituto de Informática, Universidade Federal do Rio Grande do Sul. Porto Alegre/RS. Junho 2001.

- ROCHA, L. V.; EDELWEISS, N. GeoFrame-T: A Temporal Conceptual *Framework* for Data Modeling. In: SYMPOSIUM ON ADVANCES IN GEOGRAPHICAL INFORMATION SYSTEMS, 9, 2001, Atlanta, GA, USA. **Proceedings**. Atlanta, GA, USA, November 9-10, 2001. p 47-52.
- RODDICK, J.; EGENHOFER, M.J.; HOEL, E.; PAPADIAS, D.; SALZBERG, B. Spatial, Temporal and Spatio-Temporal Databases - Hot Issues and Directions for PhD Research. **SIGMOD Record** 33(2): 126-131, 2004.
- RUMBAUGH, J.; BLAHA, M. R.; PREMERLANI, W. J.; EDDY, F.; LORENSEN, W. E. **Object-Oriented Modeling and Design**. Prentice-Hall. 1991.
- SALLES, M.A.V.; PIRES, F.; MEDEIROS, C.B.; OLIVEIRA, J. L. Development of a Computer Aided Geographic Database Design System. In: SIMPÓSIO BRASILEIRO DE BANCO DE DADOS, 13, 1998, Maringá-PR, Brasil. **Anais**. Maringá-PR, Brasil, Outubro 1998.
- SANTANA, F. S. ; FONSECA, R. R.; SARAIVA, A. M. ; CORRÊA, P. L. P; PARIENTE, C.A.B.; GIOVANNI, R. openModeller - An open *framework* for ecological niche modeling: analysis and future improvements: In: WORLD CONGRESS ON COMPUTERS IN AGRICULTURE AND NATURAL RESOURCES, 4, 2006, Orlando, EUA. **Proceedings**. Orlando, EUA. 2006. p. 787-792
- SARAIVA, A.M. **Tecnologia da Informação na Agricultura de Precisão e Biodiversidade: Estudos e proposta de Utilização de Web Services para Desenvolvimento e Integração de Sistemas**. 2003. Tese (Livre Docência) – Escola Politécnica, Universidade de São Paulo. São Paulo, 2003. 187p.
- SARAIVA, A.M. **Um Modelo de Objetos para Sistemas Abertos de Informações de Campo para Agricultura de Precisão – MOSAICo**. 1998. 235p. Tese (Doutorado) - Escola Politécnica, Universidade de São Paulo. São Paulo, 1998.
- SARAIVA, A.M.; MASSOLA, A.M.A.; CUGNASCA, C.E. An Object Model for Field Information Systems. In: INTERNATIONAL CONFERENCE ON PRECISION AGRICULTURE, ASA/CSSA/SSSA. 1998, Saint Paul. **Proceedings**. Saint Paul. July 19-22, 1998. p. 1355-1366.
- SPACCAPIETRA, S.; PARENT, C.; ZIMANYI, E. Modeling Time from a Conceptual Perspective. In: INTERNATIONAL CONFERENCE ON INFORMATION AND KNOWLEDGE MANAGEMENT. 1998, Bethesda, Maryland, USA. **Proceedings**. Bethesda, Maryland, USA. 1998. p. 432-440.

- SHEKHAR, S.; COYLE, M.; GOYAL, B.; LIU, D-R.; SARKAR, S. Data models in geographic information systems. In: **Communications of the ACM**, v40 n4 p103-111. April 1997.
- SILVA, A. B. **Sistemas de Informações Geo-referenciadas: Conceitos e Fundamentos**. Campinas/SP: Editora Unicamp, 1999. 236p.
- SILVA, C. A. B.; PEREZ, R.; FERNANDES, A. R.; MACHADO, J. V. Agriventure: A decision Support System for Agroindustrial Project Preparation and Evaluation. In: EUROPEAN FEDERATION FOR INFORMATION TECHNOLOGY IN AGRICULTURE CONFERENCE, 4, 2003, Budapeste, Hungria. **Proceedings**. Budapeste, Hungria, 5-9 July. 2003. p. 67-72.
- SINGH, I.; STEARNS, B.; JOHNSON, M. Designing Enterprise Applications with the J2EE Platform (Paperback). Enterprise Team, 2002. Disponível em <http://java.sun.com/blueprints/guidelines/designing_enterprise_applications_2> e/>
- SNODGRASS, R.; AHN, I. A Taxonomy of time in databases. IN: ACM SIGMOD INTERNATIONAL CONFERENCE ON MANAGEMENT OF DATA. Texas. May, 1985. **Proceedings**. May, 1985 p. 236-246.
- SODRE, V. F.; LISBOA FILHO, J.; VILELA, V.M.; ANDRADE, M.V.A. Improving Productivity and Quality of GIS Databases Design using an Analysis Pattern Catalog. In Asia-Pacific Conference on Conceptual Modelling, 2, 2005. Newcastle, Australia. **Proceedings**. Newcastle, Australia. 2005. p. 107-114.
- SOURCEFORGE.NET. SourceForge.net Site Documentation. Open Source Technology Group. Disponível em <<http://sourceforge.net/docs/about>>. Acesso em Janeiro 2006.
- STRUTS. Struts *framework*. The Apache Software Foundation. Disponível em <<http://struts.apache.org>>. Acesso em Janeiro 2006
- SVG. Scalable Vector Graphics (SVG) 1.1 Specification. 2003. Disponível em <<http://www.w3.org/TR/SVG>>. Acesso em Junho 2005.
- SVINTERIKOU, M.; THEODOULIDIS, B. TUML: A Method for Modeling Temporal Information Systems. In: INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION ENGINEERING, 11, 1999, Heidelberg/Germany. **Proceedings**. Heidelberg/Germany, June 16-18, 1999.

TERRALIB. GIS Classes and Functions Library. Disponível em <<http://www.dpi.inpe.br/terralib/>>. Acesso em Janeiro de 2006.

TRYFONA, N. Modeling Phenomena in Spatiotemporal Databases: Desiderata and Solutions. In: INTERNATIONAL CONFERENCE ON DATABASE AND EXPERT SYSTEM APPLICATION, 9, 1998, Berlin. Proceedings. Berlin, 1998.

VINHAS, L.; FERREIRA, K. R. Descrição da TerraLib In: In Marco Casanova, Gilberto Câmara, Clodoveu Davis, Lúbia Vinhas e Gilberto Ribeiro de Queiroz. **Bancos de Dados Geográficos**. Curitiba: Editora MundoGEO, 2005. p. 397-440.

XSLT. XSL Transformations Version 1.0 Specification. 1999. Disponível em <<http://www.w3.org/TR/xslt>>. Acesso em Julho 2006.