Pontifícia Universidade Católica de Minas Gerais
Pós-Graduação em Geografia - Tratamento da Informação Espacial

Aplicativos Computacionais em Geografia: Uma Proposta Metodológica Para Visualização,Tratamento e Análise de Informações Espaciais em Ambiente Web

Pasteur Ottoni de Miranda Junior

Belo Horizonte, 2009

Pontifícia Universidade Católica de Minas Gerais

Pós-Graduação em Geografia - Tratamento da Informação Espacial

Aplicativos Computacionais em Geografia: Uma Proposta Metodológica Para Visualização,Tratamento e Análise de Informações Espaciais em Ambiente Web

Tese apresentada ao Programa de Pós-Graduação em Geografia – Tratamento da Informação Espacial da Pontifícia Universidade Católica de Minas Gerais como requisito parcial à obtenção do Título de Doutor.
Área de Concentração: Análise Espacial
Orientador: Prof. João Francisco de Abreu, PhD
Doutorando: Pasteur Ottoni de Miranda Junior

Belo Horizonte, 2009

FICHA CATALOGRÁFICA
Elaborada pela Biblioteca da Pontifícia Universidade Católica de Minas Gerais

Miranda Junior, Pasteur Ottoni de
M672a Aplicativos computacionais em geografia: uma proposta metodológica para visualização, tratamento e análise de informações espaciais em ambiente web / Pasteur Ottoni de Miranda Junior. - Belo Horizonte, 2009.
465 f.: il.

Orientador: João Francisco de Abreu
Tese (Doutorado) – Pontifícia Universidade Católica de Minas Gerais. Programa de Pós-Graduação em Tratamento da Informação Espacial.
Bibliografia.

1. Sistemas de informação geográfica. 2.Análise espacial (estatística). 3. Armazenamento de dados. 4. Engenharia de software. 5. Arquitetura de software. I. Abreu, João Francisco de. II. Pontifícia Universidade Católica de Minas Gerais. Programa de Pós-Graduação em Tratamento da Informação Espacial. III. Título.

CDU: 91:681.3

Bibliotecária Erica Fruk Guelfi – CRB 6/2068

PASTEUR OTTONI DE MIRANDA JUNIOR

Aplicativos Computacionais em Geografia: Uma Proposta Metodológica Para Visualização,Tratamento e Análise de Informações Espaciais em Ambiente Web

Tese apresentada ao Programa de Pós-Graduação em Geografia – Tratamento da Informação Espacial da Pontifícia Universidade Católica de Minas Gerais como requisito parcial à obtenção do título de doutor.

Belo Horizonte, 2009

Avaliada e aprovada por:

_____

Prof. João Francisco de Abreu, PhD (orientador)
Pós-Graduação em Geografia – PUC Minas

_____

Prof. Dr. Leônidas da Conceição Barroso
Pós-Graduação em Geografia – PUC Minas

_____

Prof. Dr. Aurélio Muzzarelli
Universidade de Bolonha-Itália

_____

Prof. Dr. Luis Enrique Zárate
Pós-Graduação em Informática – PUC Minas

_____

Prof. Dr. Marco Túlio de Oliveira Valente
Departamento de Ciência da Computação – UFMG/
Departamento de Ciência da Computação PUC Minas

Ao meu filho Henrique e a minha esposa Daniela.
Ao meu pai Pasteur, mãe Maria Izabel, avó Geralda e avô Samuel (in memoriam).

Agradecimentos

Ao Professor João Francisco de Abreu, o apoio, paciência e sabedoria na orientação deste trabalho.

Ao meu filho Henrique, a minha esposa Daniela e aos familiares, a compreensão nos momentos de ausência.

Aos meus pais, o ensinar do caminho.

A todos os professores do Programa de Pós-Graduação em Geografia-Tratamento da Informação Espacial da PUC Minas, o excelente embasamento, amizade, apoio e capacidade de transmissão de informações.

Às agências financiadoras, CNPq e FAPEMIG e ao FIP/PUC Minas, o apoio ao desenvolvimento de algumas partes deste trabalho.

RESUMO

Dados espaciais aumentam as possibilidades de extração de fatos realmente relevantes a partir de informações disponíveis. Entretanto, o desenvolvimento de sistemas que os suportem, bem como sua análise, requerem um conjunto especial de procedimentos e métodos que adaptem as abordagens convencionais a suas particularidades. Neste contexto, este trabalho apresenta uma proposta metodológica para o desenvolvimento de aplicações que envolvam visualização, tratamento e análise de dados espaciais, em um ambiente *web.* Para atingir este objetivo, propõe-se inicialmente um processo de desenvolvimento de software dividido em etapas, realizadas por disciplinas técnicas e gerenciais compostas por atividades que, por sua vez, são constituídas de tarefas que as detalham. Complementando o processo de desenvolvimento, propõe-se uma infraestrutura arquitetural de software para aplicações *web,* constituída por uma camada de apresentação, que contém módulos para construção de interfaces com o usuário, uma camada de negócios, que contém módulos que definem as regras de negócio da aplicação e uma camada de persistência, constituída por módulos para implementação do acesso a classes de domíno do problema e a banco de dados. Um conjunto extra de módulos é também definido para realização de operações de extração, transformação e carga de dados convencionais e espaciais. A base estrutural de armazenamento de dados é a estrutura multidimensional de *data warehouses*, com extensões espaciais. Esta proposta é suportada por algumas ferramentas computacionais de apoio, que auxiliam na organização e aceleram o desenvolvimento de aplicativos. Finalmente, apresentam-se aplicativos computacionais para análise de dados espaciais, com os quais podem-se avaliar diversos aspectos relevantes da proposta e ressaltar a importância do uso de ferramentas específicas para o tratamento de tais de dados.

Palavras Chaves: Análise Espacial, Sistemas de Informações Geográficas, *Data Warehouse* Espacial*,* Processos de Engenharia de Software, Arquitetura de Software.

ABSTRACT

Spatial data increase the possibility of extracting real relevant facts from available information. However, the development of systems that support and analyse such data, requires a special set of procedures and methods to adapt the conventional approaches to its peculiarities. In this context, this work presents a methodology applicable to the development of applications involving visualization, processing and analysis of spatial data in a web environment. To achieve this goal, the thesis initially proposes a software development process divided into stages, carried out by technical and management disciplines consisting of activities that in turn consist of tasks that detail them. Complementing the development process, this work proposes an architectural infrastructure software for web applications, consisting of a presentation layer, which contains modules for building user interfaces, a business layer, which includes modules that define the application business rules and a persistence layer, consisting of modules for implementing problem domain and database access classes. An extra set of modules is also defined for carrying out the extraction, processing and storage of conventional and spatial data. The structural basis of data storage is the multidimensional structure of *data warehouses*, with spatial extensions. This proposal is supported by some computational tools to help the organization and speed-up the development of applications. Finally, this work presents some computer applications for spatial data analysis, from which it is possible to assess various aspects of the proposal and emphasize the importance of using specific tools for the treatment of such data.

Key words: Spatial Analysis, Geographic Information Systems, Spatial Data Warehouse, Software Engineering Processes, Software Arquitecture.

Sumário

Lista de Figuras

Lista de Tabelas

SIGLAS

API-*Application Program Interface*

ASB – *Adapter Session Bean*

CRS-*Coordinate Reference System*

CVS – *Control Version System*

CWM-*Common Warehouse Metamodel*

CWM-*G -Common Warehouse Metamodel -Geographic*

EJB-*Enterprise Java Beans*

EMF- *Eclipse Modeling Framework*

EPF-*Eclipse Process Framework*

ETL – Extração, Transformação, Carga

GIS- Sistema de Informações Geográficas

GML-*Geographic Markup Language*

HTTP – *Hyper Text Transfer Protocol*

ISB –*Interface Session Bean*

JEE-*Java Enterprise Edition*

JPA- *Java Persistence API*

JSF-*Java Server Faces*

JSP-*Java Server Pages*

OCS – *OpenGIS Catalog Services*

OGC-*Open Geospatial Consortium*

OLAP-*Online Analytical Process*

OMG – *Objetct Managemente Group*

RUP-*Rational Unified Process*

UML-*Unified Modeling Language*

W3C – *World Wide Web Consortium*

WCS- *Web Coverage Service*

WCTS-*Web Coordinate Transformation Service*

WFS- *Web Feature Service*

WMS- *Web Map Service*

XMI - *XML Metadata Interchange*

XML- *Extensible Markup Language*

WTS-*Web Terrain Service*

1 Introdução

No mundo atual, a informação tornou-se um componente de alto valor agregado em qualquer área. A universalização da internet multiplicou em muito o conteúdo acessível e, graças a sua topologia, a um número grande de pessoas distribuídas pelo globo. Esta informação encontra-se disponível de maneira bastante heterogênea e, principalmente, desorganizada. Ainda que democrático, o acesso a esta informação necessita de mecanismos e ferramentas que o permitam e que, principalmente, o organizem de forma que possa ser melhor interpretado. Ferramentas de busca como o Google atingem parcialmente este objetivo, já que encontram a informação, mas falham no que diz respeito à organização da mesma.

No universo empresarial, por exemplo, modelos relacionais de bancos de dados permitem o armazenamento e o controle de dados empresariais e são uma prestimosa fonte de informações que podem e devem ser utilizadas no gerenciamento, controle e no planejamento da rotina de uma empresa. Sistemas desta natureza são formalmente denominados *transacionais.* Entretanto, a tomada de decisões exige que a informação se apresente de forma a evidenciar os pontos realmente relevantes e, principalmente, a tornar o acesso a essas informações mais eficiente. Nesse ponto, os sistemas transacionais são deficientes, pois são baseados em modelos relacionais de bancos de dados, inadequados tanto em nível de forma como de disponibilização do conteúdo, pois normalmente estão normalizados, o que reduz o desempenho de pesquisas e buscas em grandes bases de dados. Existe uma categoria especial de sistemas, os chamados *data warehouses,* que são bastante adequados quando se deseja relevar as informações importantes a conceder *rapidez* ao acesso às mesmas. Consegue-se isto se estruturando dados em *modelos multidimensionais,* não normalizados e muito mais adequados ao processo de evidenciar *fatos importantes* no domínio considerado, à luz de *dimensões* que permitam a análise sob diversos pontos de vista e perspectivas. Atinge-se assim, um aumento considerável da capacidade de

se extrair semântica da informação, diminuindo a necessidade de altos níveis de cognição no processo de interpretação da mesma, facilitando, por conseguinte, a capacidade de tomada de decisões estratégicas.

O processo de obtenção de dados para um *data warehouse* convencional é constituído por atividades de extração(*extraction)* de dados provenientes de várias origens, transformação dos mesmos (*transformation)* e carga (*loading)* em um banco de dados multidimensional (estas três etapas são conhecidas como ETL). Uma vez realizadas as etapas de ETL, pode-se então realizar a análise multidimensional dos dados (denominada OLAP- *online Analytical Process*) na qual fatos são analisados à luz das diversas dimensões (principalmente o tempo), ou seja, grandes volumes de dados podem ser agregados e detalhados de maneira rápida e eficiente, sem os possíveis *Joins* e encadeamentos exigidos em modelos de dados relacionais.

A localização e a distribuição da informação em um espaço geográfico podem aumentar significativamente a capacidade de se analisar tal informação. Os chamados *Sistemas de Informações Geográficas* (GIS) são ambientes desenvolvidos para analisar e visualizar informações, do ponto-de-vista espacial, informações estas normalmente armazenadas em tabelas de bancos de dados relacionais. A *análise espacial* permite analisar a informação em função de sua localização no espaço, combinando ferramentas matemáticas, estatísticas, geométricas, de cálculo numérico, dentre outras. Em nível estratégico, informações exibidas em mapas, têm em muito aumentado seu valor, na medida em que a dimensão espacial permite determinar com maior precisão os focos de atenção, pela visualização de possíveis padrões oriundos da distribuição espacial. Ao integrar dados do mundo real provenientes de diversas fontes e formatos, um GIS permite realizar análises complexas, colocando tais dados em um novo contexto: o *georreferenciado.*

Portanto, combinar as vantagens de um *data warehouse* com as de um GIS apresenta-se como uma possibilidade bastante interessante: a estruturação das informações em um modelo multidimensional, com as capacidades de visualização

e análise espacial proporcionadas pelos GIS. Chega-se assim, ao conceito de *data warehouse* espacial , que traz, em última instância, um aumento da capacidade de análise, por permitir, graças à dimensão espacial, um incremento na capacidade de percepção de informações dos dados presentes no *data warehouse* não percebidas por métodos convencionais, ou seja, a localização geográfica torna-se um importante e útil instrumento à análise das informações, no que tange à distribuição da mesma no espaço (Braga, 2004).

1.1 Objetivos

Esta tese tem como objetivo principal o desenvolvimento de uma proposta metodológica para o desenvolvimento de aplicações para visualização, tratamento e análise de informações espaciais, em ambiente *web,* em diversas de suas etapas. Para o alcance deste objetivo geral, definem-se os seguintes objetivos específicos:

- Formalização de processo, no qual destacam-se as principais atividades, tarefas e artefatos necessários ao desenvolvimento de sistemas para tratamento e análise de informações espaciais na *web.* Este processo foi formalizado utilizando-se o *EPF, Eclipse Process Framework.*
- Definir uma infraestrutura arquitetural de software, embasada nas seguintes características desejáveis:
    - Possuir estrutura aberta, através de interfaces adaptáveis (*adapters)* que provejam Interfaces de Programação de Aplicações (*Application Program Interfaces* –APIs) universais, inalteráveis, tornando o arcabouço independente de implementações específicas de algoritmos ou tecnologias;
    - Possuir suas interfaces devidamente documentadas em páginas *JavaDoc* (páginas HTML para documentação de classes, interfaces, atributos e métodos em linguagem Java), para acesso local , ou *via web services* a estruturas e funcionalidades do arcabouço proposto.

- Visualização de informações na forma de relatórios, gráficos e mapas, que permitam o acesso fácil a estas informações via *web*;
- Aderência aos padrões do *Open Geoespatial Consortium-OGC*
- Desenvolver alguns aplicativos para análise espacial para ilustrar a implementação das interfaces definidas;
- Desenvolver um aplicativo-piloto para análise de dados, com informações provenientes do *Ranking Acadêmico Mundial de Universidades* da Universidade de Shangai, para demonstrar a viabilidade da infraestrutura proposta e evidenciar como o uso de tal tecnologia melhora a capacidade de visualização e análise de informações.

A Figura 1 abaixo evidencia os principais componentes metodológicos e artefatos produzidos nesta tese.

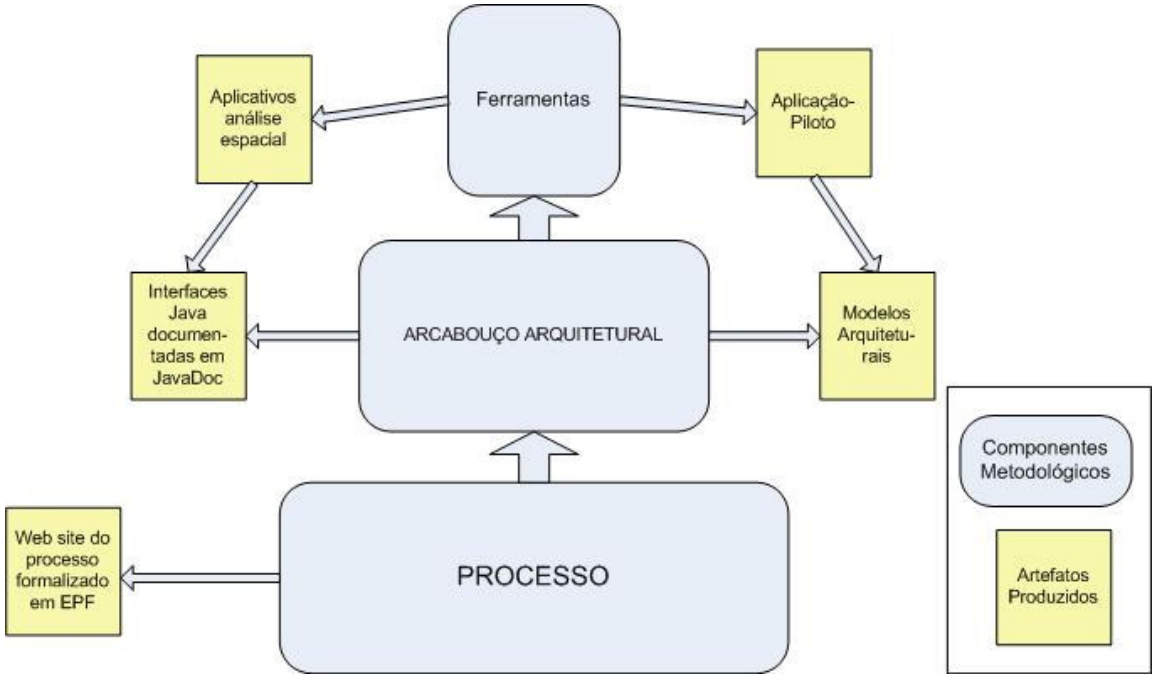


Figura 1-Componentes metodológicos e artefatos produzidos na tese

1.2 Motivação

GIS convencionais (como ARCGIS, MAPINFO, Autocad Map, dentre outros) oferecem boas plataformas para desenvolvimento de mapas e análise de dados. Entretanto, apresentam algumas limitações ao desenvolvimento de GIS específicos, executáveis em ambiente *web.* Pesquisando-se a bibliografia existente na área, percebem-se enfoques em aspectos isolados do desenvolvimento de GIS, dentre os quais podemos citar:

- (Braga, 2004) apresenta um arcabouço baseado no conceito de Tipos Abstratos de Dados para criação de *data warehouses* espaciais.
- (Fidalgo, 2005) define um metamodelo e uma arquitetura de software para desenvolvimento de *data warehouses* espaciais baseado no metamodelo para *data warehouses* CWM (*Common Warehouse Metamodel)* (OMG, 2003 e Poole,2003)
- (Stefanovic, 1993) propõe métodos para OLAP (*On-Line Analytical Processing)* espacial, por meio estruturas formais para dimensões e medidas espaciais.
- (Sampaio, 2006) propõe um modelo multidimensional para *data warehouses* espaciais calcado em bases de dados objeto-relacionais.

Por outro lado, o advento de tecnologias para disponibilização de informações geográficas na *web,* como *Google Maps, Microsoft Maps, Geoserver,* dentre outras, abre um número grande de possibilidades de desenvolvimento de GIS sobre tais plataformas, em diversos segmentos. Junte-se a isso, o aumento da capacidade de acesso a informações, em especial, as georreferenciadas, e a existência de um número expressivo de técnicas para análise de tais informações. A tríade *informação-visualização-análise* é parte importante deste contexto e constitui um ponto-de-vista bastante atrativo.

Entretanto, são poucos os trabalhos que tratam e integram um grande número de aspectos relevantes ao desenvolvimento de GIS específicos em ambiente *web,* com recursos de análise e visualização poderosos*,* em especial, no que tange ao

processo de desenvolvimento e à integração de métodos, arquiteturas e ferramentas. Assim sendo, a hipótese central desta Tese é que o desenvolvimento bem norteado de tais sistemas consegue-se através da criação de um processo definido, suportado por uma infraestrutura arquitetural de software e por ferramentas de apoio a este desenvolvimento (ver Figura 1).

1.3 Roteiro Metodológico

O desenvolvimento desta Tese baseou-se nas seguintes etapas:

- Pesquisa bibliográfica de *data warehouses* convencionais: estrutura básica, extração, transformação e carga de dados, OLAP.
- Pesquisa bibliográfica sobre GIS;
- Pesquisa bibliográfica de *data warehouses* espaciais: alterações a serem feitas no modelo convencional de forma a adaptá-lo ao espacial e sua relação com GIS.
- Pesquisa bibliográfica sobre algoritmos de mineração de dados convencionais e sua adaptação a dados espaciais.
- Estudo de ferramentas de geovisualização, em especial, o *Google Maps.*
- Estudo de modelos e serviços do *Open Geospatial Consortium-OGC.*
- Estudo de ferramentas para desenvolvimento de artefatos no ambiente: *Eclipse Process Framework (EPF),* para definição do processo e *Eclipse Modeling Framework (EMF),* para definição de modelos.
- Com base na pesquisa e nos estudos efetuados, definição do processo de desenvolvimento: disciplinas, etapas, atividades, tarefas ,artefatos e papéis;
- Definição e documentação da infra-estrutura arquitetural de software em modelos de classes da UML- *Unified Modeling Language.*
- Definição de interfaces para todas as funcionalidades do processo, segundo características citadas no item 1.1.

- Documentação de tais interfaces em *JavaDoc.*
- Desenvolvimento de alguns aplicativos sobre as interfaces já definidas.
- Desenvolvimento da aplicação-piloto: *Ranking Acadêmico Mundial de Universidades*

1.4 Organização do Texto

Na introdução apresentam-se os objetivos, justificativa e a metodologia utilizada na confecção deste texto. O capítulo 2 apresenta conceitos teóricos da Geografia, relacionando-os às particularidades da perspectiva espacial. O capítulo 3 descreve a proposta metodológica para visualização, tratamento e análise de informações espaciais em ambiente web, através da apresentação de um processo de desenvolvimento e de uma arquitetura de software adequados aos requisitos de tais aplicações. Neste capítulo apresentam-se também pressupostos teóricos que nortearam o desenvolvimento da proposta: a base estrutural para o armazenamento de informações, *data warehouses* convencionais e espaciais, a estrutura do *Open Geospatial Consortium,* arcabouço arquitetural utilizado neste trabalho para representação de objetos espaciais e serviços de mapa e manipulação de tais objetos, conceitos relacionados à API do *Google Maps* e alguns padrões de projeto. O capítulo 4 apresenta aplicativos desenvolvidos com o uso de recursos providos pela infraestrutura proposta. Finalmente a conclusão do trabalho apresenta observações sobre a proposta e aplicativos desenvolvidos, sugestões de implementações e possíveis aprimoramentos. O apêndice A apresenta um detalhamento do processo de desenvolvimento, extraído do *web site* desenvolvido nesta tese. O apêndice B apresenta um detalhamento da infraestrutura arquitetural, exibindo-se os diversos módulos da mesma em diagramas de classes. O apêndice C apresenta as APIs (*Application Program Interface),* em formato Javadoc, de algumas funcionalidades relacionadas à infraestrutura, métodos de análise espacial e de análise de dados convencionais e espaciais.

2 Revisão Bibliográfica- Antecedentes históricos

Segundo (Abreu, 2005), denomina-se "Nova Geografia" ao processo histórico de formação e consolidação da Geografia Acadêmica ao final do século XIX e início do século XX, o qual se encontra associado ao preenchimento de três condições essenciais:

- A existência de um conjunto de conceitos ou imagens comuns aceitas pelos membros da profissão como meio de formular questões e perseguir respostas.
- A formação de meios institucionais acadêmicos e profissionais dedicados à disseminação dos conceitos e métodos aceitos pela comunidade profissional.
- A atribuição de treino acadêmico aos novos profissionais e membros da comunidade profissional.

Desta feita, no início do século XX, foi fundada a *Association of American Geographers-AAG,* através do esforço de Willian Morris Davis. Formaram-se também os primeiros departamentos acadêmicos, sendo pioneiro o da Universidade de Chicago (1903). Este departamento notabilizou-se na formação de uma nova geração de geógrafos, dentre os quais destacam-se Frederick V. Emerson, que defendeu a primeira tese de doutorado em Geografia, na área de Geografia Urbana, Walter S. Tower, Bailley Willis e Ellen C. Semple. Seus estudantes de pós-graduação tornaram-se os principais líderes da comunidade geográfica norte-americana no período logo após a Primeira Guerra Mundial. Nesse grupo destacam-se Charles C. Colby, Wellington D. Jones, Stephen S. Visher, V. C, Finch, Carl O. Sauer, Mary Lanier, Mary Dopp, Mabel C. Stark e L.P. Denoyer.

A partir de sua fundação,a AAG tornou-se um forte centro de confluência das diversas tendências da Geografia norte-americana. Atualmente, sua principal publicação, *Annais of the Association of American Geographers,* constitui uma das mais prestigiosas publicações em Geografia.

Durante as décadas de 1910 e 1920, a Nova Geografia norte-americana assistiu à formação de importantes centros acadêmicos, dos quais detacam-se:

- Universidade Michigan – Departamento de Geografia e Geologia (1915), dirigido inicialmente por Carl. O. Sauer.
- Clark University – Em 1921 é fundada a *Graduate School of Geography* .
- Universidade da California em Berkeley – Fundado o Departamento de Geografia em 1923.
- Universidade de Minnesota – Departamento de Geografia criado em 1925.
- Universidade de Wiscosin – Organiza seu departamento de Geografia em 1928.

Durante a década de 1920, um grupo de sociólogos urbanos composto por Robert E. Park, Ernest W. Burgess e R. D. Mackenzie, pertencentes à Escola de Chicago, voltaram sua atenção para a distribuição espacial dos fenômenos sócio-urbanos a partir de esquemas teóricos provenientes da Ecologia. Para eles, as características da organização das cidades são encontradas na identificação da mobilidade espacial e social, nas relações das estruturas familiares, nos enclaves sociais , na existência de uma divisão do trabalho e na especialização (Capel & Urteaga, 1987).

Partindo do princípio de que as coletividades humanas se especializam em zonas da estrutura intra-urbana por meio de processos de simbiose, sucessão, invasão, dentre outros, Burgess, em 1925, teorizou que tal especialização zonal poderia ser descrita através de um modelo de áreas concêntricas, em que o lugar central seria ocupado pelo distrito comercial, circundado por anéis de uso residencial. Este modelo muito influenciou pesquisas posteriores sobre a organização do espaço urbano, nos moldes como essa se desenvolveu no seio da Geografia Teorético-quantitiva. Entretanto, este modelo não oferece um quadro completo da distribuição das zonas urbanas. Dentre os modelos que procuram alternativamente auxiliar na compreensão dos processos de diferenciação da estrutura funcional intra-urbana das cidades atuais, destacam-se os modelos de Núcleos Múltiplos e de Campos Urbanos (Hartshorn, 1992). O Modelo de Núcleos Múltiplos baseia-se na premissa

de que o crescimento urbano não se realiza ao redor de um único núcleo central, mas sim ao redor de diversos pontos capazes de polarizar áreas residenciais satélites, mais ou menos independentes do núcleo central. O Modelo de Campos Urbanos, por sua vez, sugere uma organização espacial da estrutura funcional intra-urbana fundada na emergência de grandes setores periféricos organizados ao redor de centros específicos, praticamente independentes do núcleo urbano mais tradicional. Cada um desses setores cria um "campo urbano" autônomo no interior do tecido urbano metropolitano cuja extensão, características e estrutura interna encontram-se condicionados pelo seu sítio, pela dimensão da área metropolitana, pela intensidade das atividades econômicas nele desenvolvidas e pela diferença de acessibilidade entre o seu centro de mercado e o hipercentro metropolitano.

Desde seus primeiros tempos, a Geografia Teorético-Quantitativa norte-americana concentrou-se sobre temas relacionados à distribuição espacial da atividade econômica. A Geografia econômica muito se beneficiou de trabalhos relacionados à Economia Regional, em especial, no campo das teorias locacionais da atividade econômica, dentre os quais merecem especial destaque J.H. Von Thünen, Walter Chritaller, August Lösch e Alfred Weber. A essas influências somam-se os trabalhos de Edgard M. Hoover, *The Location of Economic Activity* e Walter Isard, *Location and Space Economy.* Em essência, a nova tendência da geografia econômica buscava a compreensão funcional, procurando ser mais dedutiva e teórica.

Paralelamente às influências destacadas acima, na década de 1950, os trabalhos desenvolvidos no Departamento de Geografia da Universidade de Washington, sob a liderança de William L. Garrison, imprimiram definitivamente uma dimensão espacial à Geografia Urbana e Econômica. Seu estudo passa a se calcar na integração de conceitos relativos a arranjos e distribuições espaciais, à interação espacial e à organização e interação espacial dos fenômenos urbanos e econômicos. Segundo (Abreu, 2005), a denominada revolução teorético-quantitativa por que passou a Geografia norte-americana nas décadas de 1950 e 1960, associa-se à experimentação metodológica promovida pelo grupo de Washington.

A partir de 1955, Garrison passa a oferecer a seus alunos de pós-graduação o primeiro seminário organizado em uma universidade norte-americana com o propósito de propiciar treinamento em métodos matemáticos e estatísticos aplicados à análise espacial. Frequentaram este seminário nomes como os de Brian J. Berry, Willian Bunge, Michael F. Dacey, Arthur Getis, Duane F. Marble, Robert W. Morrill, John D. Nystuen, Waldo Tobler e Torsten Hägerstrand. (Johnston, 1986) observa que Garrison e seu grupo dirigiram seus esforços para o teste de teorias provenientes de outras disciplinas, notavelmente da Economia, e sua aplicação aos problemas de planejamento. Observa também, que desenvolveram enunciados teóricos testáveis, demonstrando possuir forte base matemática. Eles também investigavam testes estatísticos relevantes para sua pesquisa. Das ciências biológicas, adotaram o teste do "vizinho mais próximo" para padrões de pontos e da psicologia, padrões para agrupar e classificar. Em essência, segundo (Johnston, 1986), a motivação denominante do grupo envolvia o desenvolvimento de teorias normativas relevantes, métodos matemáticos e procedimentos estatísticos que servissem para desenvolver leis morfológicas. Dentre os temas de interesse do grupo, destacam-se a análise locacional, o estudo de fluxos econômicos por meio de procedimentos matemáticos de programação linear, a representação matemática de padrões espaciais, a análise do desenvolvimento urbano por intermédio de procedimento de simulação Monte Carlo e a elaboração e aperfeiçoamento de técnicas de transformação cartográfica.

O trabalho do grupo de Washington foi disseminado no período seguinte, quando se dividiu e migrou para diferentes departamentos de Geografia de universidades americanas, em especial:

- Chicago: estudos urbanos e Geografia Econômica;
- Michigan: Geografia Analítica, Métodos Quantitativos;
- Wiscosin: Métodos Quantitativos, Geografia Física;
- Ohio State: Cartografia Analítica, Métodos Quantitativos.

Segundo (Abreu, 2005) a consolidação da tradição espacialista e da análise espacial, entendida como uma abordagem que usa métodos e técnicas matemáticas e estatísticas para derivar propriedades quantitativas do espaço geográfico, decorreu da conjugação de um conjunto de fatores dentre os quais se incluem os antecedentes históricos da Nova Geografia, as contribuições específicas da Sociologia Urbana e da Ecomomia Regional, a influência de importantes estudiosos europeus e a efervecência teórico-metodológica do grupo de Washington. Ressalta-se também o trabalho de geógrafos da Universidade de Iowa (Harold Mccarty, J.C. Hook, D.S. Knos, H.A. Stafford, Fred Schaffer, J.B. Lindberg, E.N. Thomas e L.J. King), os quais, também na década de 1950, ofereceram importantes contribuições para a análise espacial, sobretudo nos campos da Teoria da Geografia, dos Métodos Quantitativos e da Geografia Econômica. (Abreu, 2005) observa que todas essas influências foram agregadas em um momento histórico oportuno, em que recursos governamentais abundantes encontravam-se disponíveis e os geógrafos buscavam novos problemas e novas metodologias, sobretudo no desenvolvimento dos *modelos de análise espacial* e de técnicas de *cartografia analítica*, os quais, posteriormente forneceriam a base teórico metodológica necessária à estruturação dos modernos Sistemas de Informações Geográficas.

Ainda neste contexto, há que se destacar o trabalho de Peter Gould, geógrafo Inglês radicado nos Estados Unidos, que lecionou por 35 anos na Pennsylvania State University. Seus trabalhos demonstram sua constante preocupação em estender ao máximo, sob o ponto de vista espacial, as fronteiras do conhecimento geográfico. Estendem-se dos temas clássicos da análise espacial, como organização e difusão espacial, para alcançar muitos dos aspectos relevantes do comportamento espcial, como demonstra sua pesquisa com mapas mentais, empregados como técnica de estudo da cognição espacial (Abreu, 2005). Tais preocupações levaram Gould a uma reflexão rica sobre os fundamentos epistemológicos da análise espacial.

No campo da Teoria da Geografia, Gould demonstrou a associação existente entre a evolução da perspectiva espacialista e os progressos alcançados na solução de

problemas humanos (Abreu, 2005). Em seu trabalho *Geographers at Work,* ressalta a forma como a perspectiva espacial auxilia na solução de problemas geográficos em um amplo espectro de preocupações humanas.

Peter Gould publicou um amplo espectro de trabalhos, dentre os quais destacam-se:

- Divulgação dos avanços da Geografia Teorético-Quantitativa;
- Estudo empírico da Geografia da Percepção e do Comportamento através do mapeamento da compreensão human sobre espaço;
- Definição do trabalho do geógrafo sob a perspectiva da Nova Geografia;
- Padrões de difusão espacial da AIDS;
- Aplicação de técnicas desenvolvidas para análise do comportamento espacial às microescalas, estruturando as bases metodologias para o estudo da microgeografia do espaço comportamental.

(Sintetizado de Abreu, 2005).

2.1 A Metodologia Científica e a Geografia

O físico italiano Galileu Galilei lançou uma das bases da metodologia científica, através da experimentação. O trabalho de Renée Descartes, *O Discurso do Método,* publicado em 1637, contém as bases teóricas do método científico, que viria a nortear grande parte do pensamento científico posterior. Os princípios básicos do *Discurso do Método* postulam essencialmente que 1)O que não for evidente deve ser testado de forma a verificar sua veracidade; 2) Um problema deve ser dividido em partes a serem analisadas separadamente; 3) A solução de um problema deve partir de objetivos mais simples para se chegar aos mais complexos e 4) O raciocínio deve ser organizado e as premissas enumeradas de forma a se estar certo de nada omitir. Estes princípios, em conjunto com a *verificação* e o método *indutivo,* constituem uma importante vertente da metodologia científica.

A Geografia Teorética-Quantitativa tem a metodologia científica em suas bases e concentra-se no desenvolvimento de uma sólida fundamentação metodológica que, com a sua utilização ao longo dos anos, convergiu em uma experiência prática adquirida na solução de diversos problemas.

(Abreu, 2005) afirma que o conhecimento teórico apresenta uma inerente propriedade sistematizadora, decorrente de serem as teorias corpos de conhecimento capazes de interconectar um grande número de proposições sobre um fenômeno, oferecer uma explicação para proposições e servir de veículo para a formulação de novas hipóteses ao fenômeno em consideração. Desta forma, o conhecimento teórico informa e organiza os conhecimentos metodológico e prático, permitindo assim uma sistematização da busca por solução dos problemas propostos pela realidade.

Imre Lakatos (Lakatos, 1970) oferece um relato da evolução das ciências, no qual não se pode atribuir um caráter de cientificidade ou acientificidade a uma teoria singular, mas sim a uma sucessão de teorias articuladas em torno de um núcleo fundamental. Cada sucessão de teorias científicas assim estruturadas formam um programa de pesquisa, o qual oferece orientação para o desenvolvimento da atividade dos cientistas. Tais programas articulam em torno de seu núcleo fundamental uma *heurística negativa* para orientação dos trabalhos científicos a serem desenvolvidos. Essa heurística negativa consiste na suposição de que as hipóteses teóricas sobre as quais se erguem o corpo teórico devem ser defendidas de tentativas de refutação, por meio de hipóteses auxiliares e pela elaboração de propostas de revisão dos procedimentos de verificação utilizados nos trabalhos de sondagem da realidade (Abreu, 2005). Ao redor do núcleo fundamental de um programa de pesquisa constrói-se um conjunto de indicadores de como o mesmo pode ser desenvolvido. A esse conjunto de indicadores ou sugestões de como desenvolver variáveis refutáveis de um programa de pesquisa, denomina-se *heurística positiva.* Esta contém uma pauta de problemas teóricos a serem resolvidos e conjuntos de propostas experimentais destinadas a permitir os testes necessários à solução das questões teóricas propostas. Ao realizar tais sugestões

teórico-metodológicas, os cientistas constroem ao redor do núcleo fundamental do programa de pesquisa a que se encontram vinculados, uma proteção, de tal forma que todo o trabalho a ser realizado pelos mesmos traduz-se em esforços de expansão e modificação dessa proteção, pela articulação de novas hipóteses e sua submissão a testes com o propósito de falsificação. No contexto de um programa de pesquisa, toda proposta teórica que venha a contrariar as premissas gerais de que se compões o núcleo fundamental, fica excluída.

Segundo (Abreu, 2005), o contexto geográfico espacialista enfatiza a rigorosa aplicação do método científico , com uma preocupação de subordinar a produção do conhecimento geográfico a um princípio de refutabilidade no qual a validade das assertivas formuladas pelos praticantes da disciplina assume a natureza de função da possibilidade de sua verificação e teste. Este acento metodológico centrado na necessidade de maior rigor no enunciado e na verificação de hipóteses, bem como na formulação de explicações para os fenômenos geográficos (Christofoletti, 1985), possui importantes reflexos na escolha das técnicas a serem utilizadas pelos geógrafos.

Outra face extremamente importante da rigorosa adoção da metodologia científica nos trabalhos dos geógrafos espacialistas pode ser identificada no impulso conferido ao desenvolvimento de teorias capazes de oferecer subsídios para a análise espacial (Christofoletti, 1985). Na Geografia Teorético-Quantitativa a teorização é vista como importante veículo para a sistematização do conhecimento geográfico em bases cientificamente rigorosas. Assim sendo, as teorias são concebidas como estruturas do pensamento geográfico capazes de permitir a superação do caráter descritivo e factual dos relatos realizados sob o paradigma regionalista, na medida em que possibilitam a explicitação dos processos, funções e formas relevantes à compreensão do espaço geográfico (Abreu, 2005).

2.2 Os Sistemas de Informações Geográficas

2.2.1 Conceitos

A tecnologia de computadores foi de grande utilidade à geografia. Sua invenção, em meados do século XX, coincidiu com o aparecimento da *quantificação* como resposta à necessidade de processamento de informações, informações estas que estavam sendo produzidas em um ritmo jamais visto. Assim, os computadores e a quantificação formaram uma combinação perfeita, aliando a possibilidade de tratamento mais adequado a estas informações (quantificação) e velocidade de resposta.

Entretanto, apenas velocidade e rapidez nas respostas não são suficientes para o tratamento da informação. É necessário que esta seja *organizada* de forma a permitir sua análise e a visualização de aspectos relevantes. Neste contexto, surgem os *Sistemas de Informações Geográficas (GIS),* sistemas computacionais capazes de organizar a informação espacial, permitindo que seja armazenada, exibida em mapas, tabelas ou gráficos e utilizadas como modelos de estudos geográficos.

GIS têm grande habilidade em tratar grandes quantidades de informação georreferenciada e ainda, em analisar estas informações. Um GIS armazena a localização, a geometria e atributos de dados e pode ser utilizado como ferramenta para produção de mapas, como suporte para análise espacial, como ferramenta para extração, tratamento e georreferenciamento de dados e também como bancos de dados geográficos, muitas vezes com funcionalidades para pesquisa estruturada, armazenamento e recuperação de informação espacial e convencional.

2.2.2 Breve histórico dos GIS

Os primeiros desenvolvimentos apropriados em Matemática para lidar com problemas espaciais começaram por volta dos anos 30 e 40, em paralelo com

desenvolvimentos em métodos estatísticos e análise de séries temporais. O progresso prático efetivo foi completamente bloqueado pela ausência de ferramentas computacionais adequadas. Foi somente após 1950 que, com a disponibilidade do computador digital, floresceram, tanto os métodos conceituais de análise espacial, quanto as reais possibilidades de mapeamento temático quantitativo e análise espacial.

Um grande passo em direção aos primeiros GIS foi dado em 1959, quando Waldo Tobler fez o primeiro mapa em computador do mundo, por meio de 343 cartões perfurados que delineavam o contorno do mapa dos Estados Unidos.

O primeiro GIS que se tem notícia surgiu em 1962 no Canadá (*Canada Geographic Information System-CGIS*) por iniciativa do Dr. Roger Tomlinson, que embora tenha construído os módulos básicos de software, só publicou seus trabalhos uma década depois.

Em 1964, Howard T. Fisher fundou o Laboratório de Computação Gráfica e Análise Espacial na Harvard Graduate School of Design. Nos anos 70, este laboratório desenvolveu e distribuiu versões preliminares softwares de GIS, como o SYMAP, GRID e ODISSEY , que mais tarde serviram de base para o desenvolvimento de GIS comerciais nos anos 80, por inciativa de empresas como Intergraph, Environmental Systems Research Institute (ESRI) e Computer Aided Resourse Information Systems (CARIS) .

No início dos anos 70, os Governos Federais americano, canadense e alguns europeus (Suécia, Noruega, Dinamarca) , apoiaram financeiramente iniciativas voltadas tanto à Cartografia Assistida por Computador (CAC), quanto aos GIS. Foi naquele período que o USGS (*United States Geological Survey*) passou a tornar disponíveis ao público bases de dados digitais, tais como os modelos digitais de elevação ou DEM (*Digital Elevation Models*).

O final da década de 80 e o início da de 90 assistiu ao aparecimento de um grande número de aplicações de GIS, o que se deve, em parte, ao advento e à disseminação dos microcomputadores pessoais, além da introdução de tecnologia

de relativo baixo custo e alta capacidade de performance, tais como as estações de trabalho (*Workstations).*

O final dos anos 90 foi marcado pela consolidação de plataformas mais populares, como ESRI ArcView/ArcGIS e MapInfo.

Os últimos anos assistem à disseminação de GIS em ambiente *web,* principalmente com a popularização de plataformas de mapas digitais na *web,* como o *Google Maps.*

2.2.3 Sistemas de Informações Geográficas- Estrutura

A estrutura geral de um SIG, esquematizada na Figura 2, é constituída pelos seguintes componentes:

a)Interface com o usuário;

b)Entrada e integração de dados;

c)Consulta e análise espacial

d)Visualização e plotagem;

e)Sistemas de Gerenciamento de Bancos de Dados (SGBD) convencionais e geográficos, para armazenamento e recuperação de dados convencionais ou geográficos

No nível do usuário, a interface define como será a interação com o sistema. No nível intermediário, um GIS deve possuir mecanismos de processamento de dados espaciais (entrada, análise, visualização e saída). Em nível mais interno, o SGBD permite o armazenamento e recuperação de informações espaciais e convencionais.

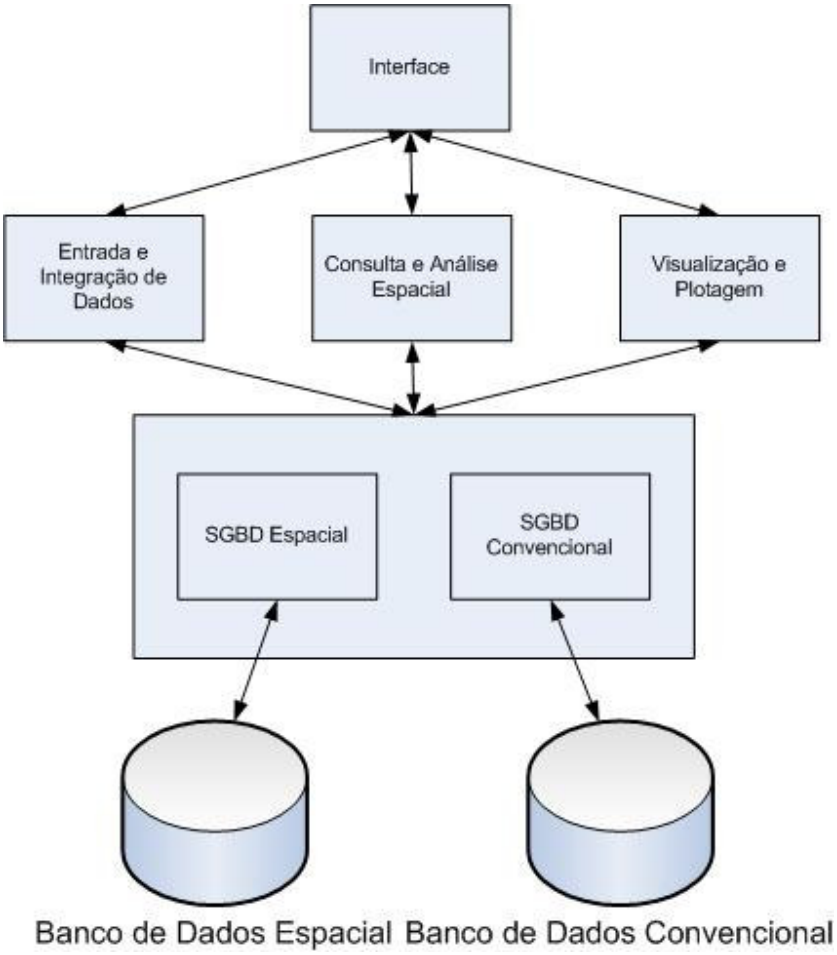


Figura 2.Estrutura geral de um SIG (Inspirado em (Câmara, 1999))

2.2.4 – Modelagem de Informações Geográficas

No projeto de um GIS, modelos descrevem a representação da realidade geográfica, em diversos níveis de abstração. A Figura 3 exibe, segundo (Abreu & Muzzarelli, 2003), os diversos níveis de abstração pelos quais a informação geográfica passa, desde a concepção até sua representação física:

- O modelo conceitual constitui uma representação da realidade em um nível de abstração facilmente inteligível pelo usuário. Captura as necessidades de informação sob o ponto de vista do mesmo.

- O modelo de representação de dados em formato digital transforma o modelo conceitual em um modelo que representa a forma como os dados serão armazenados. Os denominados diagramas entidade-relacionamento, por exemplo, são ferramentas adequadas à representação deste nível.
- O modelo físico abstrai o formato final da informação no formato de arquivos do banco de dados.

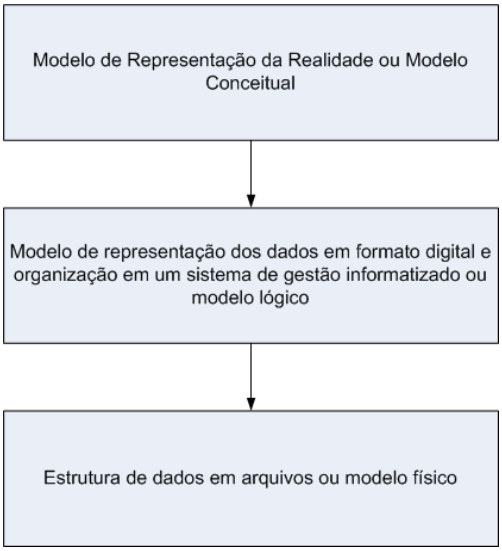


Figura 3-Niveis de abstração de modelos de dados geográficos. Fonte: (Abreu & Muzzarelli, 2003)

2.2.5 O Modelo de classes de geoprocessamento

Segundo (Davis, 2000) , um modelo de classes de geoprocessamento trabalha no nível conceitual/representação e suas classes básicas são Classes Georreferenciadas e Classes Convencionais. Através dessas classes são representados os três grandes grupos de dados (contínuos, discretos e não-espaciais) encontrados nas aplicações geográficas, proporcionando assim, uma visão integrada do espaço modelado, o que é muito importante na modelagem, principalmente de ambientes urbanos.

Uma Classe Georreferenciada descreve um conjunto de objetos que possuem representação espacial e estão associados a regiões da superfície da terra. Uma 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 os proprietários de imóveis cadastrados para fins de tributação (IPTU), e que possuem relação de propriedade com os lotes e edificações presentes no banco de dados geográfico.

A distinção entre classes convencionais e classes georreferenciadas permite que diferentes aplicações possam compartilhar dados não-espaciais, auxiliando no desenvolvimento dessas aplicações e na reutilização dos dados .

Tanto as classes georreferenciadas como as classes convencionais podem ser especializadas utilizando o conceito de herança da orientação a objetos, em classes do tipo Classes Georreferenciadas e em classes do tipo Geo-Campo e Geo-Objeto.

As classes do tipo Geo-Campo representam objetos distribuídos continuamente pelo espaço, correspondendo a grandezas como tipo de solo, topografia e teor de minerais.

As classes do tipo *Geo-Objeto* 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. Por exemplo, um usuário encarregado do gerenciamento de trânsito verá a rua como uma rede direcionada, representando vias de mão simples e dupla; um usuário encarregado do cadastro da cidade, interessado em conhecer os proprietários dos lotes, verá a rua como o espaço entre os meios-fios.

2.2.6 O Processo de Geocodificação

Denomina-se *geocodificação* ao processo de atribuição correta de coordenadas geográficas a objetos *raster* ou vetorial. Um mapa analógico digitalizado por meio de um *scanner,* por exemplo, ao ser importado em um GIS, precisa ter cada ponto de sua imagem associado a coordenadas em um sistema de projeções. Segundo (Abreu & Muzzarelli, 2003), este procedimento estabelece uma correspondência entre coordenadas locais do objeto e as coordenadas do sistema de projeção através de um sistema de equações. Sejam *X e Y* os eixos do sistema de coordenadas de destino e *x* e *y* os eixos do sistema de coordenadas local. O procedimento considera os eixos *X* e *Y* como resultantes de uma translação e de uma rotação dos eixos *x* e *y* de tal forma que:

$X = ax + by + c$

$Y = dx + ey + f$

Onde *a,b,c,d,e,f* são parâmetros incógnitos necessários para resolver o sistema por meio de 6 equações. Portanto, necessita-se conhecer as coordenadas locais e do sistema de projeção de ao menos três pontos, que determinam uma solução única para o sistema. Para um número maior de pontos utiliza-se o método dos mínimos quadrados, o que proporciona uma maior precisão no processo de georreferenciamento.

2.3 A Análise Espacial

2.3.1 Conceitos

A Análise Espacial tem seu surgimento associado à Geografia Teorético-Quantitativa. Os pressupostos teóricos da Revolução Teorético-Quantitativa importaram em uma mudança substancial na reflexão geográfica, que deslocou seu enfoque primário na ordenação de dados para o desenvolvimento de explicações da localização dos fenômenos geográficos, elaborada a partir da construção de um ferramental metodológico adequado à formulação de generalizações acerca da estrutura e da organização do espaço (Christofoletti, 1985). Esta mudança de ênfase teve como conseqüência a estruturação da pesquisa fundamentada 1) na aplicação da metodologia científica, 2) na utilização de teorias, 3) no emprego de técnicas estatísticas e matemáticas, 4) na sistematização da realidade geográfica, 5) no uso intensivo de modelos espaciais (Christofoletti, 1985).

O objetivo maior da Análise Espacial consiste em se determinar *onde* ocorrem os fenômenos, objetivo este completamente alinhado ao *princípio da extensão* do geógrafo alemão Frederich Ratzel (1844-1904), segundo o qual todo fato geográfico deve ser localizado e assim estudado (Abreu, 2005).

A partir da localização dos fenômenos e da determinação dos processos espaciais, a Análise Espacial pode formular um conjunto de questões essenciais voltadas para a adequada compreensão da realidade geográfica (Abreu, 2005):

- *Localização espacial do fenômeno* (onde ocorreu?)
- *Localização temporal do fenômeno* (quando ocorreu?)
- *Determinação e individualização do fenômeno* (o que ocorreu?)
- *Explicação das causas do fenômeno* (por que ocorreu?)
- *A direção futura do fenômeno* (por onde ir?)
- *Análise de tendências nos processos espaciais* (O que mudou?)

- *Determinação de regularidades nos processos espaciais* (Qual o padrão?)
- *Determinação de cenários futuros a partir de condições iniciais* (O que acontece se?)

Além de proporcionar melhor visualização da distribuição geográfica de determinados dados de um problema, informações espaciais podem revelar padrões nem tão óbvios, mas de extrema utilidade à descoberta de relações de causa e efeito e à tomada de decisões. A análise espacial consiste essencialmente em se descobrir padrões na distribuição da informação espacial, com o uso de mapeamentos, técnicas e modelos específicos.

O médico inglês John Snow (1813-1858), no século XIX, foi pioneiro na utilização de procedimentos lógicos, métodos quantitativos e técnicas de análise espacial para correlacionar as ocorrências de mortes por cólera em Londres à água possivelmente contaminada por bombas que abasteciam a cidade. A Figura 4 exibe o mapeamento feito por John Snow, e a maior concentração em torno de bomba de *Broad Street,* região de *Golden Square.*

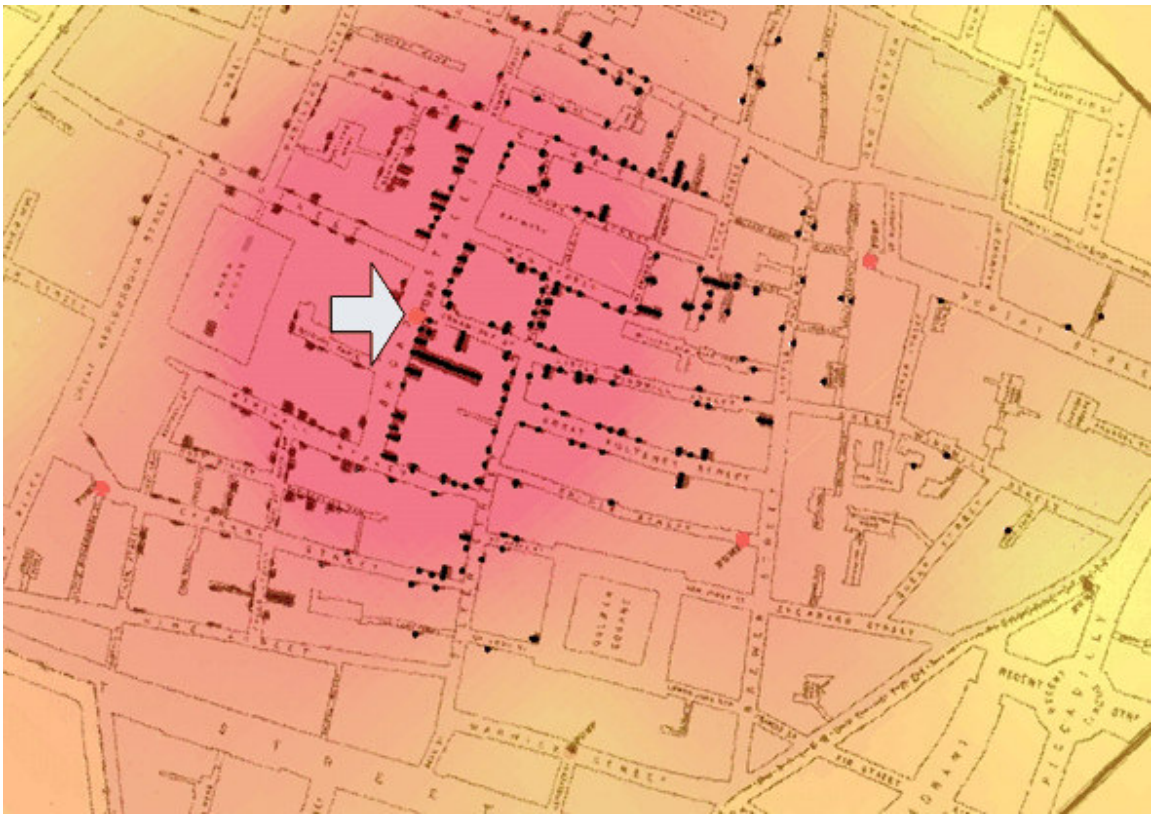


Figura 4-Mapa exibe maior concentração de mortes em torno da bomba de Broad Street, ao centro

Partindo da hipótese de ser a água o meio de transmissão da doença, a suspeita inicial de Snow era de que a origem do surto estava em uma bomba d'água localizada em *Broad Street.* Mas como o aspecto da água observado era normal, não sendo visualizadas quaisquer impurezas, ele não pôde confirmar esta hipótese. Snow obteve então, uma lista de óbitos ocorridos em Londres desde o início da epidemia, nos distritos de *Golden Square, Berwick Street, St. Ann's e Soho.* A partir dessa lista e de entrevistas com parentes das pessoas falecidas, ele estabeleceu a localização exata de cada um dos casos fatais nas proximidades de *Broad Street.* Assim sendo, foi elaborado um mapa indicando a localização dos óbitos associados ao cólera no *Golden Square.* A partir deste mapa, Snow estabeleceu a correlação entre as mortes e a bomba de *Broad Street,* em uma aplicação pioneira e intuitiva da Primeira Lei da Geografia (Tobler, 1970). Posteriormente, Snow conseguiu explicar situações que fugiam ao padrão, como a ocorrência de óbitos próximos a

outras bombas: a água de *Broad Street* era muito apreciada por seu sabor e, por conseguinte era transportada e consumida em outras localidades (Abreu, 2005).

Como se pode observar, John Snow pôde estabelecer padrões e tendências da evolução do processo estudado, seguindo os passos fundamentais dos estudos de localização da Análise Espacial.

2.3.2 Dependência espacial - Primeira Lei da Geografia

Proposta originalmente por Waldo Tobler (Tobler, 1970), enuncia que "tudo se relaciona com tudo, mas coisas próximas são mais relacionadas que coisas distantes". Ou seja, a *dependência espacial* entre informações torna-se mais fraca na medida em que os dados estão mais distantes. Em essência, a Primeira Lei da Geografia formaliza um conceito do senso comum: o de que valores semelhantes de atributos, geograficamente próximos uns dos outros, são um forte indicador de que eles estão relacionados entre si e, principalmente, de que provavelmente existe uma relação de causa-e-efeito entre eles e os eventos, fatos ou fenômenos que os produzem. Alguns exemplos:

- A ocorrência maior crimes com características semelhantes em uma região específica é um indicador de que ali provavelmente existe algum fenômeno ou fato causador (tráfico de drogas, menor renda, desemprego, ou ainda ação de alguém que mora nas proximidades).
- Na instalação de um estabelecimento comercial, a chance de sucesso é maior, por exemplo, em um local onde o número de pessoas que utilizam seus serviços é maior, ou ainda, mais próximo de seus fornecedores.
- No caso clássico do cólera em Londres (1854) Jonh Snow detectou o provável foco de contaminação a partir da proximidade de ocorrência das mortes, de uma bomba localizada em *Broad Street.*

É uma importante base para a aplicação de técnicas de análise espacial que exploram o conceito de proximidade e dependência espacial, exibindo a correlação entre valores no espaço. Encontra-se intimamente associado ao conceito de *distância*, detalhado no próximo item, na medida em que é esta que determina a proximidade entre os fenômenos geográficos.

2.3.3 Distância

Conceito

A definição matemática de distância entre dois pontos é formalizada pelo conceito de *espaço métrico,* ou seja, um espaço onde há uma *distância métrica,* definida da seguinte forma:

Dados *(E,D)* onde *d* é uma distância métrica em um espaço métrico *E, d* é a função *d: E X E* $\rightarrow$ R^+ que associa dois elementos (*a,b)* de *E* a um valor real de tal forma que:

$d(a,b) \geq 0$

$d(a,b) = 0 \Leftrightarrow a = b$

$d(a,b) = d(b,a)$ *(simetria)*

$d(a,b) \leq d(a,b) + d(b,c)$ *(desigualdade triangular)*

Um espaço *pseudométrico* não obedece às condições acima.

O conceito de distância pode ser estendido a outras métricas que não apenas a medida da distância física entre dois pontos. Desta feita, conforme for o enfoque com o qual o geógrafo aborda o problema a ser estudado, diferentes medidas de distâncias relativas podem ser empregadas, como a distância tempo, a distância social, a distância custo, dentre outras. A escolha de determinada métrica tem por conseqüência a definição de uma geometria específica para representação do

espaço sob análise. Por exemplo, ao se empregar a distância tempo, as coordenadas geográficas transformam-se em coordenadas temporais. Outro exemplo de mudança da métrica espacial que importa no abandono da Geometria Euclidiana encontra-se na análise espacial de espaços urbanos reticulados, na qual a métrica táxi é a utilizada para medir distâncias (item 4.3).

A generalização da expressão da distância entre dois pontos em um espaço métrico é dada pela *distância de Minkowski,* definida da seguinte forma:

Dados dois pontos $P=(x_1, x_2,.. x_n)$ e $Q=(y_1, y_2,.. y_n)$ tem-se

$$d(P,Q) = \left(\sum_{i=1}^{n} \left| x_i - y_i \right|^{p} \right)^{1/p} \quad (1)$$

Alguns casos notáveis desta distância:

Distância Euclidiana (p=2):

$$d(P,Q) = \left(\sum_{i=1}^{n} \left| x_i - y_i \right|^{2} \right)^{1/2} \quad (2)$$

Distância Táxi (p=1):

$$\therefore d(P,Q) = \left(\sum_{i=1}^{n} \left| x_i - y_i \right| \right) \quad (3)$$

Distância de Chebyshev:

$$d(P,Q) = \lim_{p\to\infty}\left(\sum_{i=1}^{n}\left|x_i - y_i\right|^{p}\right)^{1/p} = \max_{i=0}^{n}\left|x_i - y_i\right| \quad (4)$$

Bola no Espaço Métrico

Uma bola no espaço métrico delimita um determinado espaço a partir de um ponto de origem e de um raio *r.*

Seja *E* espaço métrico, $a \in E, r > 0$.

Se seja uma distância *d.* Se *d* < *r*, tem-se uma *bola aberta.* Se $d \leq r$, tem-se uma bola fechada.

O conceito de bola é importante no contexto da Primeira Lei da Geografia, já que é a distância a partir da ocorrência de determinado fenômeno que determina a proximidade.

Tempo/espaço

(Abreu, 2005) observa que no contexto espacial a distância entre aquele que vivencia o espaço e o fenômeno espacial vivenciado interfere significativamente na intensidade da experiência. Já no contexto temporal, a intensidade tende a ser máxima no presente, quando o fenômeno encontra-se temporalmente próximo daquele que o vivencia. Uma maior distância entre o fenômeno e o sujeito da vivência espacial tende a permitir uma maior intensidade da experiência nesse contexto em particular. Tais particularidades devem ser levadas em conta na análise espacial quando as duas dimensões em tela são associadas com o propósito de estudar os fenômenos espaciais em sua dinâmica temporal (Abreu, 2005).

Distância/Tempo

Dentre os muitos geógrafos cuja obra demonstra preocupação com o problema da dimensão temporal nos fenômenos espaciais, destacam-se R. Abler, D. Harvey e D. Janelle. O modelo de convergência espaço-tempo de Janelle avalia os impactos espaciais das inovações e melhoriasnos meios de transporte. Ele tem por ponto de partida a constatação de que a demanda por melhor acessibilidade a determinados centros fornecedores e/ou consumidores tende a pressionar por melhorias nos meios de tranporte.Se essa demanda for suficientemente grande, é provável que sejam deflagrados esforços para se buscar formas de satisfazê-la. Como resultado desses esforços, inovações nos meios de transporte são alcançadas. Como conseqüência, centros de fornecimento e/ou de consumo tornam-se mais próximos, quando sua distância relativa é medida tem termos de distância de deslocamento, provocando a convergência tempo-espaço da qual decorrem alterações significativas na organização espacial das atividades econômicas, em um contexto de crescente interação. Dessa forma, o próprio incremento da interação espacial provocado pela redução relativa das distâncias tenderá a realimentar a demanda por melhorias na acessibilidade entre os centros em consideração, tornando recursivo o fenômeno indicado (Abreu, 2005).

2.3.4 Atividades da Análise Espacial

Um processamento de análise espacial pode ser estabelecido em atividades, distribuídas nas seguintes etapas:

Tratamento de dados

Os dados extraídos e devidamente georreferenciados podem necessitar de tratamentos por meio de *filtragem,* quando em excesso ou com uma grande quantidade de ruído, ou de *interpolação,* quando a quantidade de dados for insuficiente ou possuir muitas lacunas.

Definir camadas de mapeamento

Nesta etapa são definidas camadas de mapeamento nas quais os dados serão visualizados.

Visualização de dados em mapas

Nesta etapa é importante a visualização dos dados em mapas, de forma que sejam identificadas distribuições atípicas ou de padrões facilmente perceptíveis (como no caso do cólera em Londres, em que John Snow identificou uma alta concentração de mortes em torno da bomba de *Broad Street*). Assim, é possível estabelecer hipóteses iniciais sobre as observações, de forma a se selecionar o modelo de análise espacial mais adequado.

Aplicar métodos estatísticos quantitativos

Nesta etapa aplicam-se métodos de estatística descritiva e de análise multivariada sobre os dados (média, variância, desvio-padrão, análise de componentes principais, dentre outros).

Aplicar métodos de análise espacial

Nesta etapa métodos de análise espacial são aplicados, de forma a realizarem-se *inferências* sobre os dados. Tais inferências podem ser das seguintes naturezas:

- De variação contínua, onde determinam-se superfícies contínuas nas quais a probabilidade de ocorrência de um evento é maior.
- De variação discreta, em que determinam-se vários agrupamentos discretos de dados por área.
- Pontual, onde determinam-se locais específicos onde a probabilidade de ocorrência é maior.

3 A Proposta Metodológica para Visualização, Tratamento e Análise De Informações Espaciais em Ambiente Web

Neste capítulo é apresentada a proposta métodológica para desenvolvimento de aplicações destinadas à visualização, tratamento e análise de dados espaciais. Ela é constituída essencialmente de um *processo de desenvolvimento* e de uma *arquitetura de software*, suportados por um conjunto de *ferramentas computacionais de apoio.*

3.1 –Alguns Pressupostos Teóricos

Neste item serão apresentados alguns conceitos básicos importantes, pressupostos na definição da proposta metodológica aqui apresentada.

3.1.1 *Data Warehouses* Convencionais e Espaciais

Este item introduz a estrutura básica de armazenamento de dados utilizada na proposta arquitetural , o *data warehouse* e a sua versão espacial.

3.1.1.1 *Data warehouses* convencionais

3.1.1.1.1 Conceito

Dados não estruturados se caracterizam essencialmente por não estarem uniformemente armazenados em estruturas formais, como as propostas por modelos relacionais, hierárquicos ou orientados a objetos. Infelizmente, grande

parte das organizações têm suas informações armazenadas desta forma, que tradicionalmente dificulta a obtenção de relatórios para tomada de decisões ou visualização de informações relevantes.

Segundo (Inmon, 1992), um *data warehouse* pode ser definido como um modelo de uma coleção de dados integrados e variantes no tempo, orientado a um assunto específico. O *data warehouse* é um típico processo de transformação de dados em informações relevantes à tomada de decisões. Segundo (Barquin, 1996), é através dele que as organizações extraem valor de seus recursos informacionais.

A matéria prima para as atividades de inteligência organizacional são as informações (Cantelli, 2006). Portanto, é de crucial importância que estas sejam eficientemente manipuladas, o que se consegue com *transformação, manipulação* e *visualização* adequada dos dados. As técnicas *data warehousing* , bastante evidentes na atualidade, são completamente adequadas à estruturação de dados com o objetivo de proporcionar uma visão mais estratégica da informação.

A estrutura do *data warehouse* é tal que proporciona consultas complexas, com o uso de muitos registros de dados essencialmente estáticos, organizados em tópicos. Uma característica marcante do mesmo, é que deve permitir a separação e a combinação de dados de várias maneiras possíveis. Ele é carregado a partir de múltiplas fontes de dados, como bancos de dados operacionais, planilhas, arquivos textos, dentre outros. Além disso, o *data warehouse* é uma ferramenta de análise e apresentação de informações (Cantelli, 2006).

Segundo (Barquin, 1996) e (Kimball, 1996) as principais características de um *data warehouse* são as seguintes:

- Dados são orientados ao tempo e cobrem um longo período;
- Dados são orientados e organizados por assunto, somente os dados que interessem ao negócio são considerados.
- Dados produzidos no dia-a-dia normalmente são heterogêneos, distribuídos em múltiplos formatos, como planilhas eletrônicas, arquivos-textos, tabelas, dentre

outros. O *data warehouse* integra, consolida, e homogeneiza tais dados em uma estrutura sólida, facilitando o acesso aos mesmos.

- Dados tornam-se consistentes internamente, totalmente independentes do meio de armazenamento.
- Como conseqüência, *data warehouses* manipulam eficientemente um grande volume de dados.
- *Data warehouses* são não voláteis e devem estar sincronizados com os dados de produção.
- *Data warehouses* devem permitir o acesso a informação em vários níveis de agregação, através de sumarizações e detalhamentos.

Segundo (Cantelli, 2006) e (Barquin, 1996), dentre os principais benefícios de um *data warehouse* podemos citar a possibilidade de decisões mais inteligentes pois simplifica o acesso à informação, a obtenção mais rápida de diagnósticos e tendências e a não interferência nos sistemas em produção. Por outro lado, o desenvolvimento de sistemas de *data warehouse* é complexo, oneroso, exigindo profundo conhecimento do negócio em questão e treinamento dos usuários finais.

3.1.1.1.2 Estrutura de um *data warehouse*

A Figura 5 exibe a estrutura geral de um *data warehouse.*O banco principal do mesmo pode ser dividido nos chamados *data marts,* que têm um papel semelhante ao do *data warehouse,* mas são pertencentes a um contexto específico (departamentos ou setores, por exemplo) e são também implementados por meio de modelagem multidimensional, formando os denominados *modelos estrela* (ou, menos frequentemente, o modelo floco de neve, conforme veremos adiante).Um *data warehouse* possui também um armazém de dados operacionais, segundo (Kimball,1998), um repositório temporário e volátil de dados detalhados do *data warehouse.* As ferramentas de OLAP e de mineração de dados fornecem subsídios

à análise dos dados. Todas as estruturas e processos de um *data warehouse* têm sua integridade garantida por meio de *metadados,* que permitem um entendimento completo dos dados dentro do modelo.

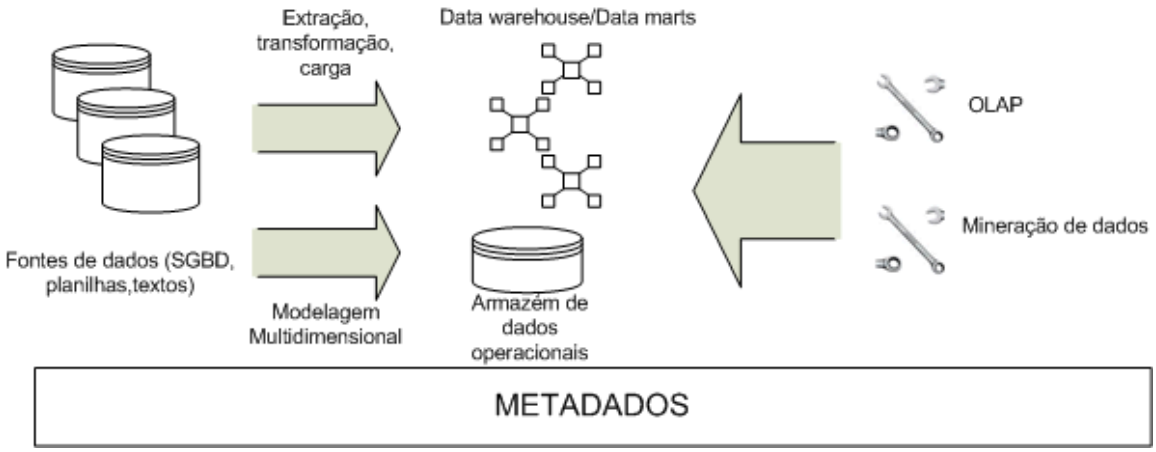


Figura 5 – Estrutura geral de um data warehouse (Adaptada de (Campos, 2000) e (Fidalgo, 2005))

3.1.1.1.3 O Processo de Um *Data Warehouse*

Essencialmente, o processo de um *data warehouse* consiste nas operações de extração, transformação e carga de dados (estas usualmente denominadas ETL) e na operação de análise de dados.

Extração de Dados

É a extração de dados de múltiplas fontes através de atividades de (Benson, 1997 e Braga, 2004):

Importação. Trata-se de um dos maiores desafios na implementação de *data warehouses.* As principais questões a serem levadas em consideração são as seguintes:

- Importação de formatos distintos de arquivos, que podem estar estruturados em bancos de dados, em planilhas, arquivos texto, ou até mesmo em formatos complexos como vídeo, áudio, imagens, etc.
- Importação de arquivos provenientes de tecnologias diferentes, como por exemplo, sistemas operacionais diferentes;
- Associação de arquivos estruturalmente diferentes, com granularidades ou temporalidades distintas;
- Dados podem ser estruturalmente complexos, sem qualquer grau de normalização.
- Deve-se manter informação de temporalidade dos dados armazenados, de forma a se evitarem duplicidades.
- Deve-se manter o mapeamento entre nomes de atributos nos arquivos de origem e de destino.

Limpeza: Trata-se da operação de uniformização da formas de referência à informação. Questões a se considerar:

- Abreviações (Av. e Avenida, por exemplo) , presença ou não de preposição (Av. Contorno e Av. do Contorno, por exemplo), sinônimos (gênero e sexo por exemplo), dentre outras.
- Uniformização de formatos de datas, horas, dias, dentre outros;
- Compatibilização de dados, como por exemplo, converter unidades de medida.
- Dependência entre dados devem ser mantidas e consistidas.
- Retirar caracteres espúrios ou desnecessários, como por exemplo, espaços.

Validação: Trata-se da verificação das duas últimas atividades, no que diz respeito à verificação da presença de todos os dados necessários, busca por mudanças

bruscas, verificação de uniformidade de versão , verificações de consistência e de uniformidade das agregações.

Transformação de dados

Consiste no conjunto de transformações a serem feitas sobre os dados, que permitirão a criação de um modelo único para o banco de dados do *data warehouse* a partir das diversas fontes de dados. (Braga,2004), (Devlin, 1997) e (Kimball, 1998) sugerem as seguintes transformações:

- Particionamento dos dados de acordo com um critério qualquer de seleção.
- Extração ou divisão dos atributos dos dados ou das tabelas dos modelos em diversas tabelas que serão utilizadas em diversos contextos.
- Concatenação de atributos pertencentes a várias tabelas sob um determinado contexto.
- "Desnormalização", que consiste em juntar informações de diversas tabelas em uma só , criando redundâncias no modelo relacional de forma a "desnormalizá-lo". Este processo, ainda que aumente o uso de espaço em disco, permite maior velocidade de recuperação da informação. O processo inverso, a normalização, pode ocorrer em *data warehouse,* mas é muito menos comum.
- Construção de consultas que resultarão na agregação de todo, ou parte, do banco de *data warehouse* em uma dimensão (tempo, ou também outra qualquer). Os dados devem poder ser desagrupados para serem vistos em resumo ou em detalhe. Estes processos são denominados *RollUp* e *DrillDown*, respectivamente. Denomina-se nível de granularização ao menor nível de detalhe que pode ser alcançado em uma consulta. Em agregação de dados, perde-se um nível de informação que só pode ser recuperado através dos dados originais, mas isto custa memória e desempenho. Quanto maior a granularização tanto menor o desempenho.

- Enriquecimento, que consiste em aumentar a complexidade de um atributo. Como exemplos, podemos citar a junção de atributos, a mudança de formato, dentre outros.
- Conversão de um tipo de dado de um atributo para outro.
- Remoção de atributos que não serão úteis
- Combinação de fontes de dados, pelo casamento exato de valores chaves ou pela busca em atributos não chaves, como a procura por equivalências textuais.
- Criação de chaves substitutas em dimensões para diminuir a dependência de chaves legadas, de forma a reforçar a integridade entre tabelas de dimensões e tabelas de fatos.
- Filtragem de dados irrelevantes ou, interpolações e extrapolações para aproximação de dados não presentes.

Carga de dados

Consiste no processo de efetivamente carregar os dados (tabelas de dimensões e de fatos) no *data warehouse.* Tais dados devem ser devidamente indexados de forma a aumentar o desempenho de consultas.

Análise de Dados

O processo de análise consiste em operações para extração, processamento e exibição de informações relevantes para tomadas de decisões, de forma que toda a complexidade torne-se intuitiva e transparente ao usuário. As técnicas de *OLAP-Online Anaytical Processing,* e de *mineração de dados* (que serão apresentadas em capítulo a parte) são intensivamente utilizadas nessa fase.

3.1.1.1.4 Modelagem multidimensional

A informação em um *data warehouse* usualmente encontra-se representada no denominado *modelo multidimensional,* que proporciona uma estrutura de dados não relacional para suportar os requisitos de informação necessária.

A Figura 6 exibe um exemplo da arquitetura multidimensional denominada *em estrela.* Tal estrutura é constituída pelas seguintes partes:

- *Tabela de fatos:* uma tabela central que agrega as informações a serem analisadas, constituída de *medidas* ou *fatos,* que são informações importantes a serem utilizadas na análise do negócio em questão, e das chaves primárias das tabelas de dimensões, descritas a seguir. No exemplo, tem-se a tabela de fatos constituída pelos fatos numcursos, numalunos e numprof, e pelas chaves estrangeiras idIES, idArea, idTit, idTipPosGrad, idMum e IdAno provenientes das tabelas de dimensões

- *Tabela de dimensões:* dimensões são informações em função das quais os fatos são analisados, proporcionando diferentes visões dos mesmos, podendo inclusive ser utilizadas como restrições em consultas. As tabelas de dimensões contêm uma chave primária, atributos descritivos e outros dados eventualmente relevantes. No exemplo, são as tabelas IES, Titulação, Município, Área, TipoPosGrad e Ano. A dimensão correspondente ao tempo, em especial, é base para quase todas as análises a serem realizadas, nos mais diversos níveis de detalhe (dia, semana, mês, ano).

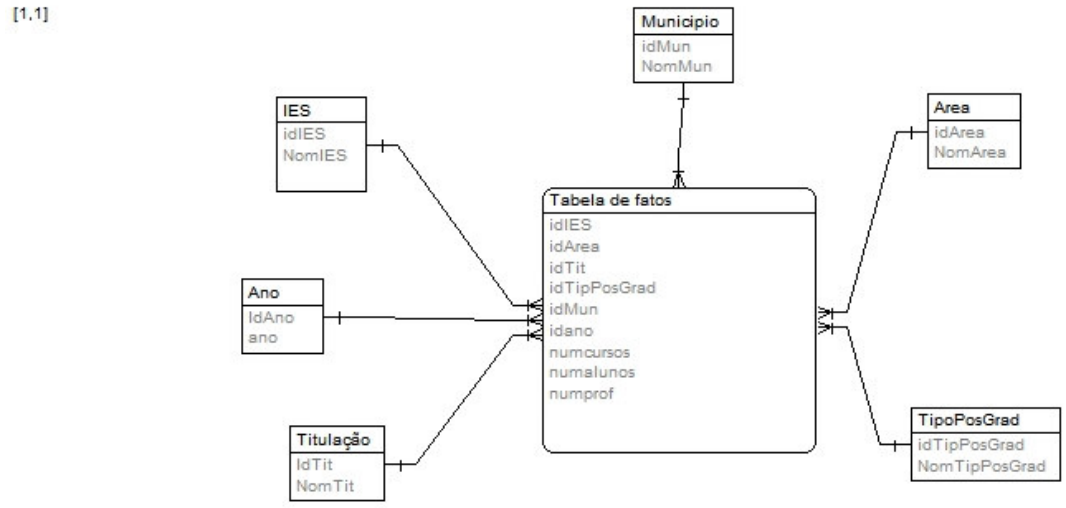


Figura 6- Exemplo de modelo multidimensional em estrela

Uma alternativa ao modelo estrela é o denominado *modelo snow flake,* em que determinadas tabelas de dimensões são desvinculadas da tabela de fatos, sendo vinculadas a outra dimensão, de forma a diminuir a redundância na tabela de fatos. A Figura 7 abaixo mostra o modelo *snow flake* correspondente ao modelo estrela acima exibido, em que a dimensão município não se relaciona mais com a tabela de fatos, tendo sido deslocada para se relacionar com a tabela IES. Modelos em *snow flake* permitem a hierarquização das dimensões. Como exemplo citam-se as seguintes hierarquias muito comuns: Continente→País→Estado→Município, Dia-→Mês-→Ano, Diretoria→Departamento→Gerência, dentre outras. (kimball, 1996) não recomenda o uso deste modelo, pois o mesmo economiza muito pouca memória (em torno de 1%), reduzindo sobremaneira o desempenho do processo de busca.

[1,1]

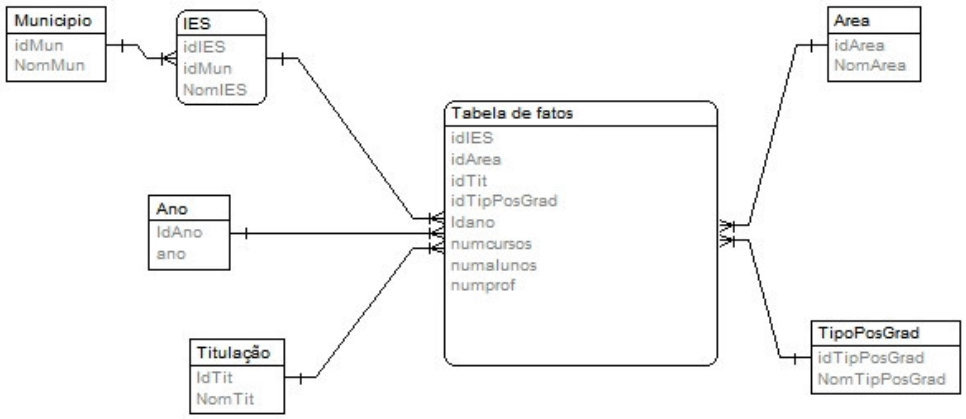


Figura 7- Exemplo de modelo multidimensional snow flake

3.1.1.1.5 *On-line Analytical Processing* – OLAP

Trata-se essencialmente do processo de análise de dados de um *data warehouse* sob diversas perspectivas. A estrutura básica do OLAP é o denominado *cubo multidimensional,* constituído por diversas dimensões em função das quais as medidas são exibidas. No modelo estrela da Figura 6 acima, podemos ter, por exemplo, os seguintes cubos:

- Número de cursos criados (medida *numcursos)* em determinadas áreas, por ano e por tipo de pós-graduação .
- Número de alunos (medida *numalunos),* por titulação, por ano *,* em cada IES (Instituição de ensino superior)

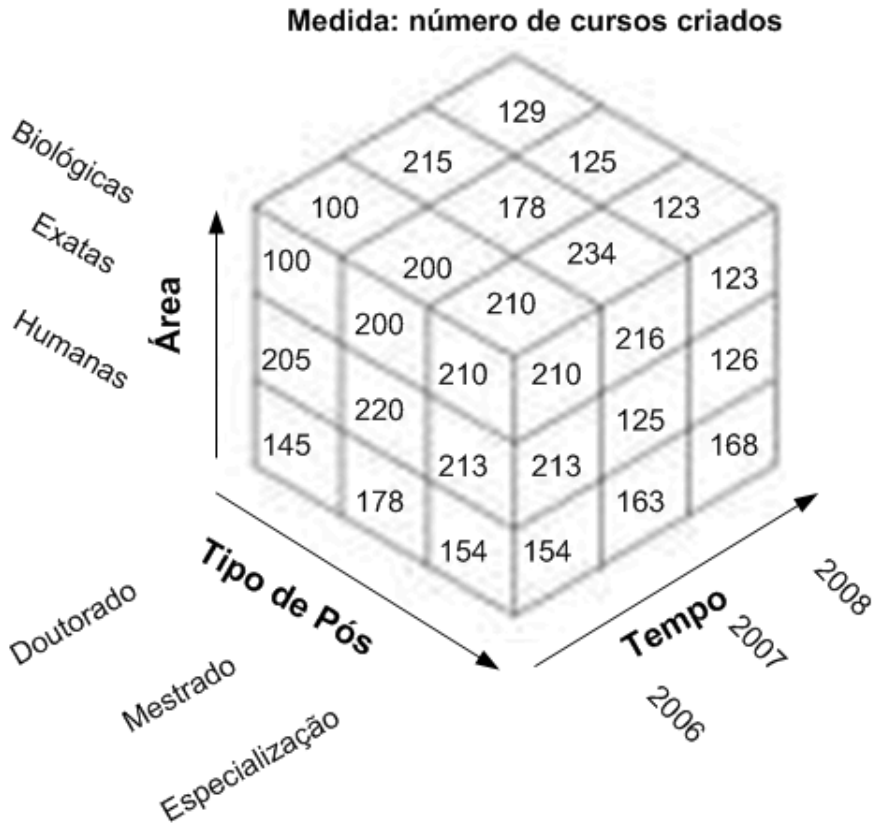


Figura 8-Exemplo de cubo multidimensional

Segundo (Stefanovic, 1997), um cubo consiste de um certo número de *views* (visões), também denominadas *cubóides.* Na Figura 8, cada pequeno cubo contendo um número de alunos é um cubóide.

(Harinarayan, 1996) sugere que as dimensões de um modelo dimensional sejam congregadas em todas as suas combinações possíveis (2^n) de forma a produzir cubóides compostos de uma combinação entre essas dimensões. No exemplo da Figura 8, a medida *número de cursos criados* pode ser analisada em termos de 8 combinações das dimensões (Área=*a*, Município=*m* e Tempo = *t*) . A Figura 9 exibe um grafo dessas combinações, onde cada vértice corresponde a um cubóide (visão) que congrega as dimensões indicadas no mesmo e as arestas exibem as dependências entre as visões. Por exemplo, a visão *am* pode ser obtida a partir da

amt e a visão *a* pode ser obtida a partir de *am* e *at.* Qualquer visão pode ser obtida a partir de *amt* e *nenhuma* é obtida a partir de qualquer outra.

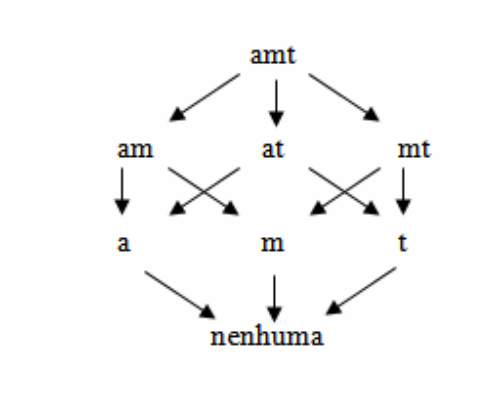


Figura 9-Níveis de agregação em um cubo de 3 dimensões

Em OLAP, definem-se *níveis* para as dimensões, que são na verdade níveis de detalhamento. Por exemplo, uma dimensão *localização* pode possuir os níveis endereço, cidade, estado, região e país. Combinações entre níveis, produzem diferentes *hierarquias*, que definem uma seqüência de agregação entre os mesmos.. Os níveis citados podem gerar, por exemplo as hierarquias endereço→cidade→estado→região→país, endereço→cidade→estado→país, cidade→estado→país, dentre outras.

As seguintes operações podem ser realizadas sobre um cubo:

Fatiar (slice): corresponde à projeção de algumas arestas do cubo. Por exemplo no cubo da Figura 8, se fixarmos o ano de 2006, criamos uma "fatia" contendo o número de cursos criados por tipo de pós-graduação e por área (Figura 10);

Rotacionar ("dice"): consiste em visualizar-se os dados sob outra perspectiva, aplicando uma rotação ao cubo de forma a obter-se uma nova projeção. Por exemplo, podemos querer visualizar agora o cubo sob a perspectiva do número de

cursos da área de biológicas por ano e por tipo de pós-graduação após rotação de 90º em torno de um eixo paralelo ao de tipo de pós-gaduação (ver Figura 11).

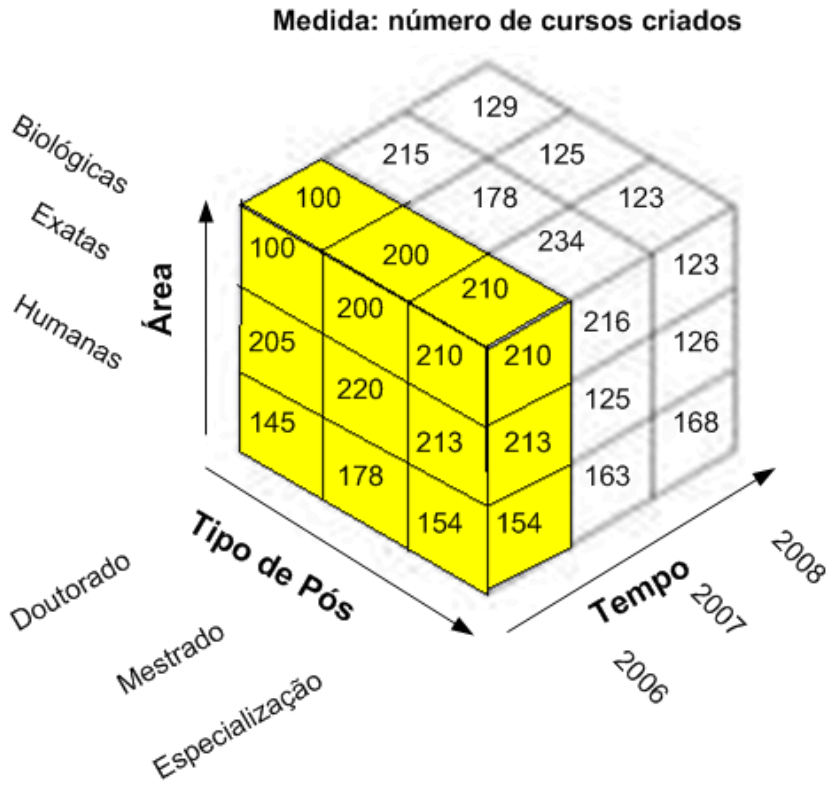


Figura 10 -Exemplo de cubo multidimensional com uma "fatia" destacada para a dimensão tempo=2006

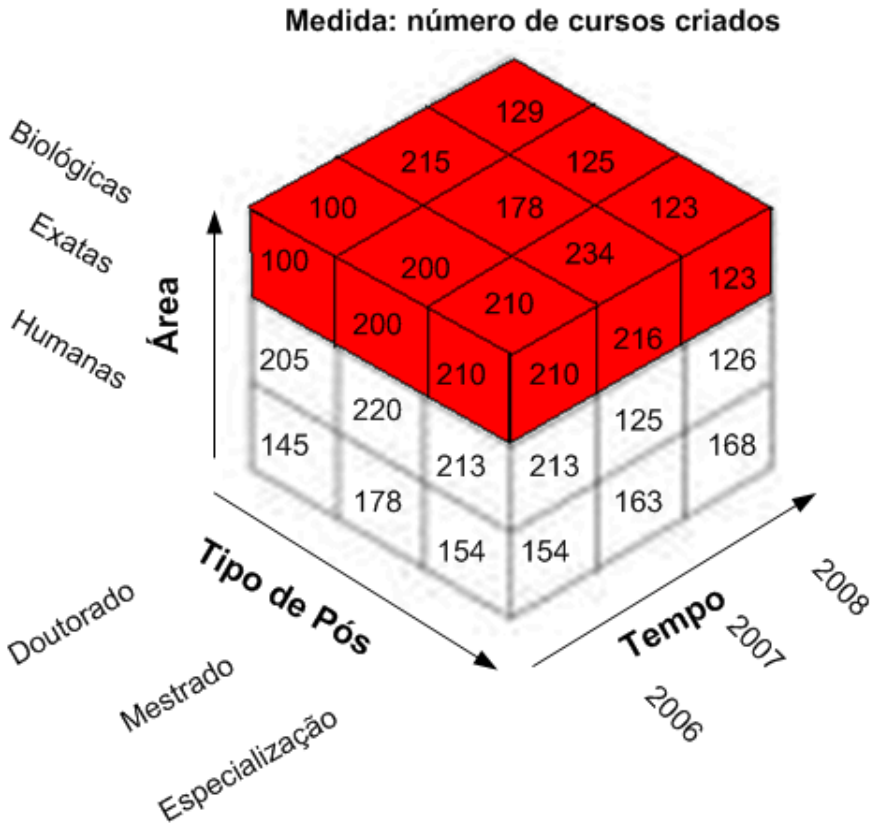


Figura 11-Exemplo de cubo multidimensional com uma "fatia" destacada para a dimensão Área=Biológicas

Roll Up

É o processo de agregar valores em uma hierarquia, obtendo-se um maior grau de generalidade. Por exemplo, na hierarquia endereço→cidade→estado→região→país, o *roll up* é feito no sentido endereço→país.

Drill Down

É o processo de detalhar valores em uma hierarquia, obtendo-se um maior grau de especificidade. Por exemplo, na hierarquia endereço→cidade→estado→região→país, o *drill down* é feito no sentido país→endereço.

3.1.1.1.6 Metadados

Os denominados metadados são um *metamodelo* (um "modelo do modelo") de todos os dados e sub-processos que ocorrem no processo de um *data warehouse* (Barquin,1996), utilizados para que se entendam com maior facilidade os papéis, processos e significados dos dados dentro do modelo de negócios. Segundo (OMG, 2003) metadados em um *data warehouse* devem se focar principalmente nos seguintes aspectos:

Gerência: Contém informações sobre gerência do processo de criação e de operação de um *data warehouse;*

Análise: São informações acerca de OLAP, mineração e visualização de dados;

Construção: Contém informações relativas a extração e transformação dos dados.

3.1.1.2 *Data warehouse* espacial

3.1.1.2.1 Conceito

Um *data warehouse espacial* agrega a dimensão *espaço* a um sistema de *data warehouse* convencional, ou seja, a localização geográfica torna-se um importante e útil instrumento à análise das informações, no que tange à distribuição da mesma no espaço. A Figura 12 exibe a estrutura de um *data warehouse* espacial. Em relação aos *data warehouses* convencionais, incorpora particularidades na etapa de extração, onde utilizam-se técnicas de geocodificação da informação a ser extraída, e na etapa de análise, onde técnicas de análise espacial são exploradas em conjunto com as convencionais. Além disso, o banco de dados deve incorporar a capacidade de armazenamento e manipulação de estruturas espaciais, dentre os quais podemos citar o Oracle Spatial, PostGis e MySql com extensões espaciais. As etapas do processo de um *data warehouse* espacial são essencialmente as

mesmas de um *data warehouse* convencional, com particularidades introduzidas pela presença da dimensão espacial, as quais serão destacadas a seguir.

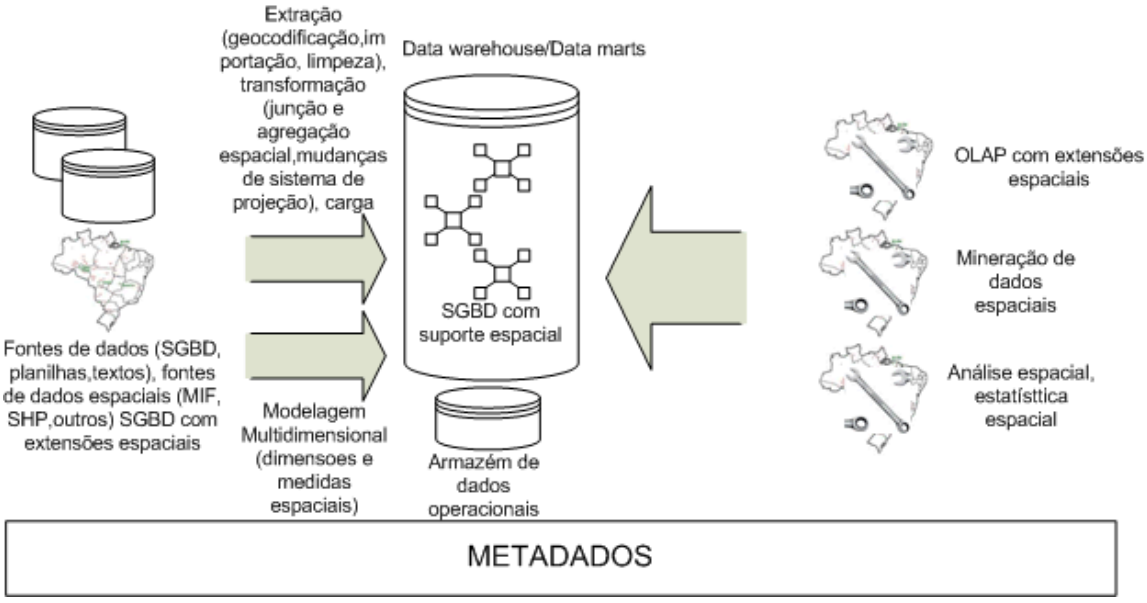


Figura 12 – Estrutura de um data warehouse espacial

3.1.1.2.2 Extração de informações espaciais

A etapa de extração de informações espaciais requer as operações de geocodificação (descritas no item 2.2.6) dos dados, normalmente realizada por GIS e/ou de junção de dados convencionais a objetos geográficos presentes em mapas armazenados em GIS. Nesse caso, os dados convencionais e os objetos geográficos devem possuir uma chave comum que permita associá-los em um banco de dados comum (por exemplo, associar uma base de dados sócio-econômicos de países a um mapa mundi, pelo código do país).

3.1.1.2.3 Transformação de Informações Espaciais

A etapa de transformação de informações espaciais envolve usualmente operações de:

- Junção espacial. Combina duas geometrias espaciais de acordo com predicados topológicos (vizinhança, por exemplo), métricos (distância, por exemplo) ou de ordem (ao lado de, por exemplo) (Fornary,2006). Por exemplo, realizar a junção espacial entre a feição geográfica que representa um município e as feições que representam rodovias segundo o predicado *intercepta(rodovia,município).* (Orenstein, 1996) sugere que a junção espacial seja feita em duas etapas: a primeira, denominada *filtragem,* testa os predicados para aproximações de objetos, como o envelope MBR (*Minimum Bounding Rectangle),*obtendo-se pares de objetos candidatos; a segunda etapa*,* de refinamento, verifica o predicado espacial para a geometria completa de cada objeto dos pares selecionados na etapa anterior. Essa abordagem pode gerar falsos candidatos, por isso alguns autores sugerem refinamentos, como (Brinkhoff, 1994) e (Kriegel,2003), que usam polígonos de 5 lados para identificar objetos que não se interceptam e (Azevedo, 2005), que usa aproximações *raster.*

- Agregação espacial. Consiste na agregação de geometrias espaciais em uma só geometria espacial resultante. Sistemas de bancos de dados com extensões espaciais realizam esta operação corriqueiramente.

- Mudança do sistema de projeção. É usual a necessidade de conversão entre sistemas de projeção, principalmente entre os mais comuns, latitude-longitude e UTM (e vice-versa). GIS (MapInfo e ARCGIS, por exemplo) realizam essa operação correntemente, portanto, muitas vezes ela é efetuada durante o processo de extração.

3.1.1.2.4 Modelagem Multidimensional Espacial

Dimensões espaciais

Dimensões espaciais estão relacionadas a objetos espaciais. (Stefanovic, 1997) identifica dois tipos de dimensões espaciais:

- Dimensão espacial-para-não-espacial: em um nível primitivo de granularidade, associa-se a um objeto espacial, mas em níveis mais altos torna-se não espacial. Por exemplo, uma dimensão localidade pode ser representada em seu nível mais baixo por um município, identificado por um objeto polígono, e em níveis mais altos, o de estado ou país, pelos nomes correspondentes.

- Dimensão espacial-para-espacial: neste caso, a dimensão é associada a um objeto espacial em todos os níveis, no exemplo acima, por objetos polígono nos níveis de município, estado e país.

(Fidalgo, 2005), por sua vez, propõe a seguinte classificação de dimensões espaciais:

- Dimensão espacial primitiva: contém apenas um campo espacial e não possui chaves estrangeiras provenientes de outras dimensões primitivas. Não contém chaves estrangeiras para tabelas de dimensões convencionais. Na Figura 13 a dimensão município possui apenas o campo espacial MUNOBJ, que é um polígono.

- Dimensão espacial composta: possui chaves estrangeiras de outras dimensões espaciais primitivas ou de outras dimensões espaciais compostas. Na Figura 14 a dimensão Localização é composta, pois possui chaves estrangeiras PAICOD, ESTCOD, MUNCOD e LOGCOD para as dimensões espaciais primitivas País, Estado, Município e Logradouro, respectivamente.

- Dimensão híbrida: possui chaves estrangeiras de outras dimensões espaciais primitivas, de outras dimensões espaciais compostas e de dimensões convencionais. Pode ser classificada em *micro ,* quando faz referência a apenas uma dimensão primitiva (Figura 15), *macro,* quando referenciar dimensões

primitivas ou compostas, e *conjunta,* quando for uma união dos dois tipos anteriores(Figura 16). As dimensões hibridas macro podem ser de *única junção,* quando se relacionarem por meio de uma única chave estrangeira a uma dimensão espacial composta (Figura 17), ou de *múltiplas junções,* quando se relacionarem com várias dimensões geográficas primitivas (Figura 18).

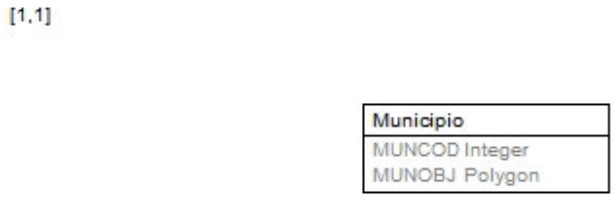


Figura 13 – Dimensão espacial primitiva

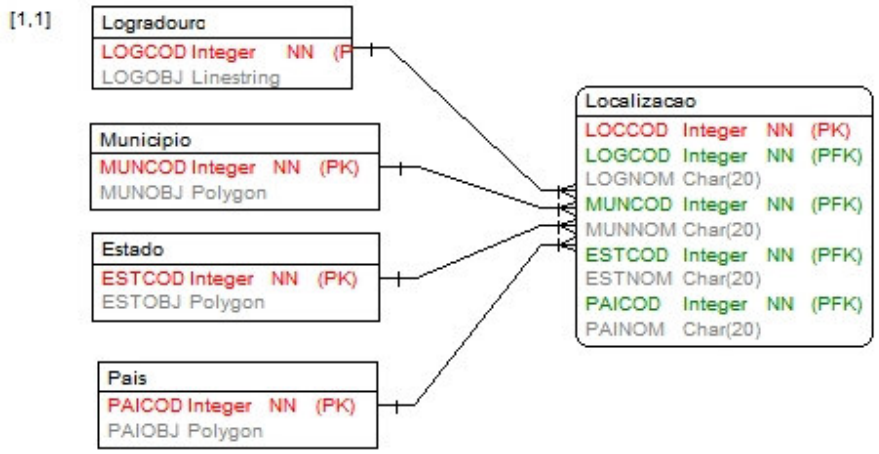


Figura 14- Dimensão espacial composta

[1,1]

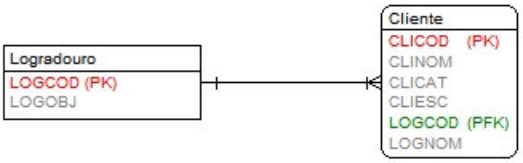


Figura 15-Dimensão espacial híbrida micro

[1,1]

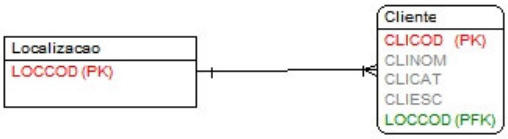


Figura 16-Dimensão espacial híbrida macro com uma junção

[1,1]

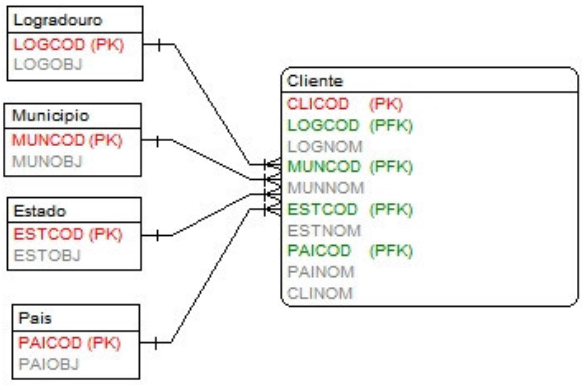


Figura 17-Dimensão espacial híbrida macro com múltiplas junções

[1,1]

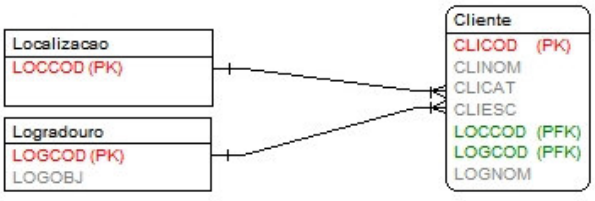


Figura 18-Dimensão espacial híbrida conjunta

Medidas espaciais

São medidas que contêm um conjunto de ponteiros para objetos espaciais (Stefanovic, 1997). Por exemplo, em um modelo meteorológico simples, podemos ter como dimensões *precipitação, temperatura e data* e como medida, em uma tabela de fatos meteorológicos, um conjunto de ponteiros para regiões que compartilhem as mesmas precipitações, temperaturas e datas. Em uma tabela de fatos, a medida espacial é representada por um campo que contém, por exemplo, um vetor de objetos espaciais ou um conjunto destes (campo *multipolygon* na Figura 19, por exemplo).

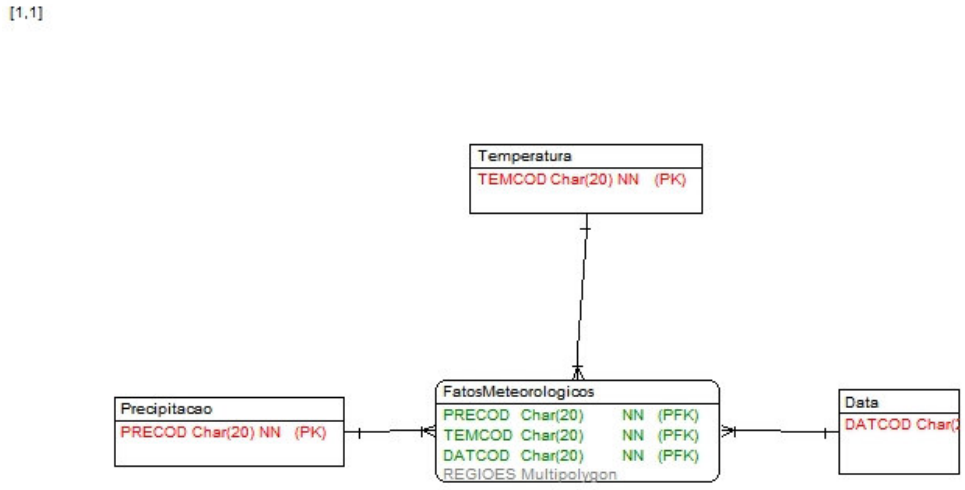


Figura 19-Exemplo de tabela de fatos com medida espacial

Granularidade espacial

Segundo (Fidalgo, 2005), de maneira a permitir consultas sobre áreas *ad-hoc,* é desejável que a menor granularidade espacial seja representada por objetos do tipo ponto, pois estes correspondem à menor localização espacial de fatos e permitam

que estes sejam consultados a partir de uma área *ad-hoc* e facilmente agregados em feições mais complexas, como polígonos. Isso ocorre porque determinadas operações espaciais, como as relacionadas a distância, por exemplo, são mais precisas nesse nível de granularidade. Se não se considerar este nível de granularidade, distâncias, por exemplo, entre polígonos, devem ser estimadas (Fidalgo, 2005).

3.1.1.2.5 OLAP Espacial

Operações de OLAP em nível espacial trazem as particularidade inerentes à presença de informações espaciais. Tais particularidades serão ressaltadas a seguir.

- ***Cubo multidimensional espacial-*** OLAP espacial utiliza cubos multidimensionais espaciais, que contêm dimensões convencionais e espaciais e podem ou não conter medidas espaciais.

- ***Níveis espaciais-*** São níveis nos quais as dimensões espaciais se agregam. Por exemplo, endereço, cidade, estado e país são níveis espaciais, ligados a dimensões espaciais.

- ***Hierarquias espaciais.*** – Analogamente à definição convencional, hierarquias espaciais definem uma seqüência de agregação entre níveis espaciais. Os níveis citados podem gerar, por exemplo a hierarquia endereço→cidade→estado→país.

- ***Agregações espaciais –*** Sistemas de OLAP espacial devem permitir as agregações de dados segundo a seqüência definida por uma hierarquia de níveis espaciais, nos dois sentidos e em quaisquer combinações de dimensões espaciais, por meio de operações de *roll-up* e *drill-down* espaciais. Por exemplo, na hierarquia espacial endereço→cidade→estado→país, pode-se totalizar um fato por meio dos seguintes agrupamentos no sentido de *roll-up* (inspirado em Fidalgo, 2005):

- Exibir país, estado, cidade, endereço e somatório do fato, agrupados por (país, estado, cidade, endereço)
- Exibir país, estado, cidade e somatório do fato, agrupados por (país, estado, cidade)
- Exibir país, estado e somatório do fato, agrupados por (país, estado)
- Exibir país e somatório do fato, agrupados por (país)
- Exibir somatório total dos fatos.

3.1.2 *O Open Geospatial Consortium*-OGC

3.1.2.1 Introdução

O *Open Geospatial Consortium* (OGC) é um consórcio internacional composto por 369 empresas, agências governamentais e universidades, criado com os seguintes objetivos formais (OGC, 2008):

- Prover padrões livres e abertos para informações espaciais;
- Liderar a criação de padrões que permitam a integração entre conteúdo e serviços espaciais a negócios e a empresas;
- Facilitar a adoção de arquiteturas espacialmente referenciadas em ambientes empresariais;
- Auxiliar na formação de novos mercados e aplicações para tecnologias geoespaciais.

3.1.2.2 O modelo de classes de objetos geográficos do *Open Geospatial Consortium (OGC)*

O modelo de classes de objetos geográficos do OGC é constituído por uma classe abstrata *feature* que se especializa essencialmente nas classes concretas *feature with geometry* (correspondente aos geo-objetos) e *coverage* (correspondente aos geo-campos). A Figura 20 exibe esta hierarquia. As Figuras 21 e 22 exibem respectivamente as especializações das classes *feature with geometry* e *coverage.*

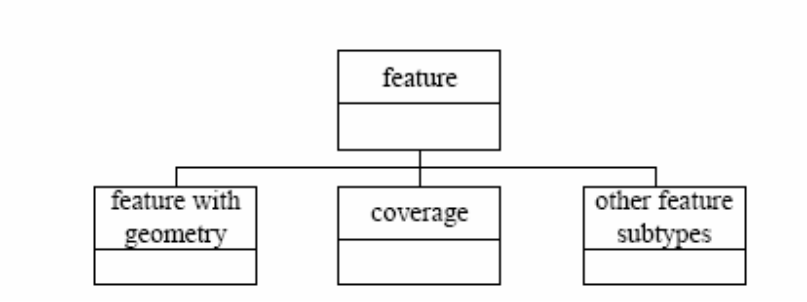


Figura 20- Estrutura de classes do OGC. Extraído de:(OGC, 2006a)

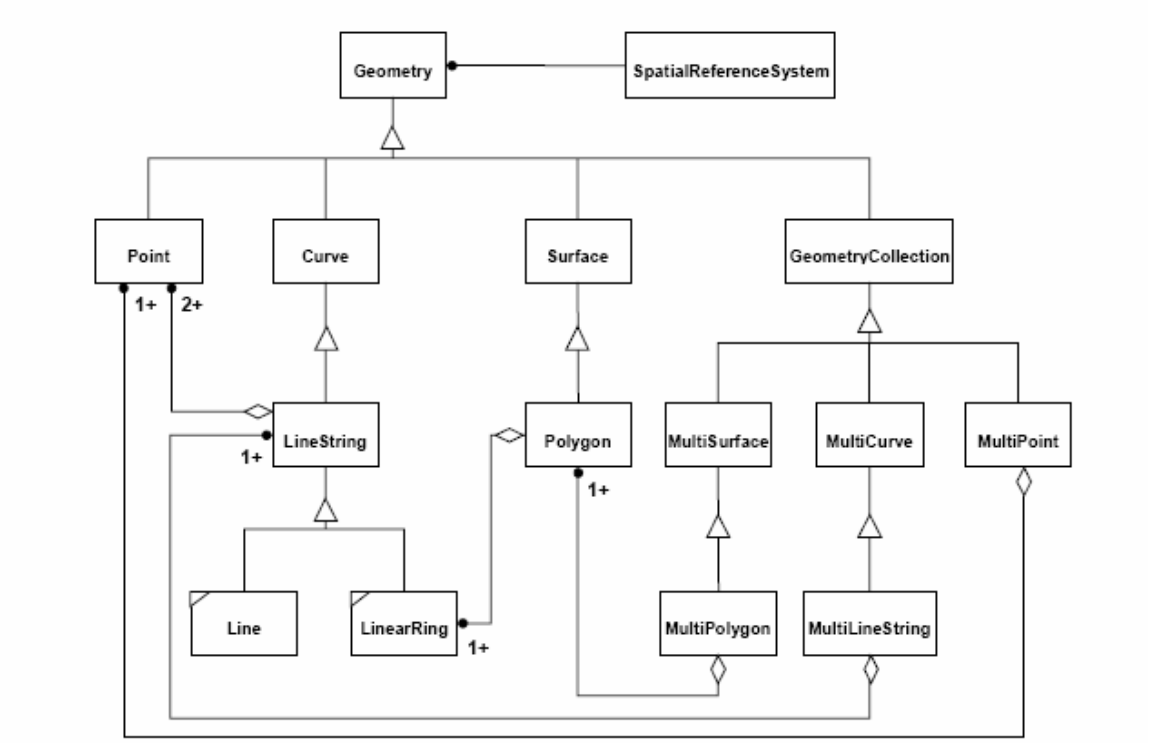


Figura 21-Especializações de feature with geometry. Extraído de:(OGC, 2003)

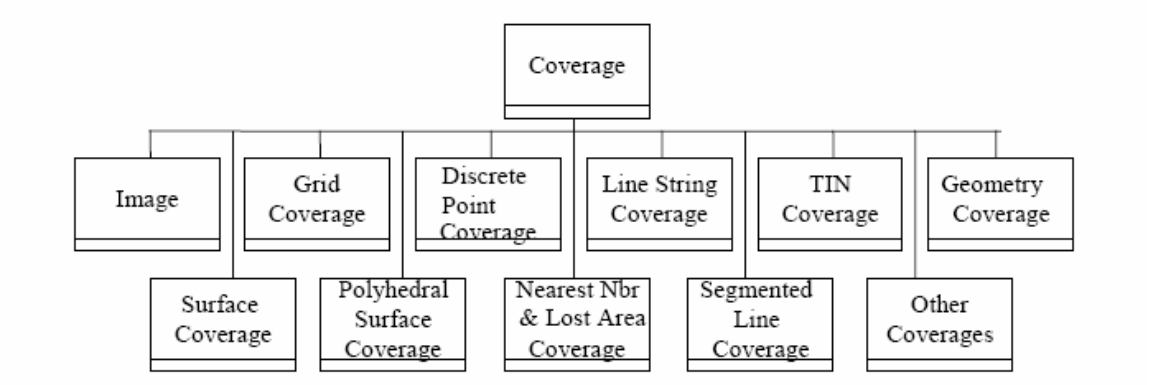


Figura 22-Especializações de coverage. Extraído de (OGC, 2006a)

3.1.2.3 Serviços Geoespaciais

Neste item serão apresentadas alguns dos serviços geoespaciais mais comuns, propostos pelo OGC. Os mais populares e de maior relevância a este trabalho serão apresentados em maior detalhe.

3.1.2.3.1 Arcabouço Arquitetural dos Serviços

Os serviços do OGC estão enquadrados no arcabouço arquitetural denominado *OpenGIS Service Framework –OSF* (OGC, 2003). Este arcabouço identifica serviços, interfaces e protocolos de troca a serem utilizados pelas aplicações (Figura 23). Compreende as seguintes categorias de serviços:

Serviços de Aplicação

Os outros serviços do OSF são acessados via Serviços de Aplicação operando em quaisquer tipos de terminais (*notebooks, desltops, handhelds,* etc). Normalmente,

provêem *displays* de conteúdo geoespacial orientados ao usuário e suportam a interação com o usuário nos terminais. Alguns exemplos de serviços incluem os seguintes:

- *Discovery Application Services*
- *Map Viewer Application Services*
- *Value-Add Application Services*
- *Imagery Exploitation Application Services*
- *Sensor Web Application Services*
- *Location Organizer (LO) Application Services*
- *Mobile Location Services*

Serviços de Registro

Provêem um mecanismo comum para classificar, registrar, descrever, buscar, manter e acessar informações sobre recursos disponíveis em uma rede. Tipos de registros são diferenciados por seus papéis: registros para catalogar tipos de dados, instâncias de dados *online,* tipos de serviços e instâncias de serviços *online* são alguns exemplos (OGC, 2003).

Serviços de Processamento

Serviços de processamento do OSF operam sobre dados geoespaciais e podem transformar, combinar ou criar dados. Podem ser acoplados a outros serviços. Podem ser seqüenciados em uma cadeia de serviços para realizar processamento especializado no suporte ao fluxo de produção de informação e ao suporte à tomada de decisões. Alguns exemplos incluem:

- *Chaining Services*
- *Coordinate Transformation Services (CTS)*
- *Geocoder Services*
- *Gazetteer Services*
- *Geoparser Services*
- *Reverse Geocoder Services*
- *Route Determination Services*

Serviços de Visualização (Portrayal Services)

Provêem visualização da informação geoespacial, ou seja, dadas uma ou mais entradas, provêem saídas renderizadas. Podem ser acoplados com outros serviços, como os de dados e processamento. Alguns exemplos incluem:

- *Web Map Services (WMS)*
- *Coverage Portrayal Services (CPS)*
- *Mobile Presentation Services*
- *Style Management Service (SMS)*

Serviços de Dados

Provê acesso a coleções de dados em repositórios e bases de dados. Normalmente esses serviços mantêm índices para auxiliar o processo de busca por nomes ou outros atributos do item. Alguns dos serviços:

- *Web Object Service (WOS)*
- *Web Feature Service (WFS)*
- *Sensor Collecting Service (SCS)*
- *Image Archive Service (IAS)*
- *Web Coverage Service (WCS)*

Codificações

São especificações para troca de informações geográficas, tais como topologias, geometrias e sistemas de referência. Dentre as várias disponíveis, destaca-se a versão geográfica do XML, a denominada *GML-Geographic Markup Language,* apresentada adiante, utilizada para representar estruturas de feições geográficas quando descrições das mesmas são demandadas como saídas de operações.

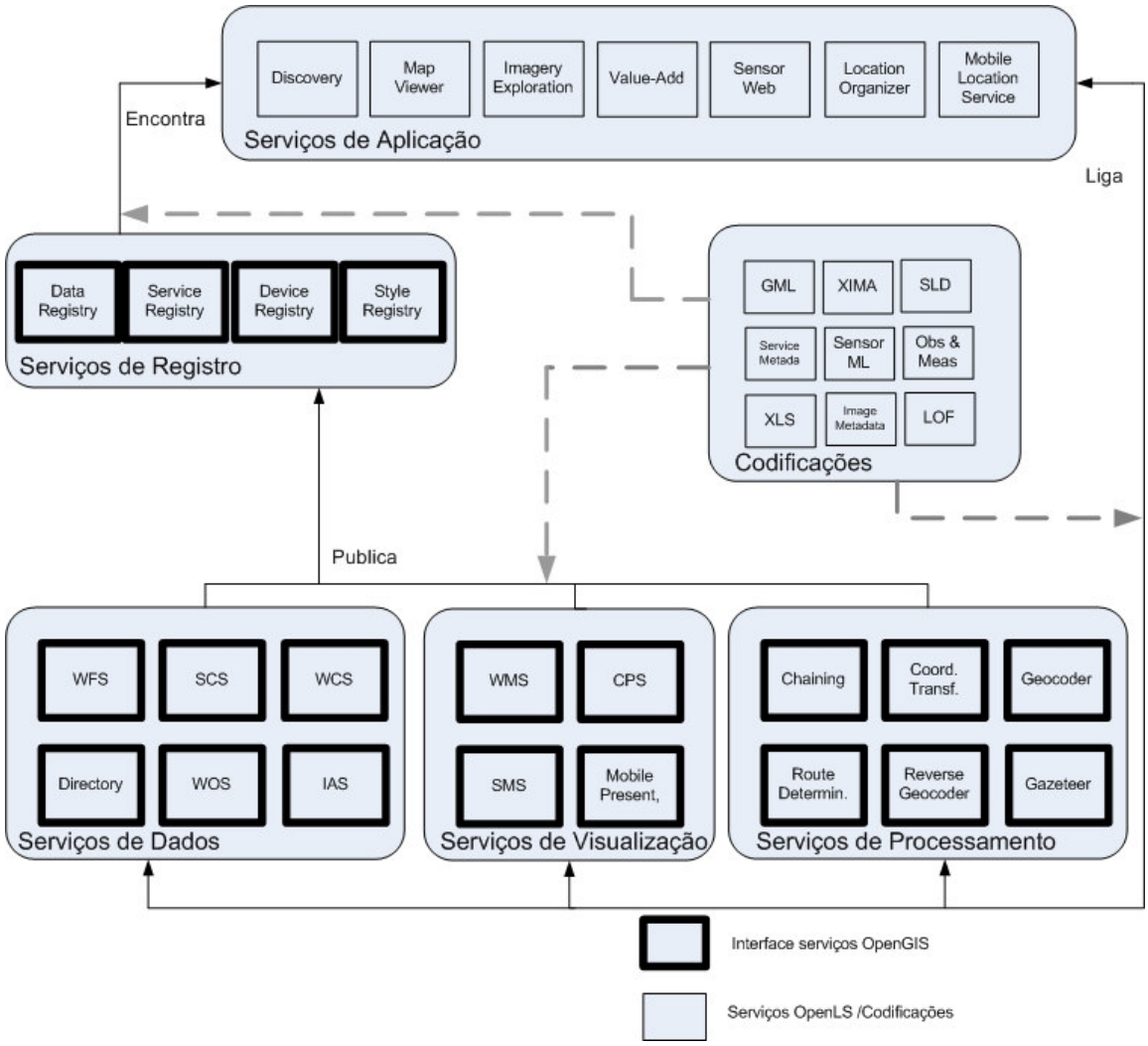


Figura 23- Componentes do OSF –Adaptado de (OGC, 2003)

3.1.2.3.2 *Web Map Service*

O denominado *Web Map Service (WMS),* tem como principal funcionalidade a produção de mapas de dados referenciados espacialmente a partir de informações geográficas. Em essência, são definidas três operações para este serviço: o retorno de informações de metadados; o retorno de um mapa gerado a partir de parâmetros geográficos e dimensionais bem definidos; o retorno de informações sobre feições específicas exibidas em um mapa (operação esta, opcional). Operações WMS

podem ser invocadas via *web browser* por meio de URLs, cujo conteúdo depende da operação requisitada (OGC, 2006b).

Operações do WMS

GetCapabilities

Segundo (OGC, 2004), tem como objetivo a obtenção de metadados do serviço, ou seja, uma descrição do conteúdo das informações do servidor e valores dos parâmetros de requisição. A URL de requisição deste serviço deve possuir os seguintes parâmetros:

Parâmetro	**Obrigatório(M)/Opcional(O**	**Descrição**
VERSION=version	O	Número de versão
SERVICE=WMS	M	Tipo de serviço
REQUEST=GetCapabilities	M	Nome da requisição
FORMAT=MIME_type	O	Formato de Saída
UPDATESEQUENCE=string	O	Núm. de seqüência

Tabela 1-Parâmetros requisição do GetCapabilities (OGC, 2006b)

A resposta desta operação deve ser um documento XML contendo os metadados do servidor: nomes e títulos, serviços gerais, capacidades, camadas e estilos.

GetMap

Retorna um mapa. A URL de requisição deste serviço deve possuir os seguintes parâmetros:

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=1.3.0	M	Versão da requisição
REQUEST=GetMap	M	Nome da requisição
LAYERS=layer_list	M	Lista de camadas separadas vírgula
STYLES=style_list	M	Lista de estilos separados vírgula, um para cada camad
CRS=namespace:identifier	M	Sistema de coordenadas referência
BBOX=minx,miny,maxx,maxy	M	Cantos da caixa de contorno mapa (inferior esquer superior direito)
WIDTH=output_width	M	Largura do mapa em pixels mapa
HEIGHT=output_height	M	Altura do mapa em pixels mapa
FORMAT=output_format	M	Formato de saída do mapa
TRANSPARENT=TRUE|FALSE	O	Transparência de fundo mapa
BGCOLOR=color_value	O	Cor de fundo do mapa hexadecimal (RGB)
EXCEPTIONS=exception_format	O	Formato em que exceçõ devem ser reportadas ao W (o default é XML)
TIME=time	O	Valor de tempo da cam desejada
ELEVATION=elevation	O	Elevação da camada desejad
Other sample dimension(s)	O	Valor de outras dimensões, for apropriado

Tabela 2 -Parâmetros requisição do GetMap (OGC, 2006b)

GetFeatureInfo

Trata-se de uma operação opcional, suportada apenas pelas camadas denominadas *"queryable",* para proporcionar aos clientes de um serviço WMS informações adicionais sobre feições nos mapas retornados pelas requisições.

Parâmetro	**Obrigatório(M)/Opcional(O)**	**Descrição**
VERSION=1.3.0	M	Versão da requisição
REQUEST=GetFeatureInfo	M	Nome da requisição
Parte a requisição de Mapa	M	Cópia parcial dos parâmetros requisição de mapas q geraram o mapa para o q informação é desejada.
QUERY_LAYERS=layer_list	M	Lista de camadas a ser consultadas, separadas vírgula
INFO_FORMAT=output_format	M	Formato de retorno informação acerca da feição.
FEATURE_COUNT=number	O	Número de feições sobre quais se deseja informação
I=pixel_column	M	Coordenada i, em pixels, feição no mapa
j=pixel_column	M	Coordenada j. em pixels, feição no mapa
EXCEPTIONS=exception_format	O	Formato em que exceçõ devem ser reportadas ao WI (o default é XML)

Tabela 3-Parâmetros requisição do GetFeatureInfo (OGC, 2006b)

3.1.2.3.3 *Web Feature Service-WFS*

Tem como objetivo a recuperação e a atualização de dados geoespaciais codificados em linguagem GML. Define interfaces para acesso a dados e operações de manipulação em feições geográficas. A partir de tais interfaces, pode-se combinar, usar e gerenciar dados georreferenciados provenientes de diversas fontes. Um WFS transacional provê a operação opcional *Transaction* para inserir, atualizar ou excluir uma feição (OGC, 2005).

Operações do WFS

GetCapabilities

Retorna uma lista de dados do servidor, bem como operadores e parâmetros do WFS.

Parâmetro	**Obrigatório(M)/Opcional(O**	**Descrição**
VERSION=version	O	Número de versão
SERVICE=WFS	M	Tipo de serviço
REQUEST=GetCapabilities	M	Nome da requisição
NAMESPACE = namespace	O	Especifica um namespace seus prefixos

Tabela 4-Parâmetros requisição do GetCapabilities (OGC, 2005)

DescribeFeatureType

Descreve a estrutura de uma feição geográfica.

Parâmetro	**Obrigatório(M)/Opcional(O)**	**Descrição**
VERSION=version	O	Número de versão
SERVICE=WFS	M	Tipo de serviço
REQUEST=DescribeFeatureType	M	Nome da requisição
NAMESPACE = namespace	O	Especifica um namespace seus prefixos
TYPENAME=namespace:featuretyp	O	Uma lista de elemen separados por vírgula or Namespace é o domínio featuretype é o nome do t de feição
OUTPUTFORMAT=myme_type	O	Formato de saída

Tabela 5-Parâmetros requisição do DescribeFeatureType (OGC, 2005)

GetFeature e GetFeatureWithLock

Retorna instâncias das feições espaciais contidas na base de dados (GetFeature) ou bloqueia uma ou mais instâncias que foram selecionadas. Pode haver seleções de feições.

Parâmetro	**Obrigatório(M)/Opcional(O)**	**Descrição**
VERSION=version	O	Número de versão
SERVICE=WFS	M	Tipo de serviço
REQUEST=DescribeFeatureType	M	Nome da requisição
NAMESPACE = namespace	O	Especifica um namespace

		seus prefixos
TYPENAME=namespace:featuretyp	M	Lista de tipos de feições selecionar, onde Namespac o domínio e featuretype é nome do tipo de feição
FEATUREID=feature	O	Filtro para de determina feição
MAXFEATURES= N	O	Número máximo de feições serem retornadas
SORTBY=property	O	Determina uma lista atributos (property) para quais será feita u ordenação no form "PropertyName [A|D][,PropertyName [A|D],..
PROPERTYNAME=property	O	Lista de propriedades a ser selecionadas, separadas vírgula
FEATUREVERSION=[all,N]	O	Versões das feições
FILTER	O	Filtros a serem aplicados feições, separados conectores lógicos *and.* De haver um filtro para ca TYPENAME
BBOX=minx,miny,maxx,maxy	M	Se FEATUREID ou FILT forem usados, pode especificar uma caixa contorno.
RESULTTYPE= resulttype	O	Usado para indicar se WFS deve gerar documento de respo completo (*results)* ou documento contendo some o número de linhas resulta de uma consulta (*hits)*.

EXPIRY=N	O	Tempo em minutos em que feição ficará bloqueada (u somente *GetFeatureWithLock)*

Tabela 6-Parâmetros requisição do GetFeature e GetFeatureWithLock (OGC, 2005)

LockFeature

O propósito desta operação é prover um mecanismo de bloqueio de determinada feição, quando a mesma estiver sendo alterada. É uma operação opcional.

Parâmetro	**Obrigatório(M)/Opcional(O)**	**Descrição**
VERSION=version	O	Número de versão
SERVICE=WFS	M	Tipo de serviço
REQUEST=LockFeature	M	Nome da requisição
TYPENAME=namespace:featuret e	M	Lista de tipos de feições selecionar, onde Namespac o domínio e featuretype é nome do tipo de feição
OPERATION=delete	M	Operação a executar
LOCKACTION=[ALL|SOME]	O	ALL tenta bloquear toda feições. SOME tenta bloqu o máximo possível.
FEATUREID=feature	O	Filtro para de determina feição
FILTER	O	Filtros a serem aplicados feições, separados conectores lógicos *and.* De haver um filtro para ca TYPENAME
BBOX=minx,miny,maxx,maxy	M	Se FEATUREID ou FILT forem usados, pode especificar uma caixa contorno.

Tabela 7-Parâmetros requisição do LockFeature (OGC, 2005)

Transaction

Utilizada em operações de inserção, exclusão ou alteração de feições.

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=version	O	Número de versão
SERVICE=WFS	M	Tipo de serviço
REQUEST=Transaction	M	Nome da requisição
OPERATION=delete	M	Operação a executar
RELEASEACTION=[ALL|SOME]	O	ALL denota que todos bloqueios de feições dev ser liberados quando transação terminar. SOM indica que somente modificados devem liberados quando a transaç terminar.
TYPENAME=namespace:featuretyp	M	Lista de tipos de feições selecionar, onde Namespa é o domínio e featuretype nome do tipo de feição
FEATUREID=feature	O	Filtro para de determina feição
FILTER	O	Filtros a serem aplicados feições, separados conectores lógicos *and*. De haver um filtro para ca TYPENAME
BBOX=minx,miny,maxx,maxy	M	Se FEATUREID ou FILT forem usados, pode especificar uma caixa contorno.

Tabela 8-Parâmetros requisição do Transaction (OGC, 2005)

3.1.2.3.4 *Web Coverage Service (WCS)*

Trata-se de um serviço específico ao tratamento de geo-campos (*coverages)*. Retorna dados sobre a semântica dos objetos representados (IGC, 2006a).

Operações WCS

DescribeCoverage

Recupera uma descrição completa das *coverages.*

Parâmetro	**Obrigatório(M)/Opcional(O)**	**Descrição**
VERSION=version	M	Número de versão
SERVICE=WCS	M	Tipo de serviço
REQUEST=DescribeCoverage	M	Nome da requisição
IDENTIFIERS=identifier	M	Lista de identificadores de g campos a serem descrit separados por vírgula

Tabela 9-Parâmetros requisição do DescribeCoverage (OGC, 2006a)

GetCovarage

Recupera propriedades de uma *coverage.*

Parâmetro	**Obrigatório(M)/Opcional(O**	**Descrição**
VERSION=version	M	Número de versão
SERVICE=WCS	M	Tipo de serviço
REQUEST=GetCoverage	M	Nome da requisição
IDENTIFIERS=identifier	M	Lista de identificadores geo-campos a ser descritos, separados vírgula
BBOX=minx,miny,maxx,maxy	M	Requisição de um subconju de *coverages* definido pe cantos da caixa de conto do mapa (inferior esquer superior direito)
TIMESEQUENCE=tempo	O	Requisição de um subconju correspondente aos instan de tempo, separados vírgula.
RANGESUBSET	O	Requisição de somente algu campos ou subconjuntos alguns campos
FORMAT= image/netcdf	M	Formato de saída
STORE= true	O	Especifica se a resposta de ser armazenada.

Tabela 10-Parâmetros requisição do GetCoverage (OGC, 2006a)

GetCapabilities

Fornece uma descrição do servidor, bem como uma série de informações acerca dos dados disponíveis.

Parâmetro	**Obrigatório(M)/Opcional(O)**	**Descrição**
VERSION=version	M	Número de versão
SERVICE=WCS	M	Tipo de serviço
REQUEST=DescribeCoverage	M	Nome da requisição
IDENTIFIERS=identifier	M	Lista de identificadores de g campos a serem descrit separados por vírgula

Tabela 11-Parâmetros requisição do GetCapabilities (OGC, 2006a)

3.1.2.3.5 *OpenGIS Catalog Service* (OCS)

A especificação *OpenGIS Catalog Services* (OCS) introduz um serviço para a publicação e busca em coleções de metadados de dados espaciais e objetos relacionados. Na especificação, é definida uma linguagem de consulta comum a todas as interfaces do serviço, a *Common Catalog Query Language*, As operações disponíveis no serviço são as seguintes (Davis, 2005):

• *getCapabilities*: recupera metadados descrevendo as características do servidor.

• *query*: executa uma consulta no catálogo

• *present*: recupera os metadados de uma ou mais referências.

• *describeRecordType*: retorna a definição do tipo de uma ou mais referências.

• *getDomain*: retorna o domínio de um atributo.

• *initialize*: inicia uma sessão interativa• close: encerra uma sessão interativa.

• *status*: recupera a situação de uma operação

• *cancel*: cancela uma operação.

• *transaction*: solicita um conjunto de ações de inserção, remoção ou alteração de itens do catálogo.

• *harvestResource*: recupera recursos de uma localização específica e que pode ser reprocessada de tempos em tempos.

• *order*: executa uma operação de compra de um recurso.

3.1.2.3.6 *Web Terrain Service*-WTS

O *Web Terrain Service* (WTS) incorpora modelos de elevação de terreno, com perspectiva e renderização tridimensional de mapas. As operações são as seguintes

- *getCapabilities e getMap* são idênticas às do WMS.
- *getView*: obtém uma cena 3D a partir do ponto de interesse, da distância e do ângulo entre o ponto de interesse e o observador da cena, do ângulo representando o campo de visão e do azimute.

3.1.2.3.7 *Web Coordinate Transformation Service* (WCTS)

Realiza de um sistema de coordenadas (CRS-*Coordinate Reference System)* para outro. O serviço recebe como entradas feições geográficas ou *coverages* e as retorna em outro sistema de coordenadas. Suas operações são as seguintes (Davis, 2005):

• *getCapabilities*: retorna as propriedades do servidor.

• *transform*: requisição para a transformação de coordenadas. Os CRS de origem e destino devem ser informados.

•*isTransformable*: indica se o servidor WCTS consegue processar a transformação entre dois CRS especificados.

• *getTransformation*: consulta às definições das transformações que o servidor pode processar de um CRS para outro.

• *describeTransformation*: recupera a definição de uma ou mais transformações pelo seu identificador.

• *describeCRS*: recupera a definição de um ou mais CRS com essa requisição.

• *describeMethod*: recupera uma ou mais definições de métodos das operações.

3.1.2.3.8 *Geolink Service*

O *Geolink Service* provê uma conexão entre um identificador geográfico para a geometria de um objeto (o geolink) e uma geometria em particular. As operações providas por esse serviço são:

- *getCapabilities*: retorna as propriedades do servidor.
- *Geolink*: acessa o geolink, retornando dados geográficos

3.1.2.3.9 *Web Gazeteer Service*

Acrescenta ao WFS operações de consulta, inserção e e objetos armazenados nos chamados *Gazeteers Digitais.* As operações são as mesmas do WFS.

3.1.2.3.10 *Web Registry Service*

Fornece uma estrutura para localização de servidores OpenGIS na *web.* Operações disponíveis:

- *getCapabilities*: retorna as propriedades do servidor.
- *getDescriptor:* retorna servidores que atendem a uma consulta específica.
- *registerService:* registra um servidor OpenGIS.

3.1.2.4 XML geográfico-*Geografic Markup Language* – GML

Seguindo a tendência do uso de padrões para intercâmbio de dados, o OGC usa o padrão XML (*eXtensible Markup Language*) para definir uma forma de codificar dados geográficos. Para isso especificou a linguagem GML (*Geography Markup Language*).

A GML foi especificada para o transporte e armazenamento de informação geográfica, incluindo propriedades espaciais e não espaciais das feições geográficas. O objetivo da GML é oferecer um conjunto de regras com as quais um usuário pode definir sua própria linguagem para descrever seus dados. Um esquema XML define as *tags* usadas em um documento que descreve os dados e contêm os modelos de geometria. Os principais são os seguintes (Davis, 2005):

- *Metadata*: Define as propriedades dos pacotes de dados que podem ser utilizados através de outros dados já existentes.
- *BasicTypes*:Engloba uma série de componentes simples e genéricos para representação arbitraria de atributos, nulos ou não.

- *Topology*: Especifica as definições do esquema geométrico dos dados, bem como sua descrição.
- *CoordinateReference Systems*: Determina sistemas de referência de coordenadas.
- *Temporal Information and Dynamic Feature*: Estende aos elementos características temporais dos dados geográficos e suas funções dinamicamente definidas.
- *Definitions and Dictionaries*: Definições das condições de uso dentro de documentos com certas propriedades ou informações de referentes à propriedade padrão.

3.2 A *API* do Google Maps

A API do Google Maps provê uma série de utilitários para manipular mapas, adicionando conteúdo aos mesmos através de uma grande variedade de serviços. Seus recursos principais são os seguintes:

- Conjunto de classes para exibir conteúdo em mapas
- Conjunto de classes para estender a API.
- Google AJAX API. Conjunto de classes para funcionamento do Google Maps juntamente com o AJAX.
- Google Ajax API *Loader* – Permite carregar aplicações baseadas na API juntamente com a Google AJAX API.

3.3 Padrões de Projeto

Segundo (Gamma, 1998) um padrão de projeto é uma solução padronizada para um problema que se repete muitas vezes dentro de um determinado contexto. Um padrão de projeto abstrai e identifica aspectos comuns de estruturas de projeto, tornando-as reutilizáveis. Portanto, cada padrão trata de um *problema* de projeto

em particular, que em última análise determina quando utilizar tal padrão. Cada problema tem uma *solução* proposta que é um gabarito a ser aplicado às mais diversas situações.

Dentre os 23 padrões de projeto propostos por (Gamma, 1998), os seguintes são considerados os mais populares e de maior utilização, inclusive para objetos gegráficos:

- Padrão *Composite* – Utilizado para estruturar objetos geográficos que exibam estrutura hierárquica do tipo árvore. Por exemplo, um mapa de municípios agregado por regionais e estas por bairros.
- Padrão *Flyweight* – Utilizado para representar estruturas de tamanho pequeno mas que existam em grande quantidade, como pontos e retas que constituem logradouros.
- Padrão *Chain of Responsibility* – Utilizado para enviar mensagens a um objeto pertencente a uma lista ou cadeia de objetos geográficos, como por exemplo, uma seqüência de pontos que representam semáforos em uma cidade.
- Padrão *Decorator* – Utilizado para acrescentar características adicionais a um objeto geográfico em tempo de execução. Por exemplo, um objeto geográfico pode inicialmente ser uma estrada de terra e posteriormente ser modificado para estrada asfaltada mediante uma decoração.
- Padrão *Abstract Factory* – Fornece uma interface para criação de famílias de objetos geográficos. por exemplo, criar famílias de mapas com projeções UTM e Lat/Long.
- Padrão *Adapter* – Intensivamente utilizado neste trabalho. Provê uma interface fixa para acesso às funcionalidades de classes existentes, de forma que uma futura mudança de API não interfira no código da aplicação.
- Padrão *Singleton* - Garante a existência de uma instância única para um objeto, quando necessário.
- Padrão *Proxy* – Garante que um objeto geográfico que ocupe muita memória (uma imagem de satélite, por exemplo), somente seja integralmente carregado quando necessário.

- Padrão *Bridge* – Diminui o número de combinações necessárias de objetos em classes filhas. Por exemplo, um objeto geográfico pode ser obtido da combinação de classes que determinem sua projeção (UTM e Lat/Long) e sua forma de armazenamento (*proxy* ou real)

3.4 Premissas Adotadas no Desenvolvimento da Proposta

Um processo de desenvolvimento está distribuído em *fases,* que possuem disciplinas *técnicas e gerenciais,* as quais expressam determinada área de conhecimento relevante ao desenvolvimento do projeto (IBM, 2003).. Atividades são o bloco principal de construção de um processo e constituem-se de *tarefas*, que as detalham. Estas são descritas por *passos* e realizadas por determinados *papéis,* produzindo, como resultado final de sua realização, *produtos de trabalho* (artefatos, diagramas, documentos, dentre outros). A Figura 24 a seguir esquematiza as relações descritas.

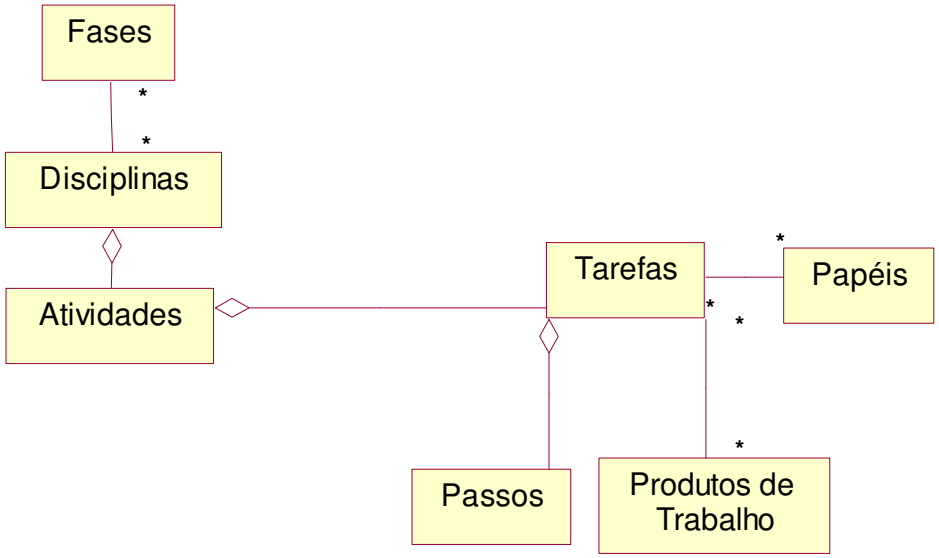


Figura 24-Elementos de um processo de desenvolvimento

Métodos para desenvolvimento de sistemas permitem o desenvolvimento organizado e disciplinado de atividades e tarefas definidas no processo de desenvolvimento.

Arcabouços arquiteturais expressam a estrutura física de classes e componentes de software, em diversas camadas ou níveis de abstração.

Ferramentas de apoio auxiliam na realização das atividades definidas no processo de desenvolvimento.

Para desenvolvimento da infraestrutura metodológica proposta nesta tese, as seguintes premissas foram consideradas:

- Suas fases devem ser calcadas em processos tradicionais de desenvolvimento do software. Para tal, tomaram-se com base o *Rational Unified Process* (IBM, 2003) e sua versão simplificada, o *Openup* (Kroll, 2007). Ambos têm sido referências consagradas no mercado.

- O processo deve ser adaptável a quaisquer projetos e ambientes. Assim sendo, o mesmo foi documentado na ferramenta EPF-*Eclipse Process Framework* (IBM, 2006), que permite a definição do processo em termos de todos seus constituintes e sua adaptação.

- O processo deve comportar atividades usuais de sistemas de *data warehouse.* Para tal, utilizaram-se como referência as atividades propostas por (Kimball,1998).

- O processo deve possuir uma componente espacial nas disciplinas, tarefas e artefatos onde for necessária, de forma a permitir a inclusão de particularidades a ela inerentes .

- A definição do arcabouço arquitetural calca-se nas seguintes arquiteturas e tecnologias:

    - CWM – *Common Warehouse Datamodel*, para definição da camada de metadados (Poole 2002 e 2003, OMG, 2003) ;
    - Arquitetura JEE (Sun 2007), para definição da camada web.

- Padrões OGC, para definição da estrutura de objetos de feições espaciais e de serviços web espaciais.
- EMF – *Eclipse Modeling Framework* (Budisnky, 2003), ferramenta de produtividade, que permite a geração de código a partir de modelos definidos em diagramas de classe da UML escritos na ferramenta *Rational Rose,* em Java anotado ou XMI (*XML Metadada Interchange*).
- Padrões de projeto (Gamma, 1998), para permitir a modelagem padronizada e intrinsecamente de melhor qualidade, em especial, de estruturas geográficas.

- A arquitetura de software deve ser flexível, de forma a permitir sua adaptação, em nível estrutural e funcional, a quaisquer tecnologias ou bibliotecas. Para tal, na especificação de interfaces para algoritmos, modelos e técnicas, utiliza-se intensivamente o padrão de projeto *adapter* (Gamma, 2008), que permite a definição de interfaces comuns que, por meio de polimorfismo, podem ser implementadas e adaptadas a quaisquer tecnologias e/ou bibliotecas.

- Os métodos de desenvolvimento utilizam a UML – *Unified Modeling Language* (descrita em Booch, 2003), linguagem gráfica para modelagem de sistemas em diagramas que expressam os diversos níveis de abstração utilizados para representar o sistema. O processo e a arquitetura proposta utilizam diagramas de casos de uso, para definição de requisitos, diagramas de classe, para modelagem da arquitetura e diagramas de interação, para modelagem de comportamento dinâmico.

3.5 -O processo Proposto

A Figura 25 exibe as diversas disciplinas do processo proposto, denominado *PADE-Processo para Análise de Dados Espaciais,* distribuídas pelas quatro fases do desenvolvimento: *concepção, elaboração, construção, implantação.* O PADE calca-se essencialmente nos processos *Rational Unified Process-RUP* (IBM, 2003) e

OPENUP (Kroll, 2007), deles importando e estendendo as fases de desenvolvimento e um grande número de atividades, tarefas, produtos de trabalho e papéis. Inspira-se também em diversas atividades e tarefas relacionadas ao desenvolvimento de *data warehouses* propostas por (Kimball, 1998)

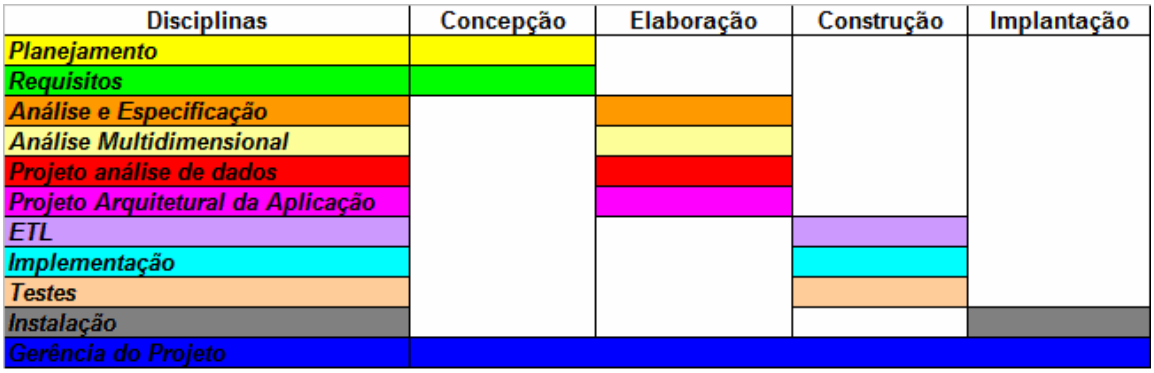


Figura 25-Processo PADE-Visão de disciplinas distribuídas pelas etapas.

Este processo foi totalmente definido na ferramenta *EPF-Eclipse Process Framework*. A estrutura do EPF é tal que permite a adequação do mesmo a um projeto específico, reutilizando-se componentes de processo. A seguir apresentam-se as descrições de cada disciplina do processo e os diagramas de atividades correspondentes. No Apêndice A as atividades de cada disciplina são detalhadas em tarefas, passos, papéis e artefatos que as constituem.

3.5.1 Etapa de concepção

O principal objetivo da fase de concepção é atingir o consenso entre todos os envolvidos sobre os objetivos do ciclo de vida do projeto (IBM, 2003). Esta fase concentra tarefas das disciplinas *Planejamento* e *Requisitos,* descritas a seguir.

3.5.1.1 A disciplina Planejamento

Esta disciplina envolve atividades e tarefas relacionadas com o planejamento do projeto em todas as suas etapas.

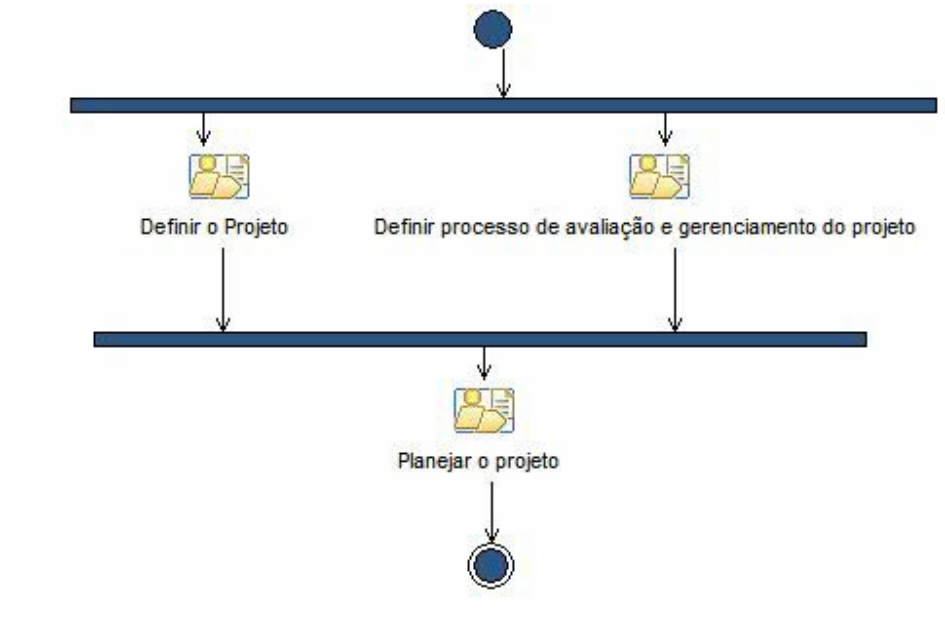


Figura 26-Diagrama de atividades da disciplina Planejamento

3.5.1.2 A disciplina Requisitos

Esta disciplina envolve atividades e tarefas para elicitar, analisar,especificar, validar e gerenciar requisitos.

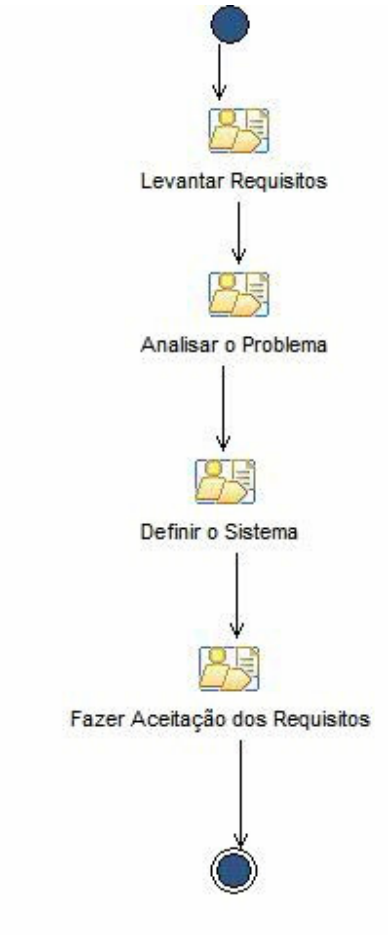


Figura 27-Diagrama de atividades da disciplina Requisitos

3.5.2 Etapa de elaboração

A etapa de elaboração envolve disciplinas relacionadas a análise de requisitos e projeto de toda a infraestrutura da aplicação.

3.5.2.1 A disciplina Análise e Especificação

A disciplina análise e especificação envolve atividades e tarefas de análise de requisitos, de forma a produzir uma especificação formal da aplicação.

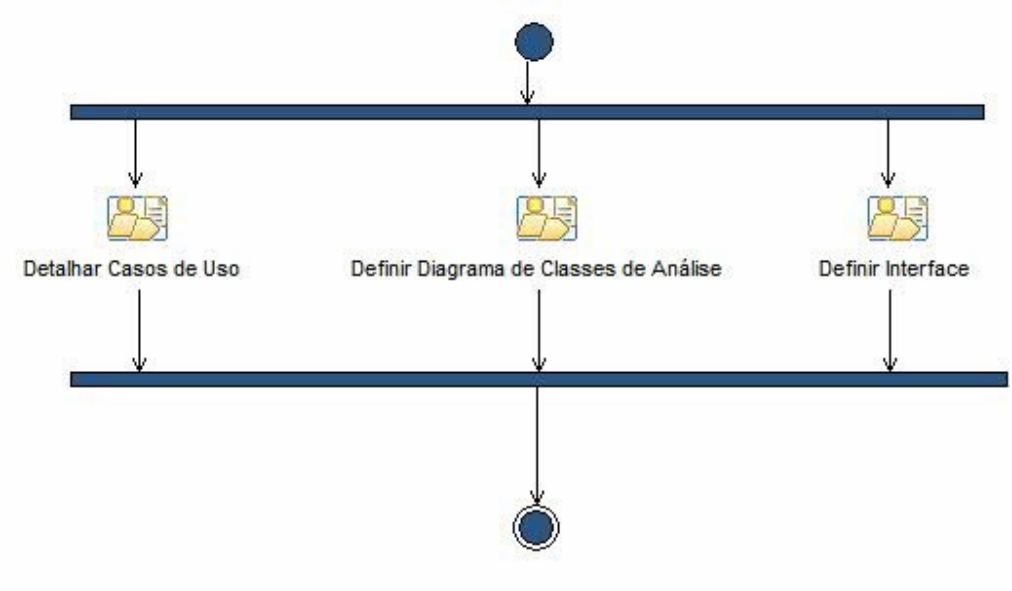


Figura 28-Diagrama de atividades da disciplina Análise e Especificação

3.5.2.2 A disciplina Análise Multidimensional

A disciplina análise e especificação envolve atividades e tarefas necessários à criação do modelo multidimensional, como definição de dimensões, medidas, tabelas e das fontes de dados, em um *data warehouse* convencional ou espacial.

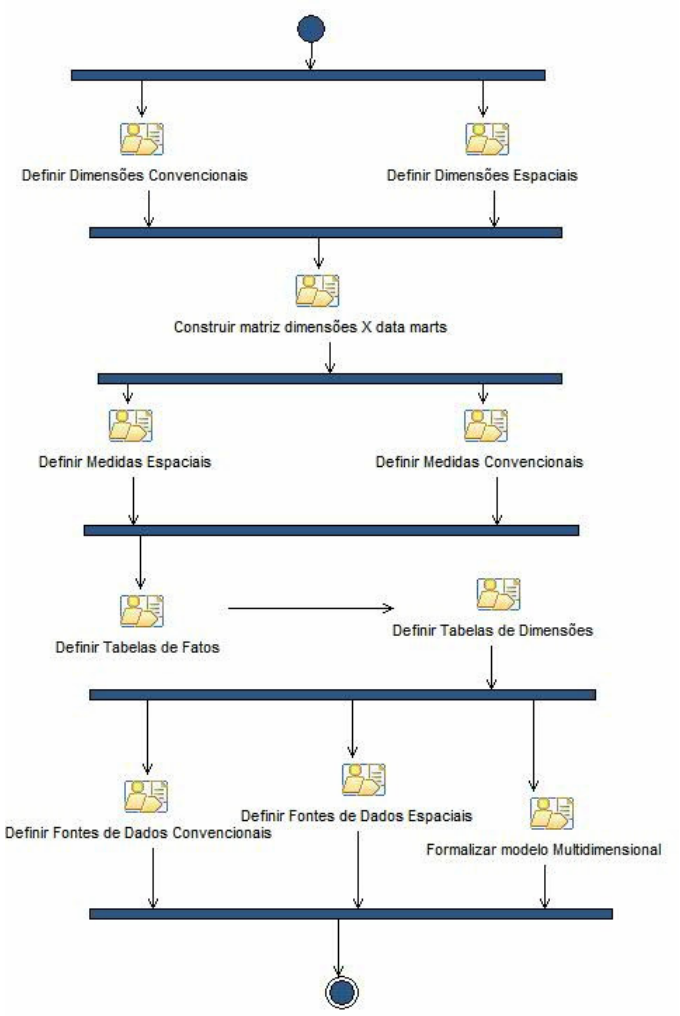


Figura 29-Diagrama de atividades da disciplina Análise Multidimensional

3.5.2.3 A disciplina Projeto Análise de Dados

Esta disciplina envolve atividades e tarefas necessários ao projeto de cubos convencionais e espaciais e de definição de técnicas de *data mining* convencional e espacial.

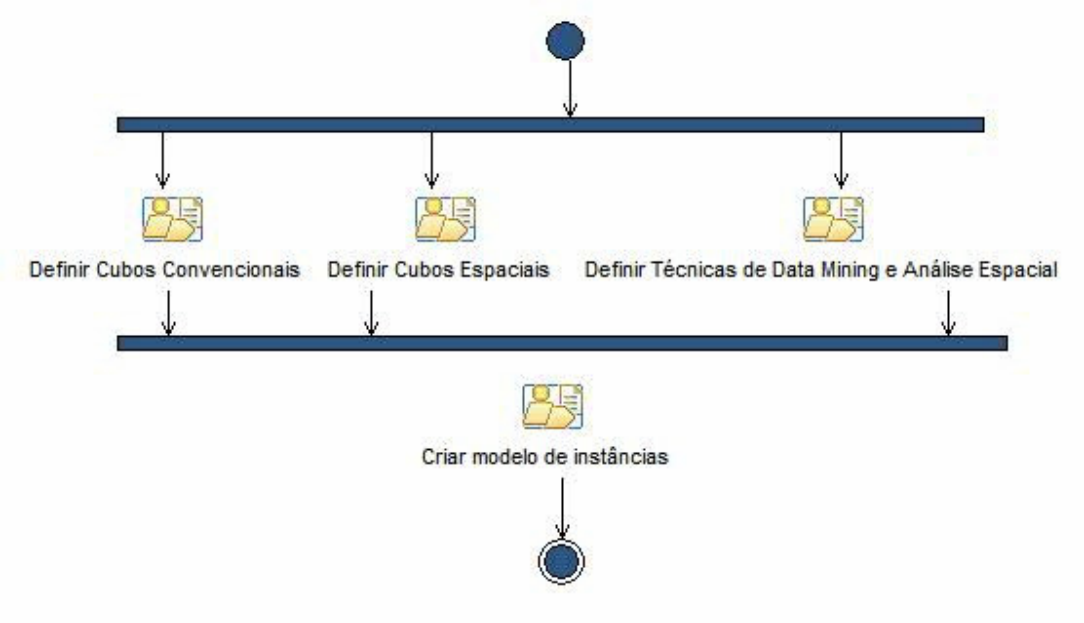


Figura 30-Diagrama de atividades da disciplina Projeto Análise de Dados

3.5.2.4 A disciplina Projeto Arquitetural da Aplicação

Esta disciplina envolve o projeto das três camadas (apresentação, controle e persistência) que definem a infraestrutura proposta neste trabalho.

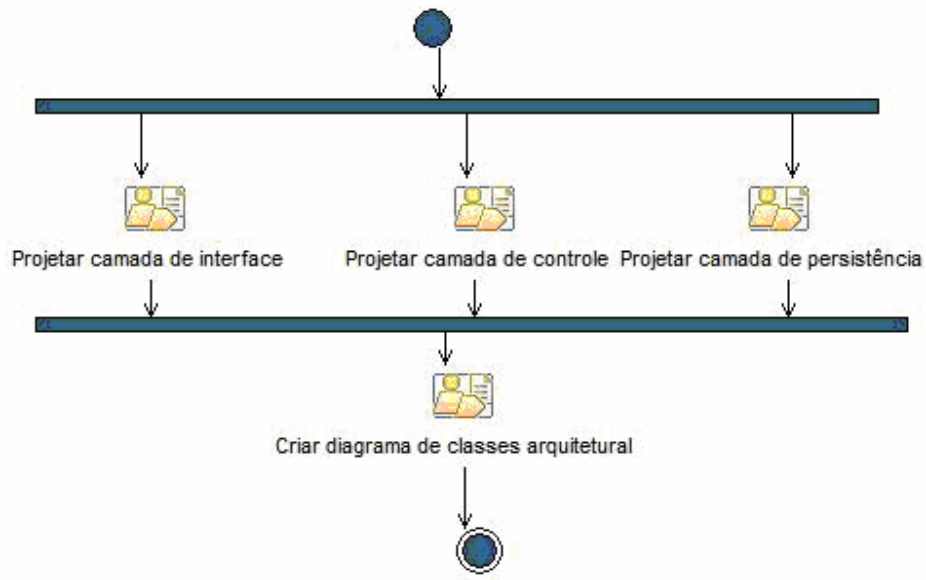


Figura 31-Diagrama de atividades da disciplina Projeto Arquitetural da Aplicação

3.5.3 A Etapa Construção

Esta etapa envolve disciplinas relacionadas à implementação da aplicação, ao teste da mesma e à importação de dados.

3.5.3.1 A Disciplina Migração de Dados

Esta disciplina envolve atividades de extração, transformação e carga de dados convencionais e espaciais.

Figura 32-Diagrama de atividades da disciplina Migração de Dados

3.5.3.2 A disciplina Implementação

Esta disciplina envolve atividades relacionadas à implementação das camadas de apresentação, controle e persistência da aplicação.

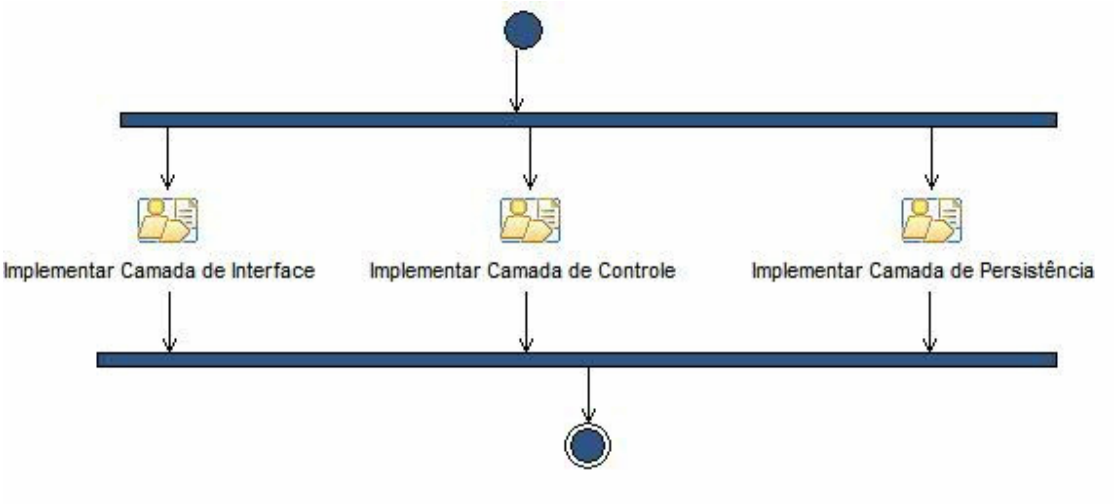


Figura 33-Diagrama de atividades da disciplina Implementação

3.5.3.3 A disciplina Testes

Esta disciplina envolve atividades relacionadas à definição, realização e verificação de resultados de testes aos níveis de unidade, integração, sistema, verificação, validação e aceitação.

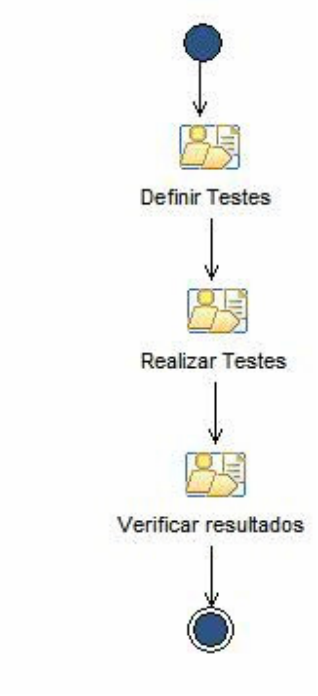


Figura 34-Diagrama de atividades da disciplina Testes

3.5.4 A Etapa Implantação

A etapa de implantação envolve a implantação da aplicação em seu ambiente operacional.

3.5.4.1 A disciplina Instalação

A disciplina instação envolve o planejamento e a realização da instalação da aplicação.

Figura 35-Diagrama de atividades da disciplina Instalação

3.5.5 A disciplina Gerência do Projeto

A disciplina gerência é realizada em toda as etapas e envolve atividades de verificação e gerenciamento do andamento do projeto e gerência de mudanças.

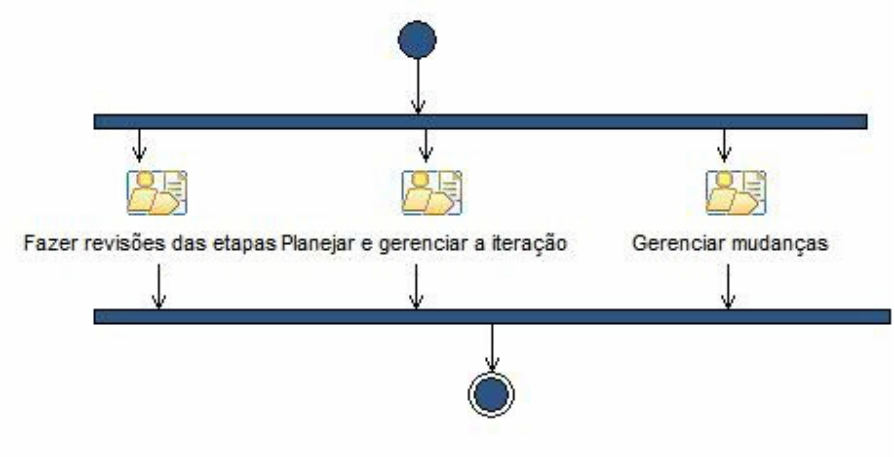


Figura 36-Diagrama de atividades da disciplina Gerência do Projeto

3.6 -Arquitetura de Software

Nesse item, será apresentada a arquitetura de software utilizada na infraestutura proposta. Ela foi desenvolvida com base nos seguintes critérios:

- *Componentização: Session Beans* (Sun, 2007) são nativamente utilizados para a criação de componentes reutilizáveis
- *Coesão: Session beans* são uma excelente infra-estrutura para encapsular a funcionalidade contida em regras de negócios ou algoritmos, aumentando sobremaneira o grau de coesão da aplicação.
- *Rapidez de desenvolvimento:* o EMF-*Eclipse Modeling Framework* (Poole, 2002 e 2003) foi escolhido como padrão para codificação de classes de domínio de problema e de metadados. Além de prover uma estrutura eficiente para geração automática de código a partir de modelos, possui um núcleo formalmente padronizado por meio de um arcabouço, o *Ecore (*Poole 2002 e 2003).

- *Facilidade para armazenamento persistente:* a tecnologia JEE JPA (Sun, 2007) é o mais robusto e consagrado arcabouço para implementação de persistência de informações armazenáveis em bancos de dados relacionais.
- *Independência entre interface e implementação:* para tal, utiliza-se intensivamente o padrão *adapter,* implementado nativamente por *Session Beans.*

Neste item descreve-se a arquitetura proposta. O detalhamento dos módulos em pacotes e diagramas de classes encontra-se no Apêndice B.

3.6.1 O Núcleo arquitetural

A Figura 37 abaixo é um metamodelo que exibe o núcleo arquitetural da infraestrutura de software, implementado sobre a tecnologia JEE (Sun, 2007).

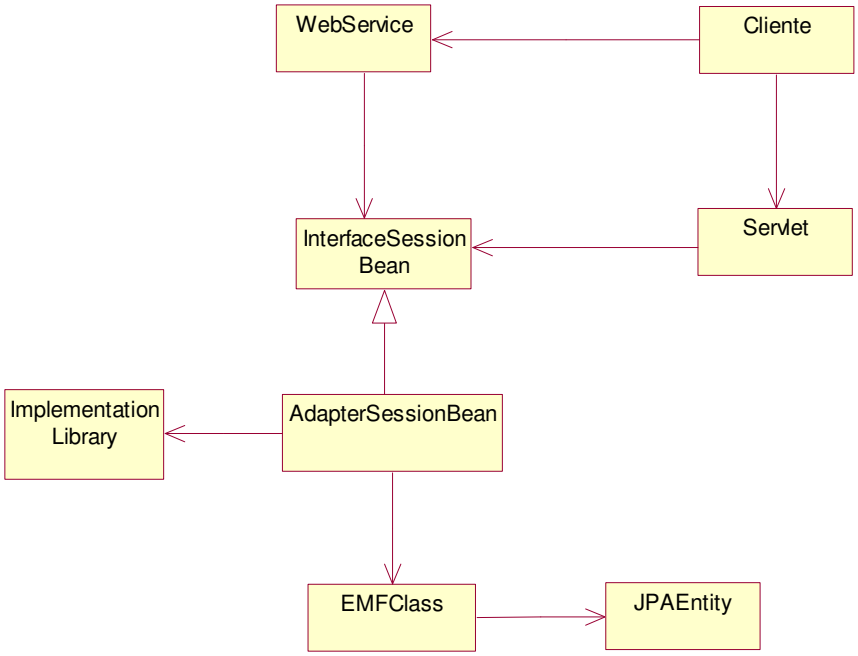


Figura 37-Núcleo Arquitetural do Arcabouço

Clientes acessam as funcionalidades providas via *Servlets* e *Web Services.* Servlets são classes Java executadas em servidores web. Servem para processar requisições provenientes de clientes e enviar respostas, via protocolo HTTP. *Web Services,* segundo o W3C (*World Wide Web Consortium)* "são sistemas de software projetados para suportar interação máquina-a-máquina via internet", ou seja, sua API pode ser acessada remotamente.

As regras de negócios são implementadas por *Session Beans,* classes que, através de seus métodos, disponibilizam funcionalidades (regras de negócio) a um cliente (Sun, 2007). Provêem um mecanismo bastante adequado à componentização, favorecendo, consequentemente, o uso e reuso de funcionalidades. Além disso, os *Session Beans* são estruturados por meio do mecanismo de polimorfismo no qual o acesso às funcionalidades é feito através de *interfaces,* as ISB-*InterfacesSessionBean.* Esse mecanismo permite que a implementação de tais interfaces possa ser modificada em função do uso de uma nova biblioteca ou algoritmo, conforme o padrão de projeto *Adapter* (Gamma, 1998) preconiza. Este padrão define uma interface-base, que contém métodos acessíveis pelos clientes. Esta interface é implementada por classes que acessam bibliotecas. No modelo aqui proposto a implementação de tais interfaces é feita nas classes do tipo ASB-*AdapterSessionBean* que na verdade apenas provêem acesso às bibliotecas de implementação (*ImplementationLibrary*), as quais contêm as implementações propriamente ditas. Caso novo algoritmo, implementação ou biblioteca sejam utilizados via novos pacotes ou classes presentes na *ImplementatinLibrary*, novas classes do tipo *AdapterSessionBean* são criadas, substituindo as antigas, provendo novo acesso às novas implementações, sem alterar a forma como os clientes acessam as funcionalidades,pois as assinaturas dos métodos não se alteram. As classes do tipo *EMFClass*, são classes geradas automaticamente pelo *EMF-Eclipse Modeling Framework* (Budinsky, 2003) aderentes ao arcabouço Ecore do EMF, cuja estrutura encontra-se descrita no apêndice B. As classes EMFClass provêem toda a funcionalidade necessária ao acesso às classes de domínio do problema. As classes *EMFClass* que necessitem persistência ao nível de bancos de dados relacionais o fazem através das classes

JPAE-JPAEntity, as quais são estruturadas por meio do arcabouço *JPA (Java Persistence API, Sun, 2007)* do JEE, que além de gerenciarem muito bem o mapeamento objeto/relacional, provêem recursos para a realização de consultas (*queries)*, de maneira transparente e amigável. As classes JPAEntity são na essência *JavaBeans* que acessam tabelas e campos do banco de dados, através de métodos para armazenamento (*setters)* e recuperação (*getters)* da informação.

Em nível de cliente utilizam-se as seguintes tecnologias para exibição e gerenciamento de interfaces com o usuário em ambiente *web:*

- Páginas JSP – *Java Server Pages.* Tecnologia que permite a geração de páginas dinamicamente, ao nível de servidor. Em essência, possui *tags* HTML entremeadas por *scripts* de código Java executado no servidor.
- JSF-*Java Server Faces* – Trata-se de uma interface com o usuário implementada em nível de servidor constituída pelos seguintes componentes principais:
    - Um conjunto de páginas JSP que contêm componentes de interface com o usuário.
    - Um conjunto de *backing beans,* que são classes Java do tipo *Java Beans* (classes javas que têm seus atributos acessados por meio de métodos de armazenamento (*setters)* e recuperação (*getters)* da informação) que definem propriedades e funcionalidades associadas a componentes de interface com o usuário na página.
    - Arquivos de configuração e descritores de instalação.
    - Objetos de interface adaptados pelo usuário.
- *Java Script* – Código Java embutido em páginas HTML, capaz de realizar operações em nível de cliente, no *web browser.*
- AJAX –*Assynchronous Java Script and XML* . Constitui um conjunto de tecnologias que permitem uma interação assíncrona entre o cliente e o servidor, de forma que recuperação de dados não interfira no comportamento

das páginas HTML, por evitar a recarga destas. O resultado são páginas com usabilidade superior e experiência de navegação mais agradável.

3.6.2 Arquitetura Geral

A Figura 38 exibe um esquema da arquitetura de software proposta. Ela se baseia essencialmente em orientação a serviços e na distribuição das classes em camadas lógicas, segundo o modelo *MVC- Model-View-Controler.* Este modelo é constituído por 3 camadas: a de *apresentação* (*View)* responsável pelo controle da interface com o usuário, a de *controle (Controler)* responsável pelas regras de negócio da aplicação, e a de *persistência (Model),* responsável pelo armazenamento e recuperação persistente da informação. A Figura 63 exibe as três camadas com os respectivos blocos e uma camada de *Extração, Implementação e Carga,* que não compõe especificamente a aplicação, mas realiza funções essenciais ao processo de obtenção de dados do *data warehoue* espacial. Tais camadas comunicam-se entre si e a dependência é vertical, de baixo para cima, ou seja uma camada superior depende do conteúdo da camada que lhe é imediatamente inferior. Nos itens a seguir cada camada e seus blocos são descritos de forma a apresentar os principais aspectos da arquitetura proposta para o arcabouço. O Apêndice B apresenta detalhes técnicos de implementação e os digramas de classes desta infraestrutura.

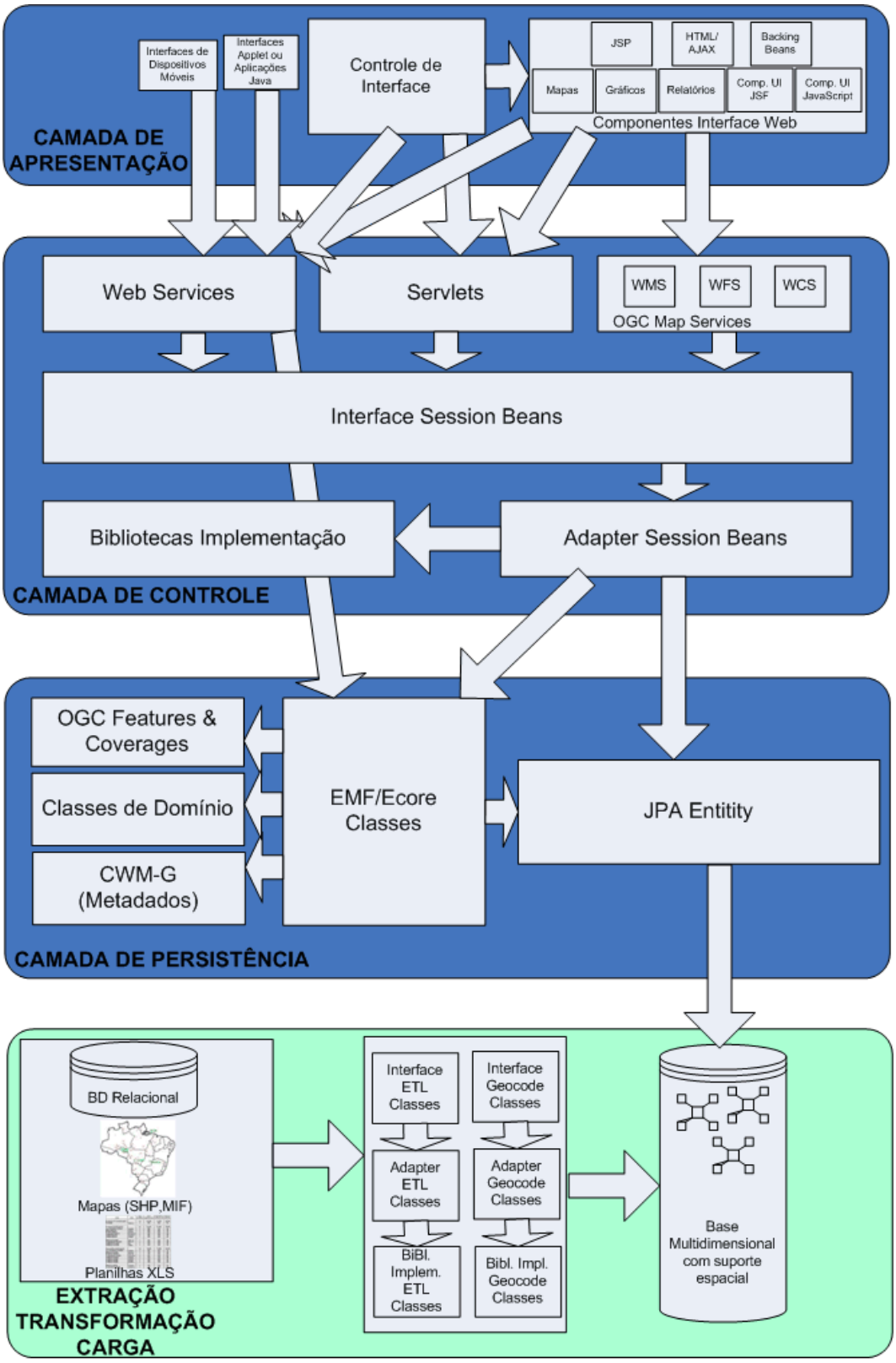
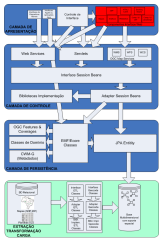


Figura 38-Infraestrutura geral da arquitetura de software proposta

3.6.3 –A Camada de Apresentação

3.6.3.1 -Módulo Componentes Interface *Web*



Este módulo comporta classes básicas de interface para exibição de mapas relatórios e gráficos e objetos básicos (listas, *combo-boxes,* botões, caixas de checagem, etc) por meio de tecnologias JSF (*Java Server Faces)* ou JavaScript/AJAX acessíveis respectivamente por arquivos JSP (*Java Server Pages*) ou HTML. Envolve três tipos de classes:

- *Interface Fixa:* corresponde ao conjunto de classes utilizadas para gerar o padrão de interface mostrado na Figura 39, que é a proposta para visualização de cubos em aplicações do *data warehouse.* O *layout* desta interface contém as seguintes áreas:

  - A área *definição de filtros e dimensões* contém *list-boxes* para obtenção de filtros aplicáveis às dimensões.
  - A área *seleção de dimensões* contém um *combo-box* para seleção da dimensão a ser visualizada.
  - A área *seleção de fatos* contém um *combo-box* para seleção do fato a ser visualizado.
  - A área *exibição de gráficos, relatórios e mapas* permite a visualização das informações na forma de gráficos torta ou barras , de relatórios sintéticos ou analíticos e em mapas (quando aplicável).

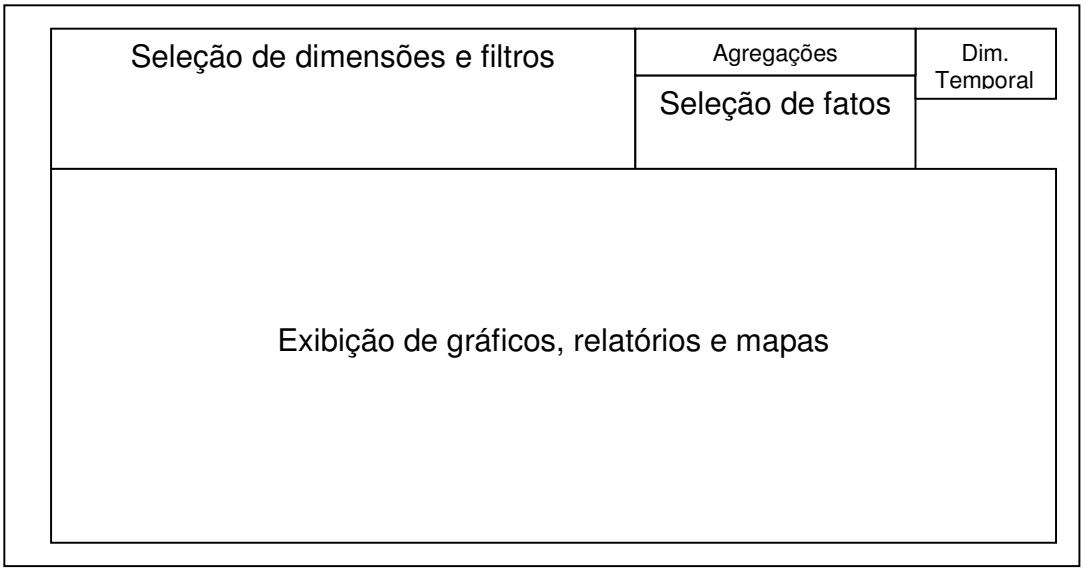


Figura 39-Padrão de interface

- *Interface Dinâmica:* esta interface, exibida na Figura 40, tem dois propósitos: permitir a seleção de dimensões e fatos de forma a gerar q*ueries* dinamicamente, exibidas em relatórios, gráficos e mapas, bem como permitir a geração rápida de uma Interface Fixa a partir dos fatos, dimensões e filtros selecionados, por meio de um código XML que pode ser traduzido em uma interface conforme descrita no *layout* exibido na Figura 39.
- *JSF BackingBeans:* envolve um conjunto de *backing beans JSF* que definem propriedades e métodos que implementam funcionalidades de interfaces JSF.

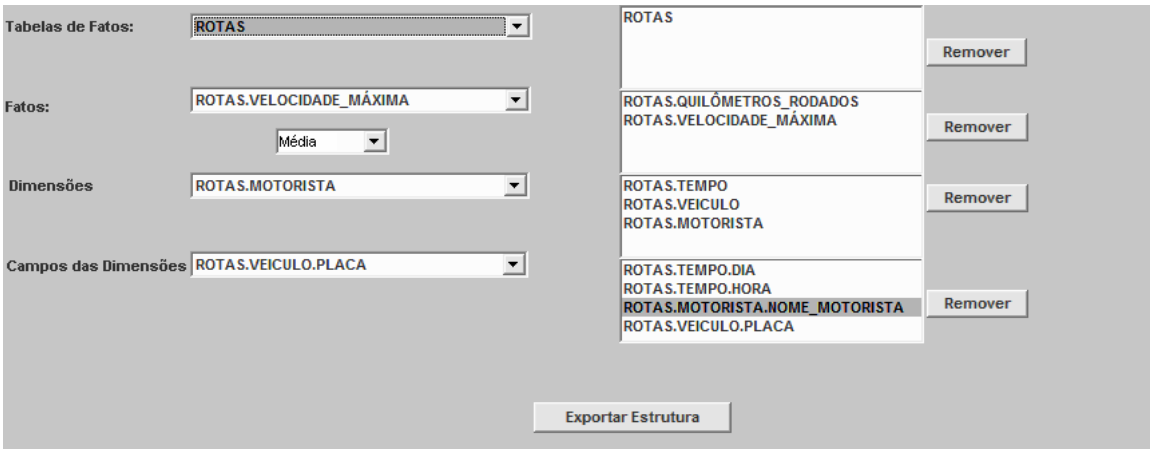
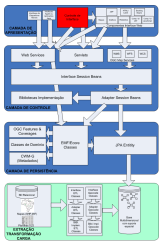


Figura 40- Layout da Interface Dinâmica

3.6.3.2 -Módulo Controle de Interface

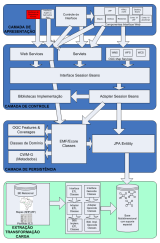


Este módulo possui classes que gerenciam a utilização de tecnologias JSF, JSP, Java Scripts(JS) e Ajax para exibição das interfaces. O controle da exibição de interfaces é todo feito em nível de servidor, por meio de *servlets*, *Java Beans* e JSPs. O padrão geral de interface é apresentado na Figura 41. Trata-se de um arquivo JSP, em que a interface *web* é dividida em setores, cuja disposição pode ser alterada livremente. Cada setor corresponde a um *frame* que podem exibir arquivos HTML, JSP ou texto livre.

<table>
<tr><td>Logo</td><td>Header</td><td>Header2</td></tr>
<tr><td colspan="3">Band</td></tr>
<tr><td>LeftSide1</td><td rowspan="4">Center</td><td>RightSide1</td></tr>
<tr><td>LeftSide2</td><td>RightSide2</td></tr>
<tr><td>LeftSide3</td><td>RightSide3</td></tr>
<tr><td>LeftSide4</td><td>RightSide4</td></tr>
<tr><td colspan="3">Footer</td></tr>
</table>

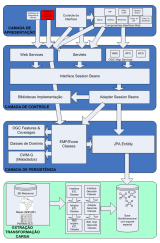
Figura 41-Template da estrutura sugerida para uma interface web

3.6.3.3 -Módulo Interface de Dispositivos Móveis



Este módulo tem como objetivo a implementação de classes que permitam a exibição de informações em dispositivos móveis como telefones celulares, PDAs, *Pocket PCs, Palm Tops,* dentre outros. Não serão detalhadas classes deste módulo pois existe um número muito grande de modelos e de linguagens de implementação para os mesmos, os quais têm em comum o fato de acessarem os serviços disponibilizados no arcabouço via *Web Services.*

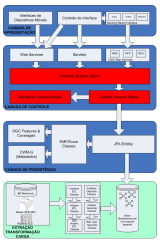
3.6.3.4 -Módulos Interfaces Applet ou Aplicações Java



Este módulo contém classes de interface que implementam *applets* e aplicações Java convencionais, por meio, por exemplo, de componentess AWT ou Swing. O acesso a dados ou a funcionalidades proporcionadas pelas classes ISB é feito, principalmente, por meio de *web services.*

3.6.4 -A Camada de Controle

3.6.4.1 -Módulos relacionados a *Session Beans*



Os módulos *Interface Session Beans, Adapter Session Beans* e *Bibliotecas de Implementação* constituem o núcleo funcional do arcabouço. Eles descrevem e implementam todas as interfaces de acesso a regras de negócio, algoritmos e bibliotecas, além de prover acesso a classes da camada de persistência. O núcleo arquitetural descrito no item 3.6.1 é, principalmente, aqui instanciado. As classes do tipo *Interface Session Bean* -ISB têm sua interface detalhada no APÊNDICE C, de forma que possam ser implementadas por classes *Adapter Session Bean*-ASB, via acesso a *Bibliotecas Implementação,* que contêm pacotes e classes prontas com a implementação da funcionalidade. Ressalta-se novamente que essa estrutura provê flexibilidade na alteração de implementações, de forma que as classes e pacotes contidas nas Bibliotecas de Implementação possam ser substituídas, sem impacto na aplicação que as acessa, pois estas o fazem via ISBs, que não se alteram. Desta forma, quando houver necessidade de alterações, implementam-se novas classes ASB específicas sem, entretanto, qualquer alteração na assinatura de seus métodos.

3.6.4.1.1 Pacotes para Análise de Dados

Na versão atual, estes pacotes contêm definições de interfaces relacionadas a algumas técnicas de análise de dados, definidas em função de sua popularidade ou utilização em projeto específico. Encontram-se distribuídas nos seguintes pacotes(Figura 42):

TecnicasDataMining : Especifica interfaces para implementações de algoritmos de *data mining* convencionais e geográficos. As interfaces que implementam as versões convencionais e geográficas estão divididas em pacotes que implementam algoritmos de *clustering,* regras de associação, regras de associação espacial e regras de classificação, nas versões convencional e espacial.

TecnicasdeIA: Especifica interfaces para implementações de redes neurais e algoritmos genéticos

ModelosAnaliseEspacial: Especifica interfaces para implementações de modelos de análise espacial

AnaliseOLAP: Especifica interfaces para implementações de OLAP convencional e geográfico

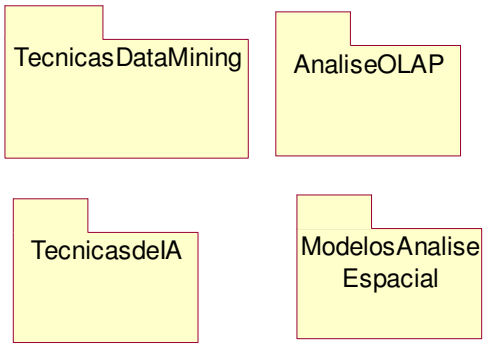
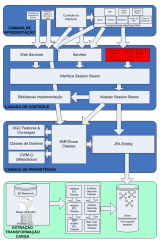


Figura 42 -Pacotes de Análise de Dados

3.6.4.2 Módulos OGC Map Services



3.6.4.2.1 Pacote de Interfaces para Serviços OGC

Este pacote possui interfaces ISB para implementação de alguns serviços OGC, conforme a descrição apresentada no item 3.1.2.3. As interfaces especificadas são as seguintes:

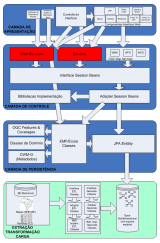
- *ISBWMS:* especifica operações do serviço OGC *Web Map Services, (WMS);*
- *ISBWFS:* especifica operações do serviço OGC *Web Feature Services, (WFS);*
- *ISBWCS:* especifica operações do serviço OGC *Web Coverage Services, (WMS);*

Estas interfaces são acessadas via *web services, servlets* ou arquivos JSP, que provêem acesso às mesmas.

As interfaces ISBWMS e ICBWCS foram implementadas por meio das classes ASBWMS e ASBWCS que utilizam respectivamente as APIs do *Google Maps* para acesso a mapas e do pacote *Geotools* , que possui implementação de todos os serviços OGC, inclusive do WCS. A aplicação-piloto (item 4.6) e a aplicação para consultas em bases de dados multidimensionai (item 4.5) utilizam *servlets* e JSPs para acesso às classes ASBWCS e ASBWMS implementadas.

As interfaces deste módulo foram apresentadas separadamente para que seja destacada sua importância dentro da arquitetura proposta.

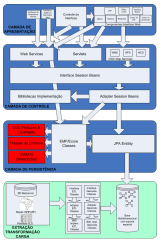
3.6.4.3 *Servlets e Web Services*



Este módulo provê todas as interfaces para conexão entre as camadas de controle e apresentação da arquitetura, principalmente para as classes de análise (item 3.6.4.1.1). *Servlets e Web Services* (Sun, 2007) são tecnologias *web* que permitem a comunicação de dados e funcional (pela execução de procedimentos) entre camadas da aplicação. Provêem o acesso da camada de apresentação às ISBs definidas na camada de controle.

3.6.5 A camada de Persistência

3.6.5.1 Classes de Domínio da Aplicação



Este módulo especifica todas as classes de domínio de problemas que, em última instância, constituem o metamodelo da aplicação. Três categorias de classes foram definidas na arquitetura proposta:

Features e Coverages OGC

Especificam os modelos *Feature e Coverage* do OGC, apresentados no item 3.1.2.2.

Classes de domínio do problema propriamente ditas

Especificam classes de domínio de problema para análise de dados,

Classes CWM-G

Este pacote estende os pacotes OLAP e *data mining* do modelo CWM-*Common Warehouse Metamodel* (Poole, 2003), de forma a torná-lo adequado ao tratamento de informações espaciais. Isto é feito pelo acréscimo de hierarquias, níveis, dimensões, medidas e atributos espaciais.

O CWM é um modelo de domínio genérico para *data warehouses* constituído por um conjunto de módulos que contêm classes as quais descrevem diversos aspectos dos mesmos. A Figura 43 exibe a estrutura do CWM em camadas que contêm tais módulos. Neste trabalho, foram realizadas extensões sobre os módulos *multidimensional e data mining,* detalhadas no Apêndice B.

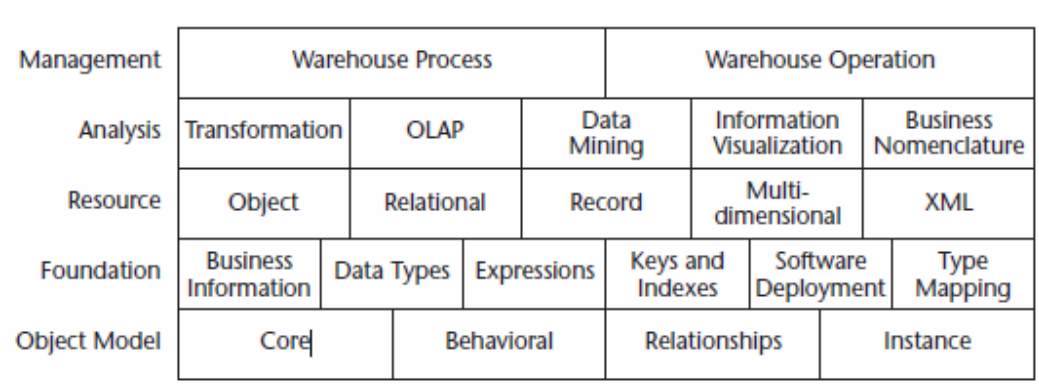
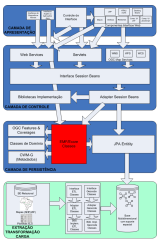


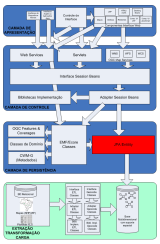
Figura 43- Estrutura em camadas do CWM. Extraído de (Poole, 2003)

3.6.5.2 Classes EMF/Ecore



O Modelo ECore (Budinsky, 2003) do *Eclipse Modeling Framework (EMF)* é a base para criação de *abstract factories* (padrão de projeto descrito em (Gamma,1998)) utilizadas para instanciar as classes de domínio de problema. O EMF provê um mecanismo eficiente para geração rápida do código Java a partir do modelo, que pode estar, inclusive, em formato MDL da ferramenta *Rational Rose.*

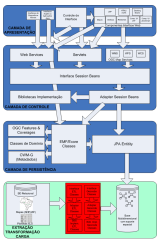
3.6.5.3 JPA *Entity*



A JPA-*Java Persistense API* (Sun, 2007) é a base utilizada para implementação de persistência. As classes JPA provêem um mecanismo para inserção, exclusão e alteração de dados em uma base de dados, bem como um mecanismo de implementação de relacionamentos entre classes e de geração de *queries*. Na aplicação desenvolvida para consulta a bases multidimensionais (ver item 4.5) , classes JPA foram criadas para acesso a dados nas tabelas de fatos e dimensões e para a realização das consultas (*queries).*

3.6.6 Extração-Transformação-Carga

3.6.6.1 Classes ETL e de Geocodificação



Este módulo especifica interfaces para extração de dados espaciais e georreferenciamento, bem como para realização de operações de tratamento de dados. As interfaces para extração de dados foram implementadas e uma aplicação foi construída para realizar a extração de múltiplas origens e georreferenciamento de dados provenientes de arquivos MIF e SHP. A Figura 44 exibe a interface desta aplicação. No painel da esquerda, escolhem-se as fontes de dados: banco de dados (via ODBC), planilhas Excel (CSV), arquivos Mapinfo (MIF) e ArcGis (SHP). Estas fontes são associadas a fatos e dimensões de um *data mart,* listadas no painel da direita. Pressionando-se o botão V , a tabela na parte inferior da tela armazena as associações. Pressionando-se o botão extrair, os dados (convencionais e espaciais) são armazenados em um banco de dados.

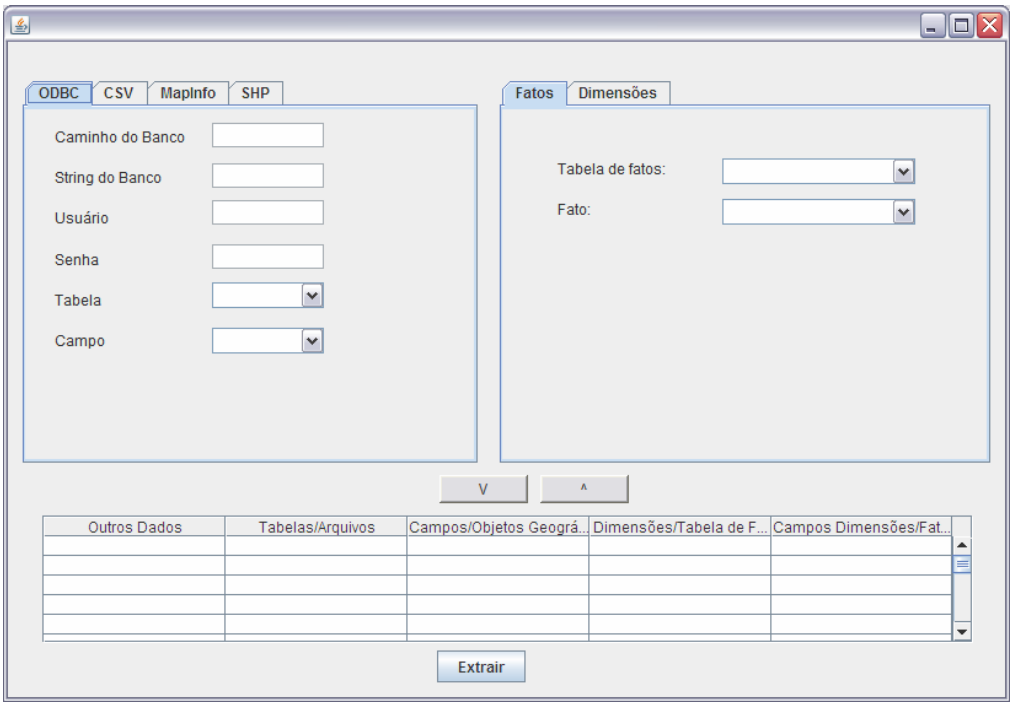
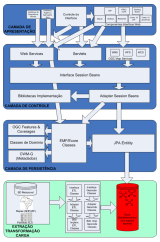


Figura 44-Interface para extração e georreferenciamento de dados

3.6.6.2 Bases de dados Multidimensionais Espaciais



Este módulo comporta bases de dados com extensão espacial, que armazenam tabelas de modelos multidimensionais. Bancos de dados com MySQL espacial, PostGIS e Oracle *Spatial* possuem extensões que implementam as especificações espaciais do OGC. As aplicações desenvolvidas nesta tese utilizam o banco de dados MySQL.

3.7 – Ferramentas de apoio

Para garantir rapidez no desenvolvimento e controle sobre as principais atividades do processo, as seguintes ferramentas de software são recomendadas, selecionadas por notável reconhecimento, funcionalidade e, em alguns casos, por serem *open source*:

Ferramentas de apoio à disciplina de implementação

Eclipse- Trata-se de uma interface gráfica para desenvolvimento de código em linguagem Java. É uma ferramenta de código livre e que possui como característica notável uma grande capacidade de extensão de suas funcionalidades pela inclusão de *plugins* desenvolvidos por terceiros. Maiores informações sobre a mesma podem ser encontradas em (IBM,2009).

EMF- É um *plugin* do Eclipse que permite a geração de código-fonte Java a partir de diagramas de classes UML do Rational Rose, código Java anotado ou código XMI (*XML Metadata Interchange).* O código é gerado segundo o arcabouço Ecore, cuja estrutura encontra-se descrita em (Kroll, 2007).

Ferramentas de apoio às disciplinas de planejamento e gerência de projetos

EPF- É uma ferramenta CASE para criação de processos de desenvolvimento. Foi utilizada para definir o processo proposto neste trabalho e facilita sobremaneira o pocesso de adaptação do processo a um projeto ou ambiente específico. Encontra-se descrita em (IBM, 2006).

MS Project- Popular ferramenta para gerência de projetos da Microsoft. Atividades, tarefas e passos gerados no EPF podem ser exportados para utilização nesta ferramenta.

CVS – Control Version System. Ferramenta *open source* para gerenciamento de mudanças em artefatos (http://www.nongnu.org/cvs/).

Ferramentas de apoio às disciplinas análise e especificação e projeto arquitetural da aplicação

Rational Rose – Ferramenta CASE desenvolvida pela *Rational Software Corporation,* atualmente sob o controle da IBM. Permite a criação de todos os diagramas da UML, bem como a geração de código em linguagens C++ e Java.

Case Studio. Existe um grande número de ferramentas para modelagem de dados, mas esta se destaca por gerar código SQL de boa qualidade para um grande número de bancos de dados existentes no mercado.

Argo UML. Ferramenta CASE *open source* desenvolvida na UCLA (*University of Califórnia at Los Angeles),* para criação de todos os diagramas da UML. Boa alternativa ao *Rational Rose,* mas seus diagramas não são importados pela ferramenta EMF.

Ferramenta de apoio à disciplina requisitos

Open Source Requirements Management Tool. Ferramenta de código aberto que fornece suporte ao processo de geração e rastreamento de requisitos. Maiores informações podem ser obtidas em http://freshmeat.net/projects/osrmt/.

Requiste Pro. Poderosa ferramenta para gerenciamento de requisitos da *Rational Software Corporation.* Recomendada quando existirem recursos para investimento em ferramentas mais robustas

Ferramenta de apoio à disciplina testes

Junit - Trata-se de um arcabouço para execução de testes e conjuntos de teste, suportado pela ferramenta Eclipse.

Ferramenta de apoio à disciplina análise multidimensional

Recomenda-se o uso da ferramenta para consulta em bases de dados multidimensionais (item 4.5), que permita a geração do modelo multidimensional a partir da seleção de fatos e dimensões.

Ferramenta para a disciplina ETL

Recomenda-se o uso da ferramenta para ETL desenvolvida como parte deste trabalho (item 3.6.6.1).

Bancos de dados com suporte espacial

Atualmente os bancos de dados Oracle *Spatial* e PostGis apresentam implementações completas dos padrões OGC para bancos de dados espaciais. O banco de dados MySQL possui uma implementação incompleta, mas espera-se para breve a conclusão desta implementação.

4 - Aplicativos Desenvolvidos

Neste capítulo são apresentadas implementações de algumas interfaces , bem como da aplicação-piloto.

4.1 O Aplicativo Modelo Têmporo-Espacial de Hägerstrand

A dimensão espacial é uma das partes mais importantes da representação da informação geográfica. Isto é válido quando considerados fenômenos e eventos estáticos, ao menos em uma escala de tempo macroscópica. Entretanto, quando se consideram as atividades humanas, a dimensão tempo torna-se bastante relevante (Abreu & Miranda Jr, 2007a).

O geógrafo Törsten Hägerstrand, na década de 1960, notabilizou-se por integrar o tempo ao comportamento espacial como uma dimensão intrínseca, através de um diagrama tempo- espacial. Entretanto, segundo (Adams, 1996), Hägerstrand restringiu sua visão das atividades humanas ao movimento do corpo, limitando suas interações espaciais aos objetos ao seu alcance, ignorando a natureza social do ser humano, capaz de estender tais interações a escalas muito maiores (Abreu & Miranda Jr, 2007a).

Extensões ao diagrama tempo-espacial de Hägerstrand têm sido propostas. (Adams, 1996) com seu diagrama de extensibilidade, representa em uma escala temporal as atividades realizadas por um indivíduo e a escala de transação de tais atividades nos níveis próximo, metropolitano, estadual, regional, nacional e internacional.. (Kwan, 1995) criou os chamados caminhos espaço-tempo individuais em múltiplas escalas, uma variação do trabalho de (Adams, 1996).

4.1.1 -O diagrama têmporo-espacial multi-escopo

(Abreu & Miranda Jr, 2007a) definem o diagrama têmporo-espacial multi-escopo, que representa as transações de um indivíduo em termos das dimensões tempo e espaço em até 4 escopos de interação, cada um deles a uma escala espacial distinta:

- Escopo local (próximo)
- Escopo estadual
- Escopo nacional
- Escopo internacional

Três eixos de coordenadas são utilizados para representar a informação, um para tempo e os outros dois para latitude e longitude. Cada escopo é representado em uma camada diferente do diagrama, de forma que todas as transações podem ser simultaneamente visualizadas. A Figura 45 apresenta o aspecto geral do diagrama. Os cilindros em azul representam os tempos gastos em cada atividade, ou seja, a base do cilindro corresponde ao início da atividade e o topo ao fim (que podem ser lidos no eixo vertical). As linhas que partem da base dos cilindros para os mapas nas camadas indicam o local exato onde ocorreu a transação. As linhas azuis conectando os cilindros representam a seqüência em que as atividades ocorreram. Neste exemplo, as camadas representam os mapas de Belo Horizonte, do Estado de Minas Gerais, do Brasil e das Américas do Norte/Central e Europa, correspondentes ao escopo das transações que o indivíduo realizou.

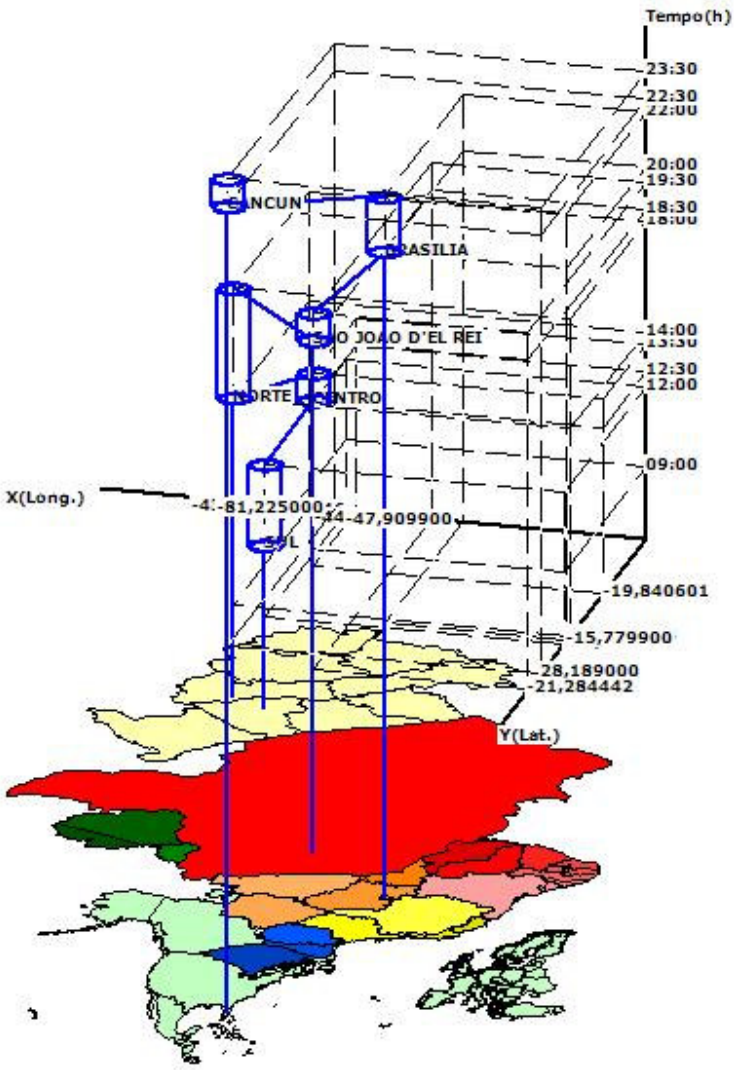


Figura 45 – Aspecto Geral do Diagrama Têmporo-Espacial Multi-Escopo. Fonte: (Abreu & Miranda Jr, 2007a)

4.1.2 –O Aplicativo

4.1.2.1 –Características Principais

A Tabela 12 exibe os dados utilizados nos diagramas desta seção.

Hora	Min.	Tempo Base	Desloc.	Estação	Long.	Lat.	Atividade	Escopo
9	0	3	11,12	SUL	-43,97361786	-19,94045671	Trabalho 1	L
12	30	1	9,68	CENTRO	-43,96854571	-19,84060071	Almoço	L
14	00	4	154,26	NORTE	-43,993574	-19,923958	Trabalho 2	L
18	30	1	733,92	SAO JOAO D'EL REI	-44,266118	-21,284442	Fala com filial por telefone	E
20	0	2	6133,09	BRASILIA	-47,9099	-15,7799	Contata cliente por telefone	N
22	30	1		CANCUN	-81,225	28,189	Busca preço de hotéis via internet	I

Tabela 12 - Dados utilizados no diagrama-exemplo. Fonte:(Abreu & Miranda Jr, 2007a)

A aplicação desenvolvida permite a visualização das diversas camadas de escopo separadamente, conforme exibido na Figura 46.

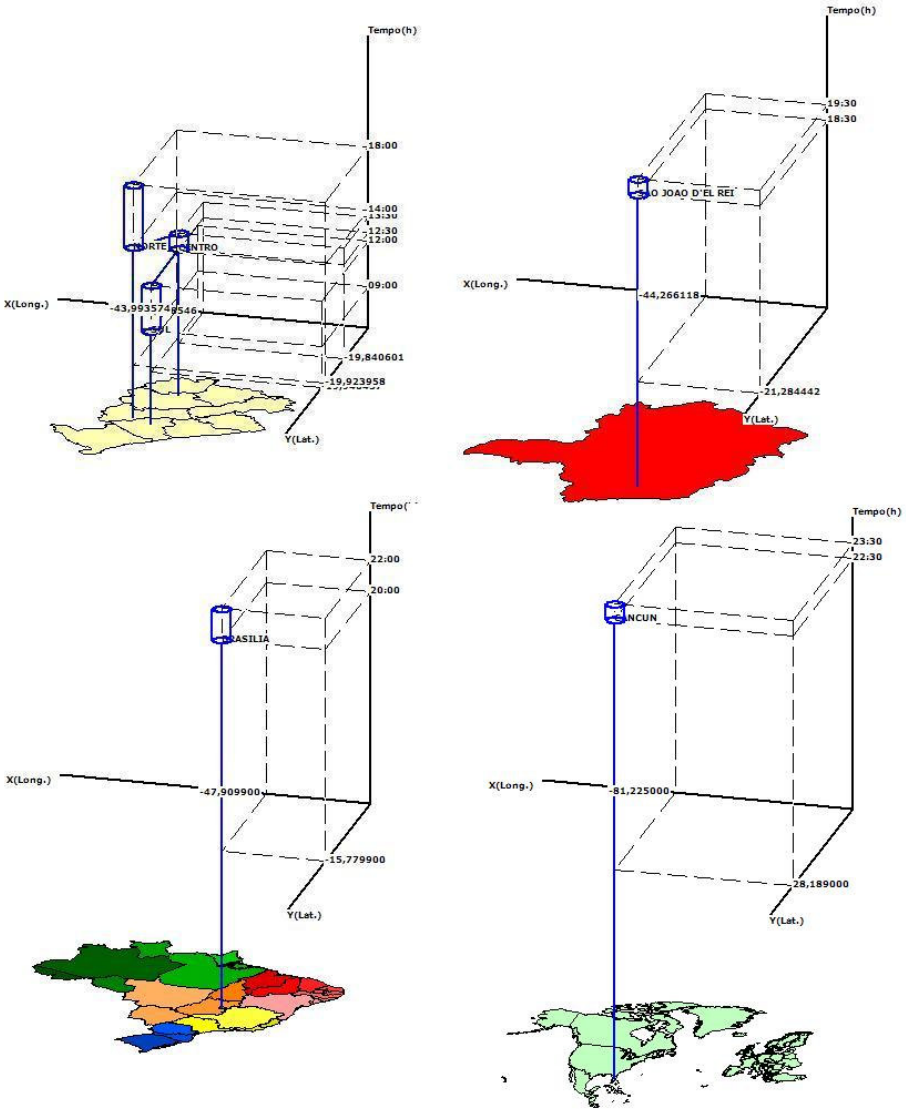


Figura 46 - Visualização isolada de cada camada do diagrama. Fonte:(Abreu & Miranda Jr, 2007a)

O diagrama pode ser visto sob diversas perspectivas, por meio da rotação dos três eixos. Na Figura 47, exibe-se apenas a perspectiva espacial para o escopo local (Belo Horizonte), podendo-se visualizar claramente os deslocamentos efetuados.

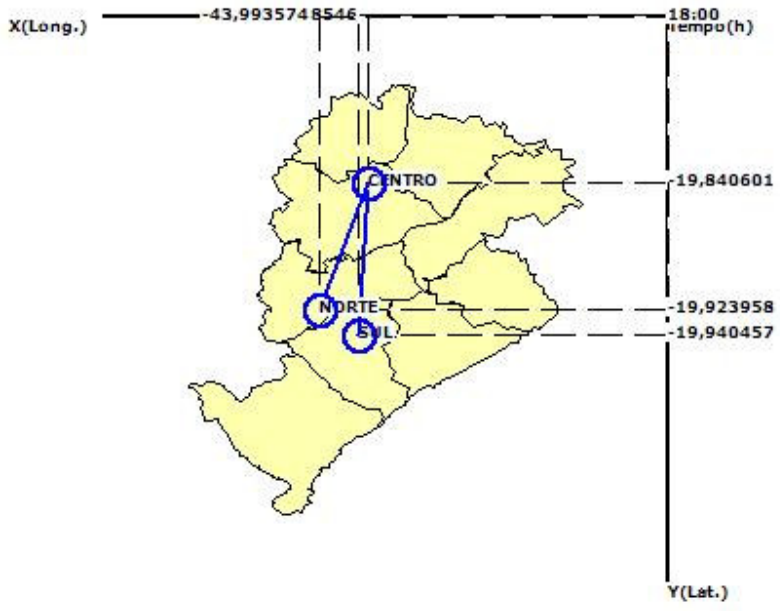


Figura 47- Visualização das dimensões espaciais, obtida por rotação dos eixos. Fonte:(Abreu & Miranda Jr, 2007a)

Na Figura 48, exibem-se apenas as dimensões tempo e longitude, podendo-se facilmente visualizar os tempos gastos em cada atividade, o tempo de deslocamento entre elas, e o escopo exato de cada uma delas (através das linhas azuis que partem da base do cilindro e chegam às linhas horizontais que representam as camadas).

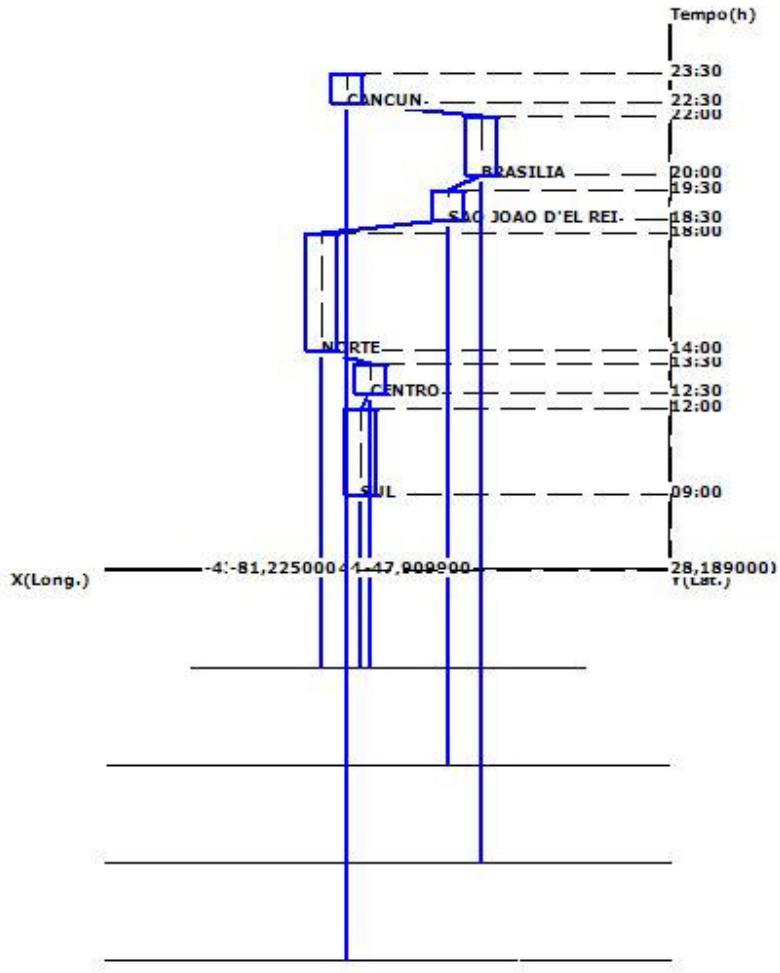


Figura 48- Visualização das dimensões tempo e longitude. Fonte:(Abreu & Miranda Jr, 2007a)

4.1.2.2 Interface da aplicação

A Figura 49 exibe a janela para entrada de dados necessários à confecção do diagrama. A tabela deve ser alimentada em cada linha com dados das atividades realizadas pelo indivíduo, informando-se respectivamente a *hora/minuto* de início da atividade, o *tempo* de realização da atividade, o *local* onde a atividade foi realizada, *longitude* e *latitude,* uma descrição breve da *atividade* realizada e o *escopo* (L=

local, R= regional, N=nacional e I= internacional). Estes dados podem também ser importados de um arquivo Excell/CSV. Mais abaixo, no painel *escopo,* devem-se informar os arquivos que contêm os mapas a serem exibidos nas camadas (apenas formatos MIF). O *check-box visível* permite selecionar qual (is) camadas será (ão) visualizada (s).

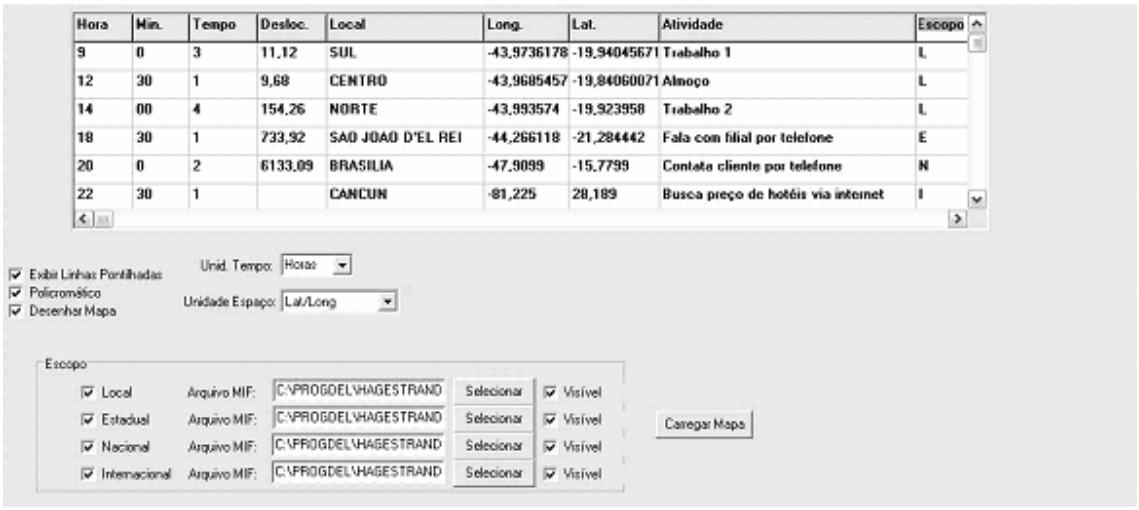


Figura 49 - Janela para entrada de dados do diagrama. Fonte:(Abreu & Miranda Jr, 2007a)

A Figura 50 exibe a janela para movimentação do diagrama. A rotação pode ser feita nos planos vertical ou horizontal do diagrama, em unidades de ângulo (grau). Pode-se também deslocar o diagrama pela tela (*translação,* alternativamente pode-se arrastar e soltar*)* ou ampliar/reduzir o mesmo (*zoom).*

Figura 50- Janela para movimentação do diagrama. Fonte:(Abreu & Miranda Jr, 2007a)

A Figura 51 exibe o aspecto final de um diagrama com várias camadas de mapas gerado pelo aplicativo.

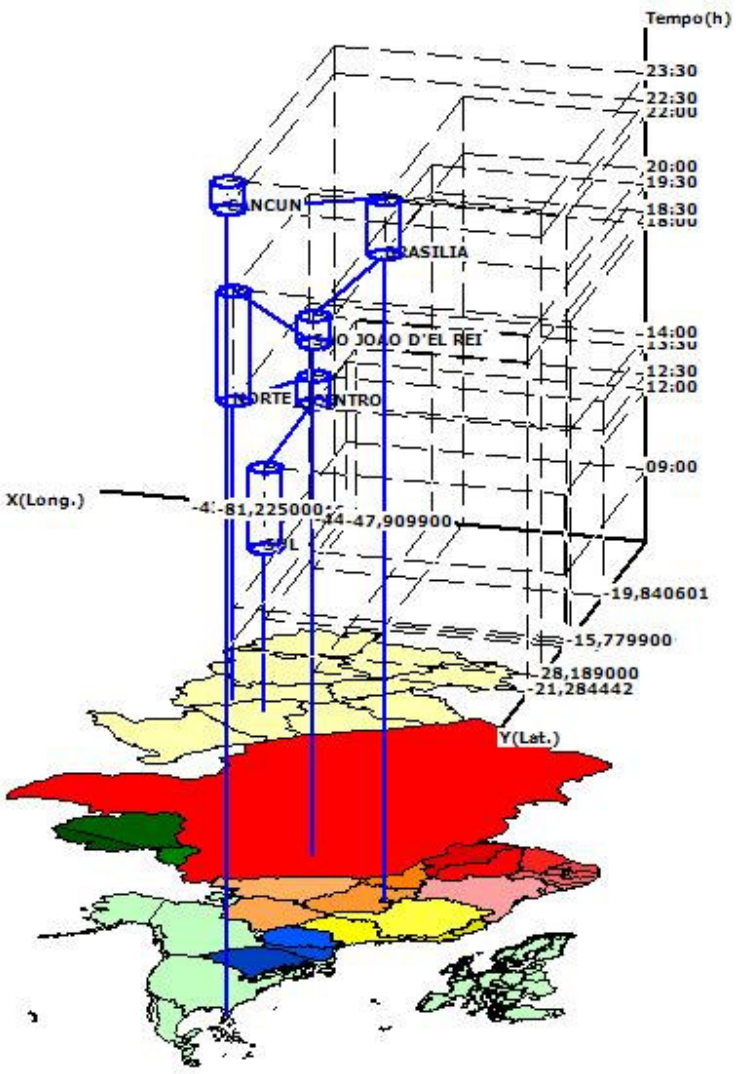


Figura 51-Diagrama com múltiplas camadas gerado pelo aplicativo. Fonte:(Abreu & Miranda Jr, 2007a)

O aplicativo foi utilizado para retratar o período de vida de alguns geólogos notáveis, conforme apresentado em (Abreu & Miranda Jr, 2007a). A Figura 52 mostra o diagrama têmporo-espacial da vida de Waldo Tobler.

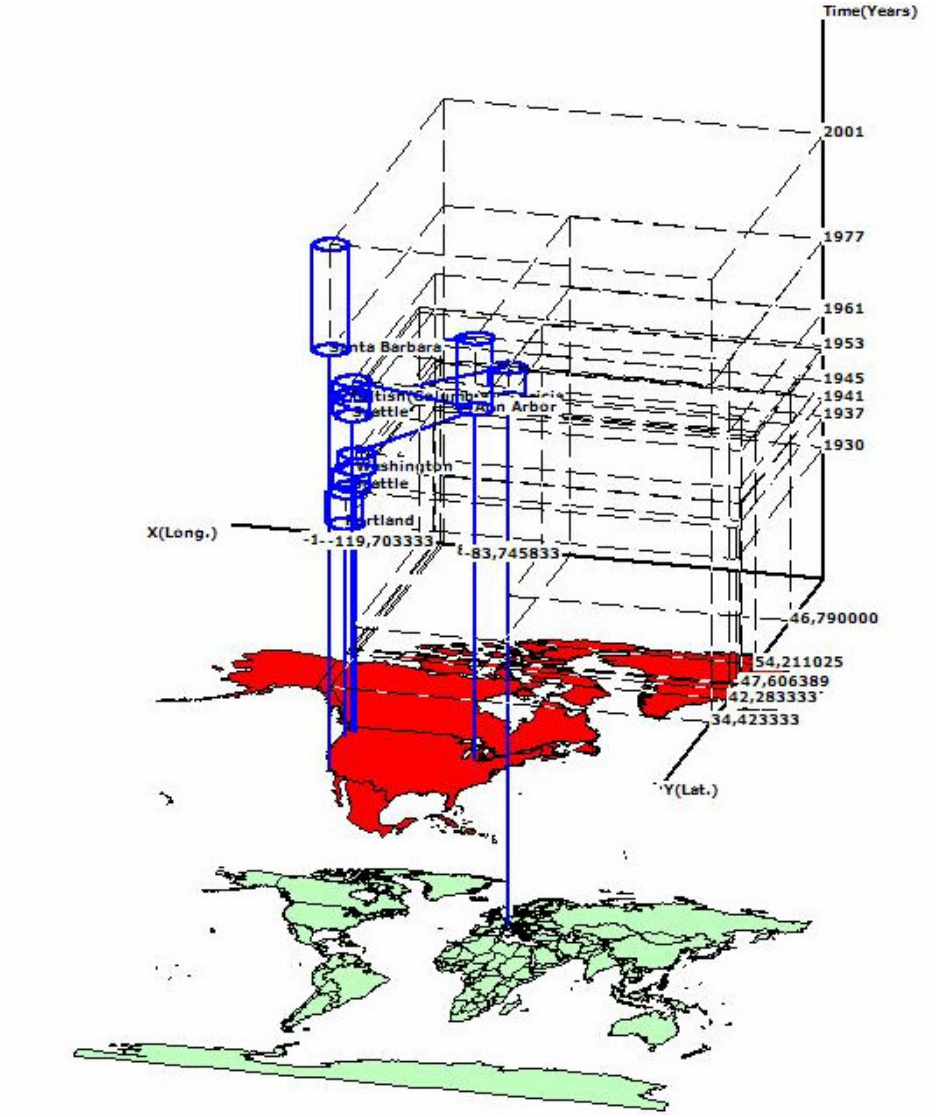


Figura 52-Diagrama têmporo-espacial da vida de Waldo Tobler.

4.2 - Aplicativo para Cálculo de Índices LISA e Autocorrelação

4.2.1 -Autocorrelação

A autocorrelação expressa quanto o valor observado de um atributo numa região é dependente dos valores deste mesmo atributo nas localizações vizinhas. O índice de Moram I é o mais utilizado para estimar a força da correlação entre valores observados em função da distância que os separam. Este índice é dado pela expressão:

$$I = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(z_i - \bar{z})(z_j - \bar{z})}{\sum_{i=1}^{n}(z_i - \bar{z})^2} \quad (1)$$

onde

z_i = o valor normalizado do atributo considerado na área i

$\bar{z}$ = é o valor médio do atributo na região sob estudo.

n = número de áreas

w_{ij} = peso da área i em relação à área j. Se forem vizinhas, seu valor pode ser 1, caso contrário, 0. Este peso pode ser também a distância entre as áreas.

Valores positivos (entre 0 e +1) do índice de Moran indicam uma correlação direta, valores negativos (entre -1 e 0) indicam correlação inversa.

Outro importante indicador de autocorrelação é o LISA (*Local Indicator of Spatial Association)* (Anselin, 1995), que é o equivalente local do índice de Moran (a soma de todos os índices locais de uma região é proporcional ao índice de Moran). Ele é obtido pela seguinte expressão:

$$I_i = z_i \sum_j^n w_{ij} z_j \qquad (2)$$

onde I_i é o índice LISA de correlação espacial local para cada área i na região considerada.

Para cada área, os índices LISA permitem avaliar sua similaridade em relação às áreas vizinhas e testar sua significância. Desta análise, quatro cenários podem emergir:

- Áreas com valores altos e vizinhos similares: *high-high*. São denominadas *hot spots.*
- Áreas com valores baixos e vizinhos similares: *low-low*. São denominadas *cold spots.*
- Áreas com valores altos e vizinhos com valores baixos: *high-low*. São denominadas *potential spatial outliers.*
- Áreas com valores baixos e vizinhos com valores altos: *low-high*. São também denominadas *potential spatial outliers.*
- Áreas com autocorrelação não significativa .

Para estimar a significância do índice LISA, utiliza-se o chamado *teste de pseud-significância:* para cada área, permutam-se os valores dos atributos nas diversas regiões ao seu redor e calcula-se um novo índice *I* para cada permutação. Obtém-se assim uma nova distribuição empírica do índice LISA para cada área da região. O valor real de cada índice é testado em relação à distribuição obtida pelas diversas situações, através do chamado *p-value,* que é definido como "a probabilidade de qualquer média da amostra ser mais extrema que a média da amostra $\overline{I}$ extraída para o teste". O *p-value* é calculado da seguinte forma:

Para cada área i tem-se I_i e a média $\overline{I}$ dos valores permutados. Obtém-se o *p-value* da seguinte forma:

Se $\overline{I} >= I_i$ então p-value = $2\ x\ P(Z >= Z_o)$

e $\overline{I} <= I_i$ então p-value = $2\ x\ P(Z <= Z_o)$

Onde P é a distribuição normal de probabilidades e $Z_o = (I_i - \overline{I})/\sigma$ (valor normalizado da variável).

Uma vez calculada a significância, pode-se exibir em um mapa as regiões que exibem um p-value menor que um determinado valor (alta significância). A Figura 53 exibe um exemplo deste mapa, denominado *mapa de clusters,* que mostra os *hot spots,* os *cold spots* e os *spatial outliers,* com *p-value <= 0,002.* A Figura 54 exibe o denominado *gráfico de dispersão*, cujos quadrantes correspondem aos *hot spots* (quadrante 1), os *cold spots*(quadrante 3) e os *spatial outliers* (quadrante 2 e quadrante 4).

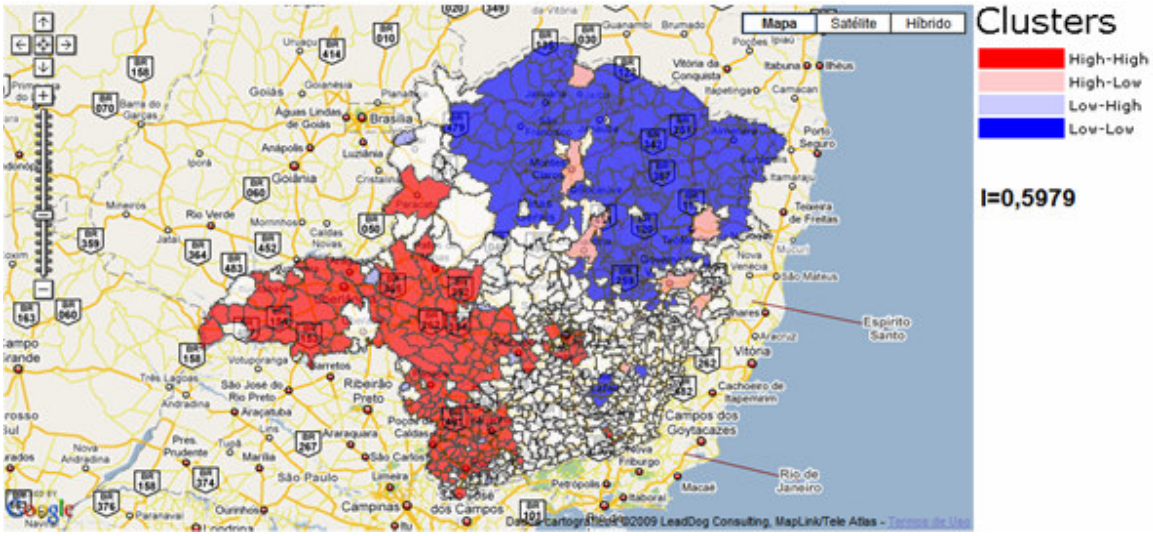


Figura 53- Mapa de clusters para Renda Per Capita em Municípios de Minas Gerais- Fonte: FJP

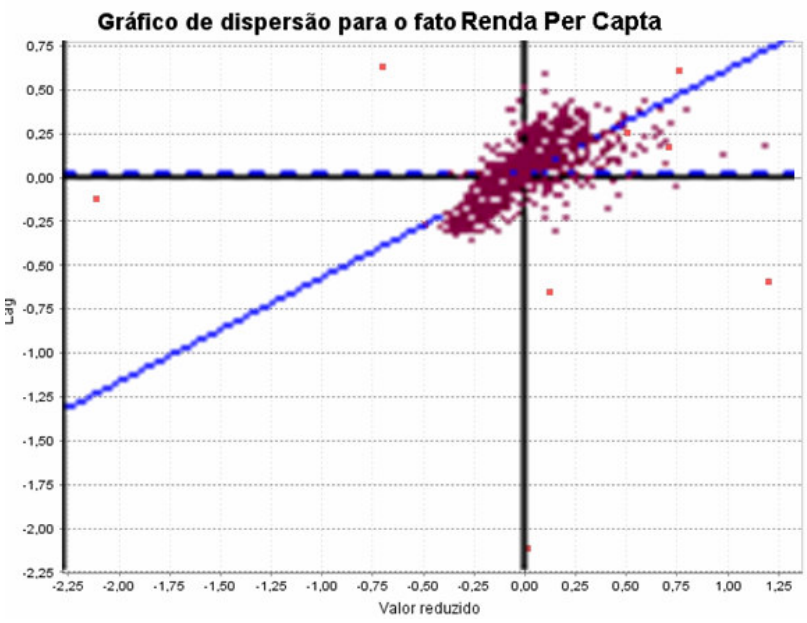


Figura 54 - Gráfico de dispersão para Renda Per Capita em Municípios de Minas Gerais- Fonte: FJP

O mapa de *clusters* exibido na Figura 53 para Renda Per Capita mostra claramente que as regiões de *hot spots* (*High-High*), correspondem realmente às regiões mais ricas do estado (região central, ao redor de Belo Horizonte e triângulo mineiro), e que as regiões *low-low*, como era de se esperar, correspondem ao norte-nordeste de minas, mais pobres. As regiões em branco não obtiveram nível de significância suficiente.

4.2.2 – A Aplicação

Foi realizada uma implementação da interface para os modelos LISA e de autocorrelação especificados na infraestrutura proposta. Os resultados são exibidos

por meio dos pacotes desenvolvidos para gráficos, mapas e relatórios: mapas de *clusters* e índices LISA, gráfico de dispersão e relatórios de indicadores LISA.

As Figuras 53 a 58 exibem os relatórios, gráficos e mapas (baseados no Google Maps) produzidos pela aplicação e exibidos via *web browser*. Em especial, o mapa de *clusters* exibe também o índice de Moran.

Indicadores Locais de Associação Espacial

21/02/2008 17:31:05

NOME REGIÃO	LISA
Barreiro	-0,0319
Centro-Sul	0,1309
Leste	-0,7155
Nordeste	-0,4441
Noroeste	-0,0778
Norte	0,1293
Oeste	0,2521
Pampulha	0,1215
Venda Nova	0,4631

Figura 55 – Relatório de Indicadores LISA produzido pela aplicação

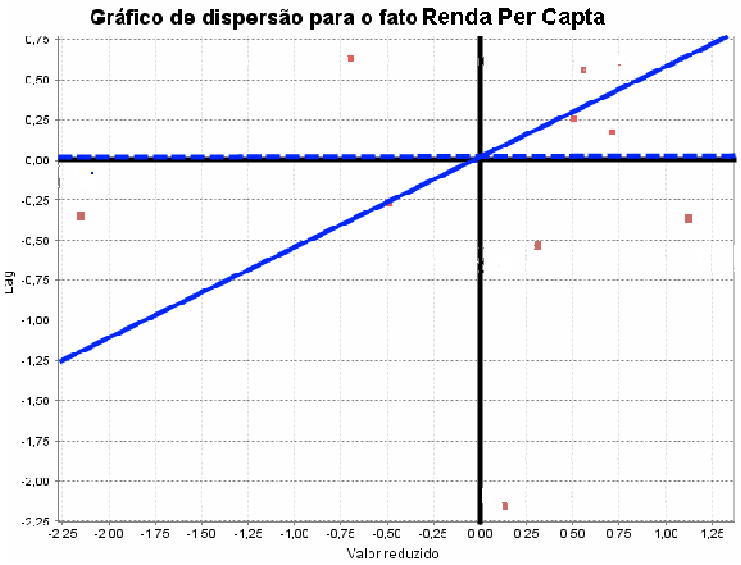


Figura 56 – Gráfico de dispersão produzido pela aplicação

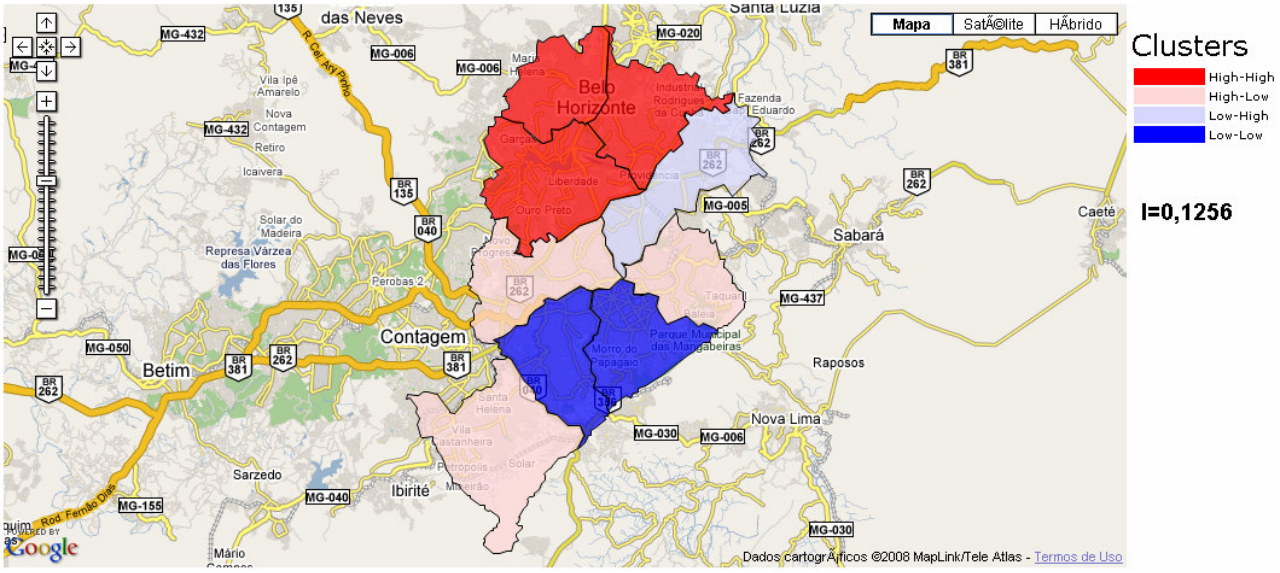


Figura 57 – Mapa de clusters produzido pela aplicação, sobre o Google Maps, da Região Metropolitana de Belo Horizonte

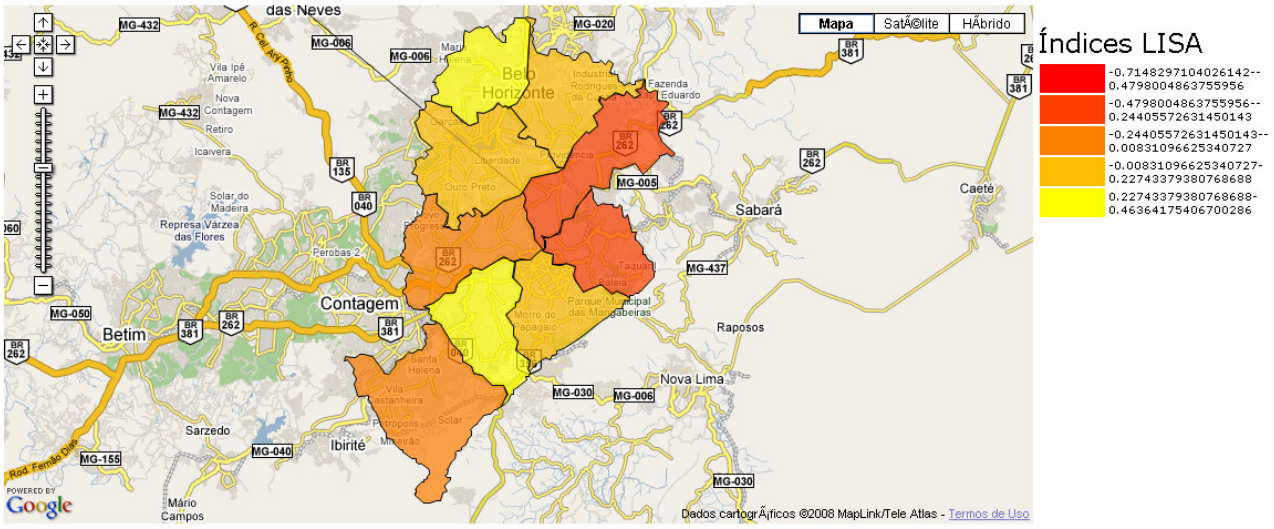


Figura 58 – Mapa de indicadores LISA produzido pela aplicação sobre o Google Maps, da Região Metropolitana de Belo Horizonte

A Figura 59 exibe a entrada de dados necessários à execução da aplicação:

- Tabela de fatos
- Fato a ser analisado
- Objeto geográfico em relação ao qual o resultado será exibido
- Seleção de mapas, gráficos e relatórios a serem exibidos.

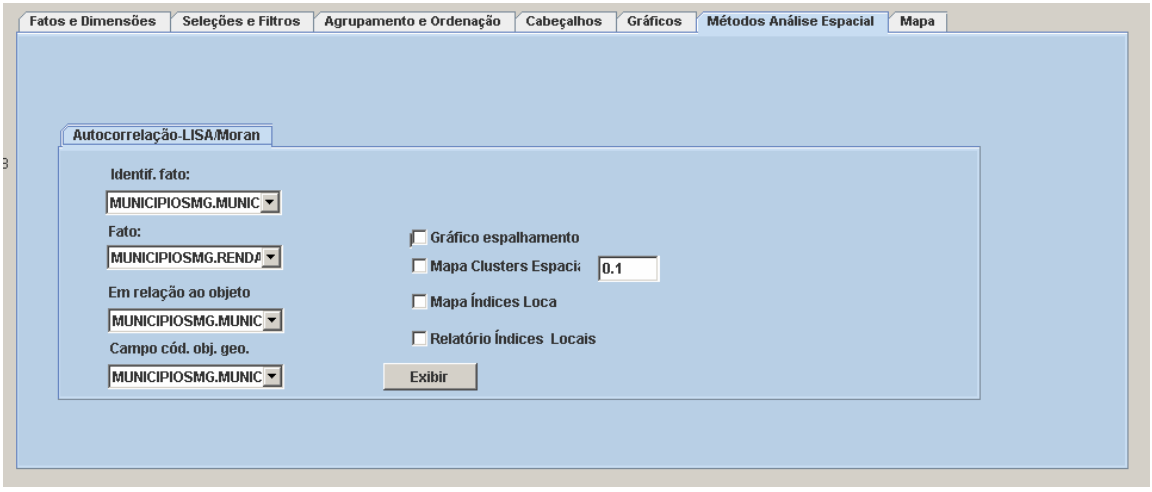


Figura 59 – Tela para entrada de dados necessários ao cálculo dos Índices LISA

4.3 –Aplicativo para Análise Espacial de Reticulados por meio da Geometria Táxi.

4.3.1 –A Geometria Táxi

As formas geométricas das malhas que compõem o mapa urbano de uma cidade são indicadores dos caminhos a serem percorridos em suas vias. Existem traçados assimétricos (malhas orgânicas), que se formaram naturalmente-como é o caso de Londres ou Recife- ou simétricos, compostos de linhas retas ortogonais (malhas racionais), que foram projetados – como é o caso de Manhatan, Curitiba ou Chicago (Abreu, 1982).

As Tabelas 13 e 14 exibem um levantamento estatístico das formas geométricas de cidades, considerando-se os seguintes tipos de malhas:

a)Radiais: as ruas se abrem em raios, como um leque, do centro para a periferia.

b)Reticulares: malha retangular, própria de cidades racionais planejadas

c)Em anéis: as ruas se assemelham a laços ou argolas.

d)Outras: não há forma definida

FORMA DAS PRINCIPAIS CIDADES AMERICANAS				
RETICULAR	**RADIAL**	**EM ANEL**	**OUTRAS**	**TOTAL**
61%	6%	7%	26%	100%

Tabela 13- Geometria das cidades americanas –Extraído de (Abreu, 1982)

FORMA DAS PRINCIPAIS CIDADES BRASILEIRAS				
RETICULAR	**RADIAL**	**EM ANEL**	**OUTRAS**	**TOTAL**
35%	20%	15%	30%	100%

Tabela 14- Geometria das cidades brasileiras - Extraído de (Abreu, 1982)

Nos casos americano e brasileiro, observa-se a predominância da geometria reticular, que é o objeto de interesse da geometria táxi, onde o movimento de pessoas e objetos se faz tal qual o movimento de um táxi pelas ruas e avenidas.

4.3.1.1 Propriedades

Conforme mostrado na Figura 60, uma pessoa ao deslocar-se em uma cidade de malha reticular de um ponto P até outro Q, separados por 5 quarteirões de 100 metros cada, pode fazê-lo de 10 maneiras distintas, pelo menor caminho táxi, percorrendo uma distância de 500 metros em quaisquer das 10 viagens. A distância em linha reta (dita euclidiana) entre os dois pontos será de 360 metros.

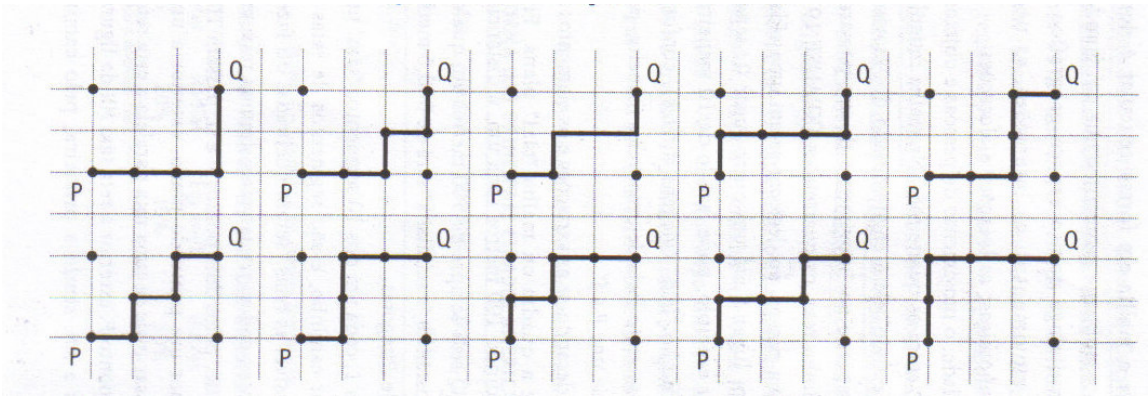


Figura 60- Distância entre dois pontos- Extraído de (Abreu, 1982)

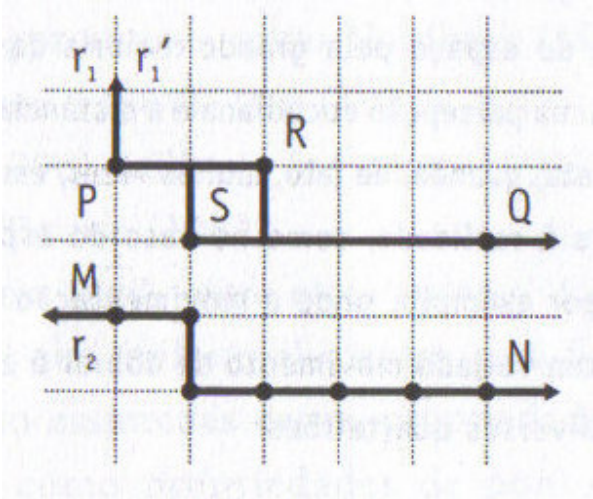


Figura 61- Retas na geometria táxi- Extraído de (Abreu,, 1982)

.

A geometria táxi tem as seguintes propriedades notáveis:

-As retas podem se quebrar ou fazer curvas.

-Por dois pontos podem passar mais de uma reta (como se pode ver na Figura 60).

-Dada uma reta táxi, por um ponto do plano exterior a ela pode passar mais de uma reta paralela à mesma (Figura 61).

Estas propriedades sugerem um modelo geométrico diferente do euclidiano, mais adequado a estudos e pesquisas em um espaço urbano reticulado, no qual a realidade é muito mais a de dobrar à direita e à esquerda nos quarteirões do que a de seguir uma linha reta entre dois pontos. Portanto, a geometria euclidiana presta-se mais ao mundo natural, ao passo que a táxi melhor se aplica ao mundo artificial das cidades (Barroso, 1982).

Esta geometria diferencia-se da euclidiana pela métrica utilizada, fato que acarreta significativas diferenciações.

4.3.1.2 -A métrica táxi

A geometria táxi também utiliza um sistema de coordenadas cartesianas. O matemático russo Minkowiski definiu a distância entre dois pontos A (x_1,y_1) e B (x_2,y_2) do plano como sendo (Figura 62):

$$D_M (A,B) = [| x_2-x_1|^r + |y_2-y_1|^r]^{1/r} \quad (1)$$

onde r é um inteiro positivo arbitrado.

Se r= 2 tem-se a métrica euclidiana já conhecida do R^2:

$$D_E (A,B) = [| x_2-x_1|^2 + |y_2-y_1|^2]^{1/2} \quad (2)$$

Se r= 1 tem-se a métrica táxi:

$$D_T (A,B) = | x_2-x_1| + |y_2-y_1| \quad (3)$$

A métrica táxi satisfaz às 4 condições formais que determinam se uma métrica d é realmente uma métrica em M, para quaisquer x,y,z pertencentes a M (Barroso, 1992):

(a) d(x,y) = 0 se x=y (nulidade)

(b) d(x,y) >0 se x <> y (positividade)

(c) d(x,y)=d(y,x) (simetria)

(d) d(x,y) <= d(x,z) + d(y,z) (desigualdade triangular)

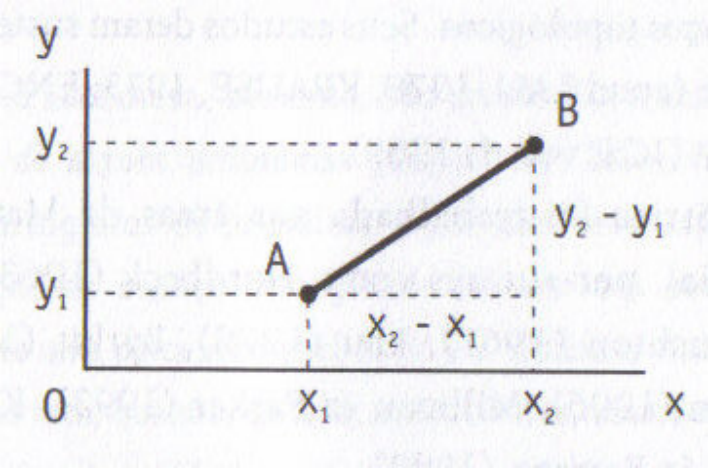


Figura 62- Sistema de Coordenadas Cartesianas- Extraído de (Abreu, 1982)

As métricas táxi e euclidiana diferem entre si por uma constante K (Figura 63).

$D_T\ (P,Q) = k\ D_E\ (P,Q)$ (4)

Da Figura 63, a inclinação da reta r é:

$|y_2-y_1|/|\ x_2-x_1| = m$ (5)

Após algumas manipulações algébricas chega-se a

$$k = 1 + |m| / (1 + m^2)^{1/2} \quad (6)$$

Então temos que:

$$D_T(P,Q) = (1 + |m| / (1 + m^2)^{1/2}) D_E(P,Q) \quad (7)$$

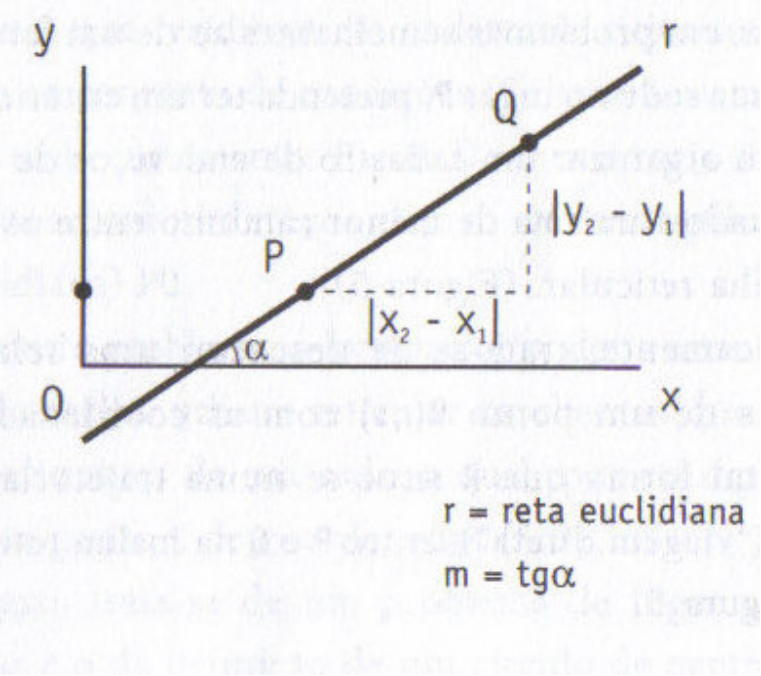


Figura 63- Equivalência entre as distâncias euclidiana e táxi- Extraído de (Abreu, 1982)

4.3.1.3 Menor caminho entre dois pontos

Segundo (Barroso, 1982), para determinar o menor caminho entre dois pontos P e Q em um reticulado, temos que encontrar a relação entre as coordenadas de um ponto R (r,s) com as coordenadas de P (x,y) e Q(x',y') de forma que R esteja numa trajetória de caminho mais curto entre P e Q (Figura 64). Podemos então escrever:

$$| x-x'| = | r-x'| + |x-r| = |r-x| + |r-x'| >= |r-x+r-x'| = |2r-x-x'|$$

$$| y-y'| = | s-y'| + |y-s| = |s-y| + |s-y'| >= |s-y+s-y'| = |2s-y-y'|$$

Ou seja, todos os caminhos cujos pontos satisfaçam às condições

$$|2r-x-x'| <= | x-x'| \quad (8)$$

$$|2s-y-y'| <= | y-y'| \quad (9)$$

são caminhos mínimos entre P e Q.

As duas inequações acima nos dizem, portanto que os caminhos mínimos estão dentro do retângulo táxi de diagonal euclidiana PQ.

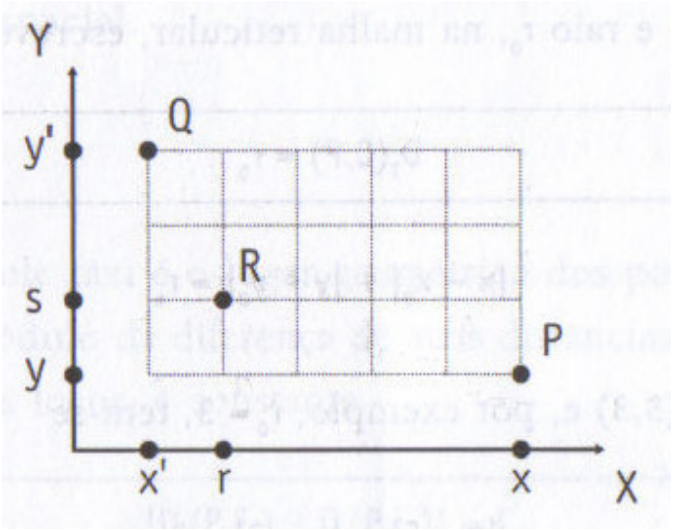


Figura 64 - Menor caminho entre dois pontos- Extraído de (Abreu, 1982)

Algoritmicamente temos (Figura 64):

Enquanto Q(x', y') não for alcançado tome o próximo ponto R (r,s) tal que (r = x e s > y) ou (r < x e s = y), onde x e y são as coordenadas do ponto anteriormente percorrido, partindo de P(x,y).

Este é um algoritmo dito *O(n)*, ou seja, o esforço para se obter o melhor caminho é proporcional a n, onde n é o número de pontos percorridos entre P e Q.

A título de curiosidade, temos o algoritmo A*, o mais popular para o cálculo do menor caminho entre dois pontos em uma malha qualquer:

Sendo: x= ponto atual;

f (x)= função de avaliação estática;

h(x)= esforço restante estimado mínimo para ir do ponto x até o ponto final;

g(x)=esforço acumulado para ir do ponto inicial até o ponto x;

1-Forme uma fila constituída por uma via contendo apenas o nó-raíz.

2-Até que a fila esteja vazia,ou que ponto meta seja encontrado, FAÇA:

2.1-Expanda a via que está na cabeça da fila em um nível;

2.2-Para cada ponto x resultante da expansão, calcule f(x)=h(x)+g(x);

2.3-Insira as novas vias resultantes da expansão para cada ponto x na fila;

2.4-Ordene a fila, a via com menor valor de f(x) na cabeça;

2.5-Remova da fila todas as vias que chegam a um nó em comum, exceto a que chega nele com valor mínimo de f(x);

2.6-Se a via que está na cabeça da fila atingir o ponto-meta,não faça nada, senão

volte a 2.

3-Se o ponto meta foi alcançado, proclame sucesso, senão, fracasso.

O esforço para computar este algoritmo é notoriamente maior *(O(n^2))*, pois temos que computar caminhos acumulados e restantes, além de ordenar os caminhos acumulados.

Assim sendo, concluímos que na situação em que a geometria da cidade for predominantemente reticular, a geometria táxi produz um resultado mais rápido para o cálculo do menor caminho entre dois pontos.

4.3.1.4 -Lugares Geométricos

Um lugar geométrico é um conjunto de pontos que satisfazem a determinada propriedade. A seguir abordaremos alguns lugares geométricos clássicos, sob a ótica da geometria táxi.

O círculo

O círculo é o lugar geométrico dos pontos P(x,y) eqüidistantes de um centro C. Assim sendo, a equação de um círculo de centro C (x_0,y_0), raio r_0 e passando pelo ponto genérico P(x,y) é:

$D_T\,(C,P) = r_0$

ou seja

$|x - x_0| + |y - y_0| = r_0$ (10)

Na Figura 65 temos um círculo de raio 3 e centro em C(5,3).

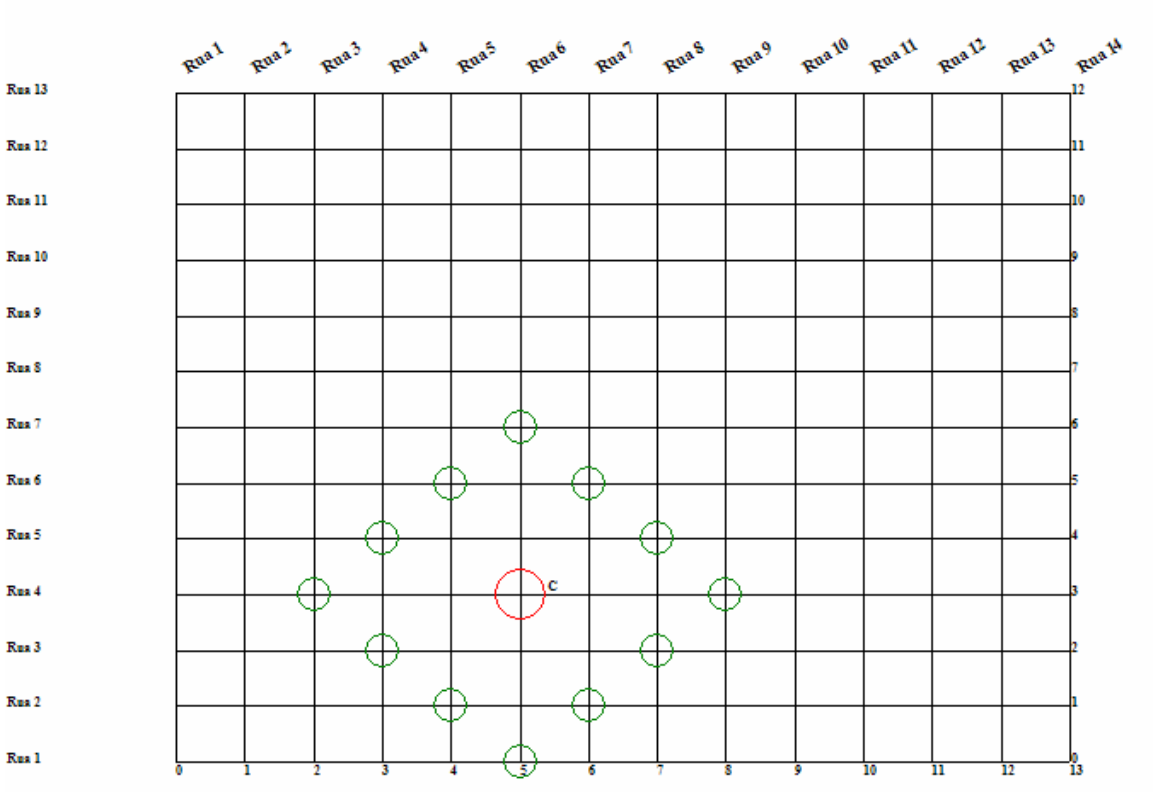


Figura 65 - Círculo táxi de centro C (círculo vermelho). Círculos verdes denotam o lugar geométrico.

A hipérbole

O hipérbole é o lugar geométrico dos pontos P(x,y) tal que o módulo da diferença de suas distâncias a dois pontos fixos, chamados *focos,* é constante.

$|D_T(P,F_1) - D_T(P,F_2)| = C$ onde F_1 e F_2 são os focos e C é uma constante.

Na Figura 66 abaixo, temos a hipérbole de focos F1(1,4) e F2(11,4) e C = 4

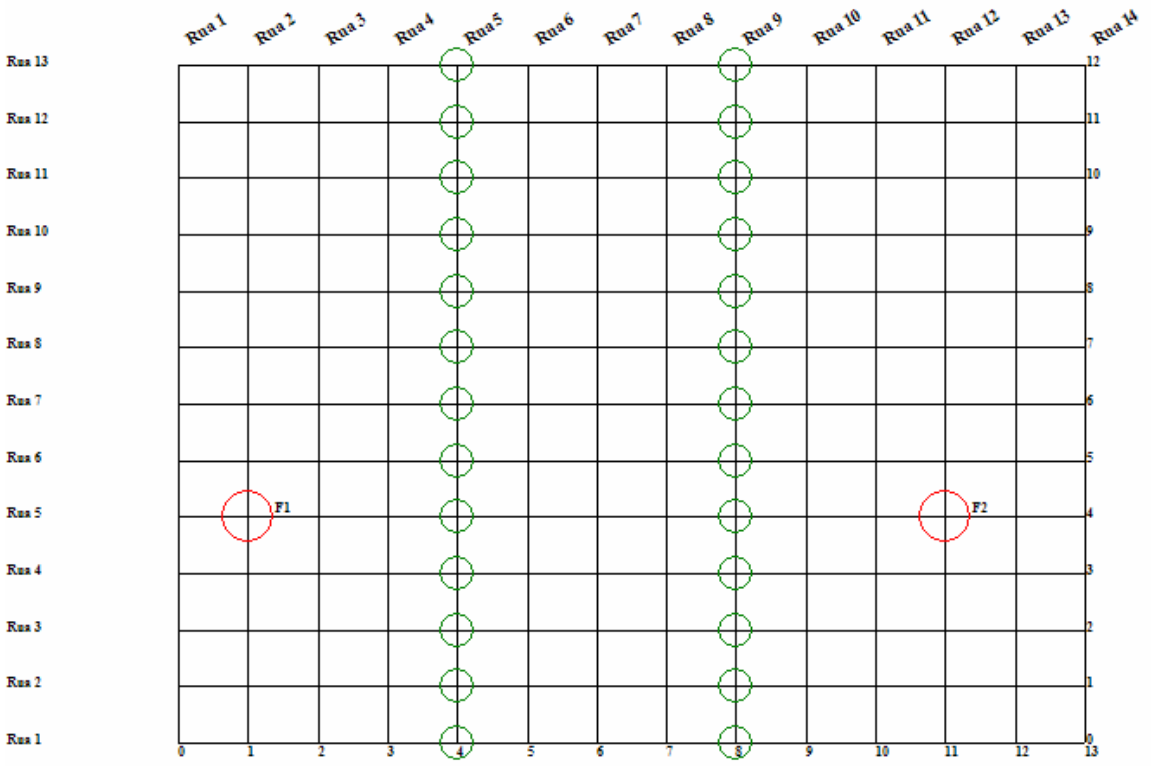


Figura 66- A hipérbole táxi de focos F1 e F2 (círculos vermelhos). Círculos verdes denotam o lugar geométrico.

Na Figura 47, temos uma hipérbole de focos F1(5,4) e F2(10,9) e C = 2

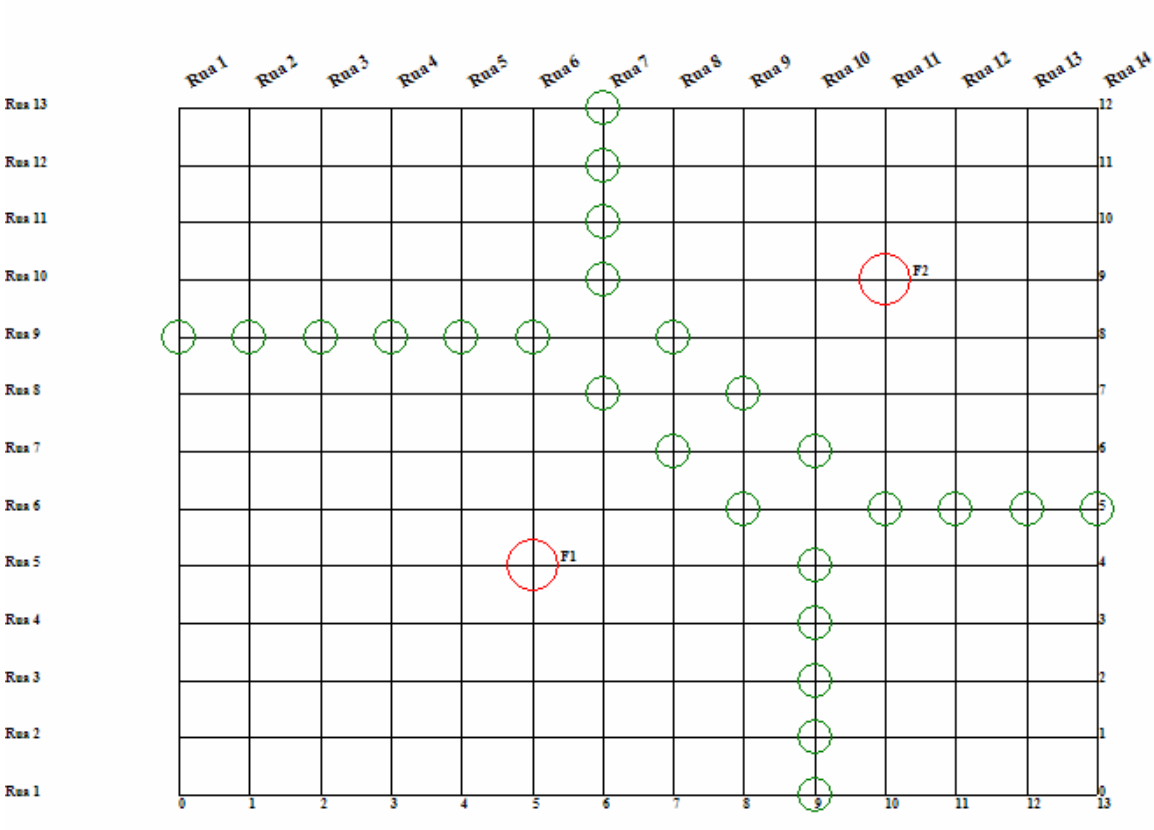


Figura 67 – A hipérbole táxi de focos F1 e F2 (círculos vermelhos). Círculos verdes denotam o lugar geométrico.

Na Figura 68, temos uma hipérbole de focos F1(5,3) e F2(10,6) e C = 2

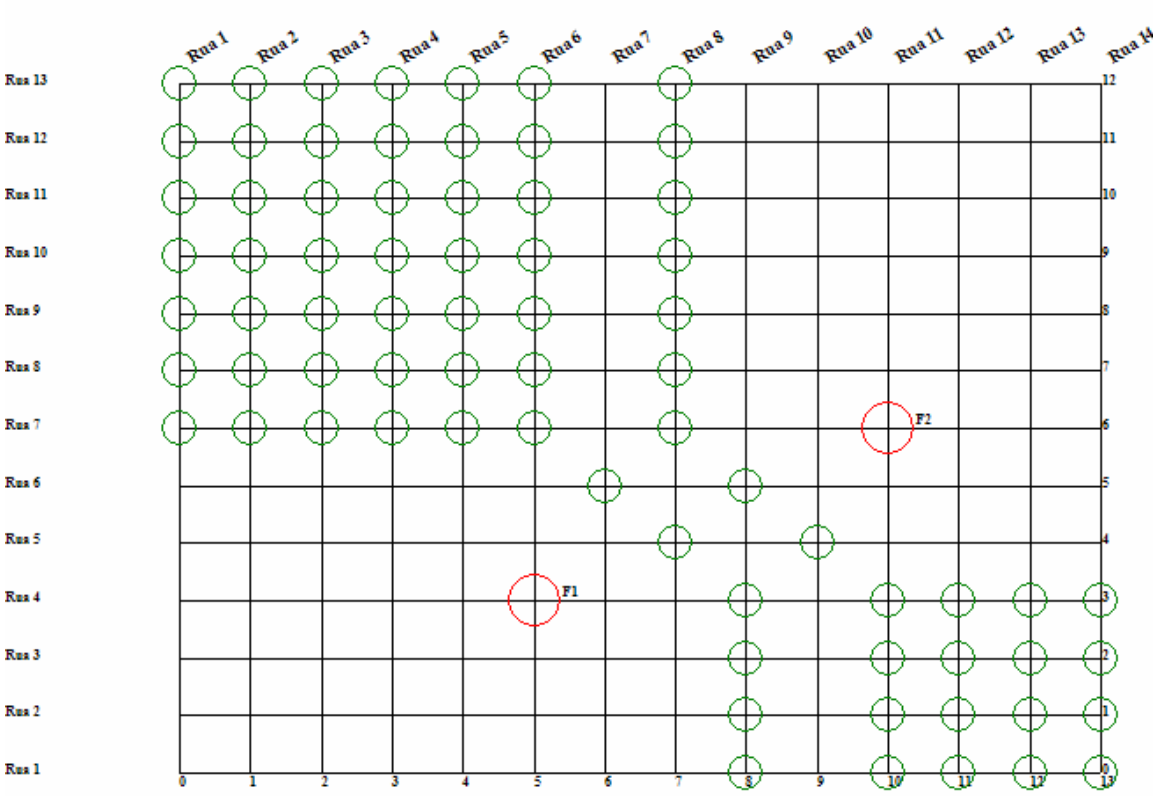


Figura 68 – A hipérbole táxi de focos F1 e F2 (círculos vermelhos). Círculos verdes denotam o lugar geométrico.

A elipse

O elipse é o lugar geométrico dos pontos P(x,y) tal que a soma de suas distâncias a dois pontos fixos, chamados *focos,* é constante.

$|D_T(P,F_1) + D_T(P,F_2)| = C$ onde F_1 e F_2 são os focos e C é uma constante.

Na Figura 69 abaixo, temos a elipse de focos F1(3,4) e F2(7,7) e C = 9

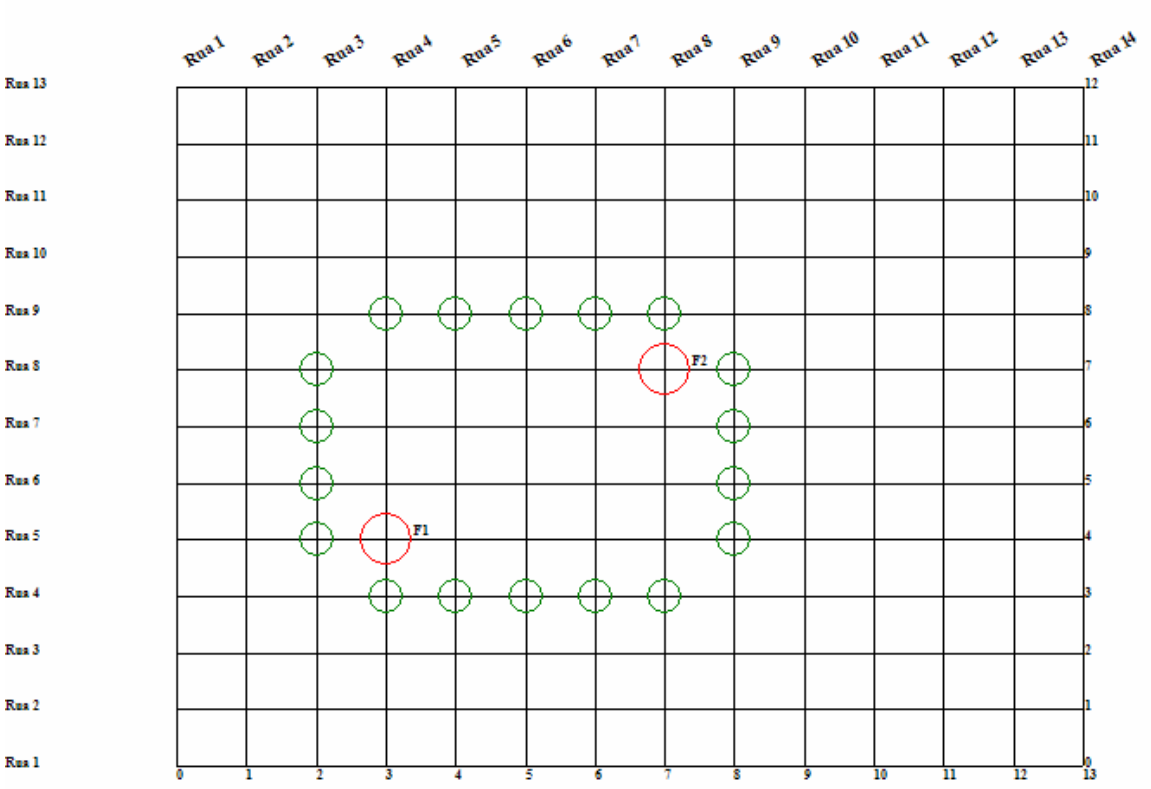


Figura 69 – A elipse táxi de focos F1 e F2 (círculos vermelhos). Círculos verdes denotam o lugar geométrico.

A parábola

Dado um ponto fixo F, o foco, e uma reta euclidiana r (a *diretriz)*, a parábola táxi é o lugar geométrico dos pontos P(x,y) tal que:

$D_T\,(P,F) = D_T\,(P,r)$

Observe que D_T (P,r) = min D_T (P,Q), sendo Q um ponto táxi da reta.

Na Figura 70 temos a parábola com foco em F(0,0) e de diretriz y=2x-12.

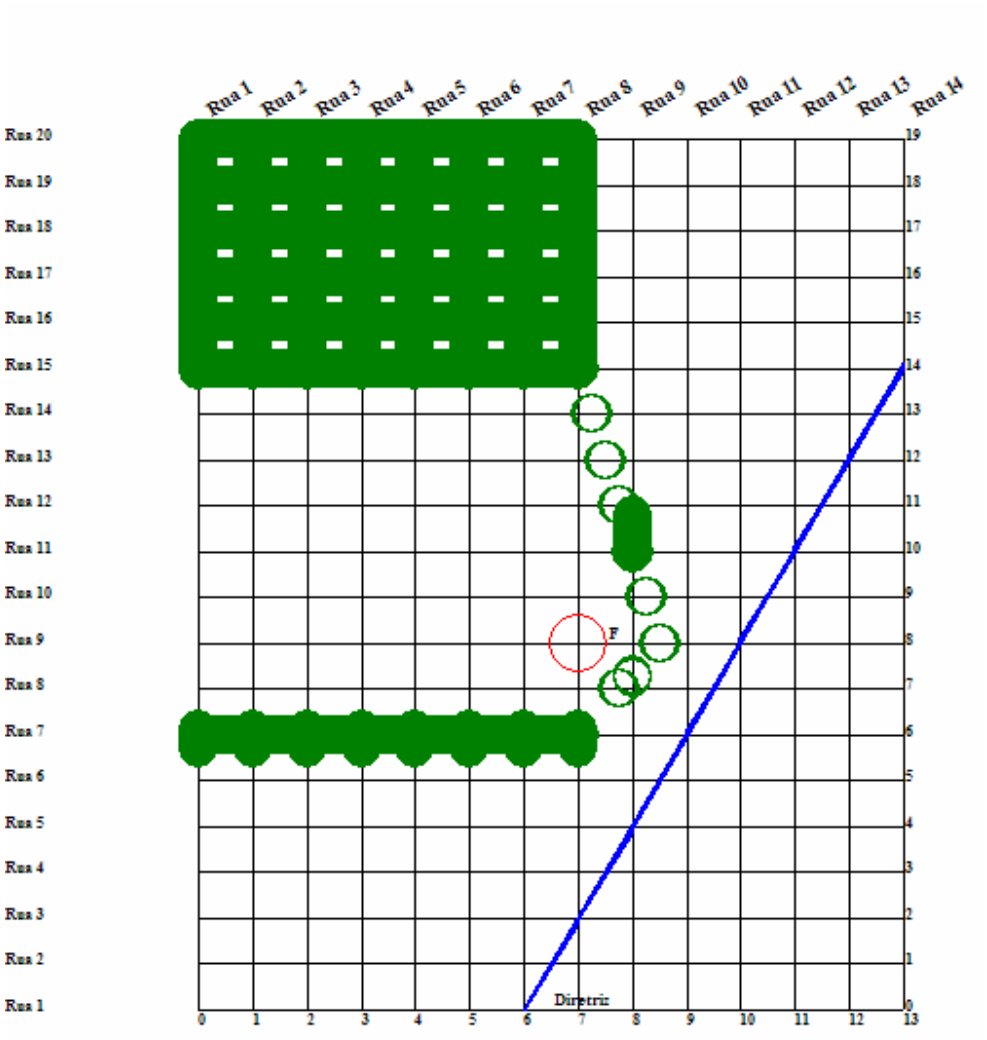


Figura 70- A parábola táxi de foco F (círculo vermelho). Círculos verdes superpostos denotam o lugar geométrico.

4.3.2 – O Aplicativo

Esta aplicação implementa a interface especificada para os modelos LISA e de autocorrelação.

4.3.2.1 -Entrada de dados topológicos da região a ser estudada

A topologia da região a ser estudada é armazenada através da estrutura reticular de vias ortogonais , da distância entre estas, de pontos de referência e de retas presentes na mesma. A Figura 71 abaixo exibe o formulário de entrada de tais dados.

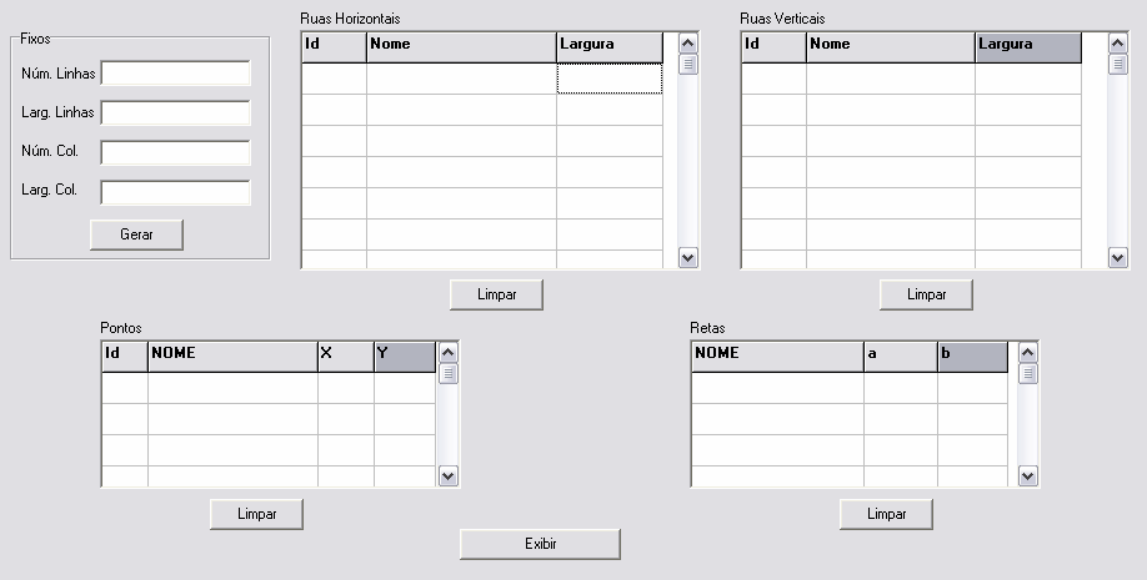


Figura 71 – Formulário para entrada de dados de topologia da região

As grades *Ruas Horizontais* e *Ruas Verticais* armazenam os dados de vias (Nome e Largura). A grade *Pontos* armazena pontos de referência (nome e coordenadas *x*

e *y).* A grade *Retas* armazena dados de retas (Nome, coeficiente angular *a* e coeficiente linear *b).* O quadro *Fixos* no alto à esquerda permite a geração automática das ruas do reticulado a partir do número de linhas e colunas ("Núm. Linhas" e "Núm. Colunas") bem com as dimensões dos quarteirões ("Larg. Linhas e "Larg. Col."). Pressionando-se o botão "Gerar", os quadros "Ruas Horizontais" e "Ruas Verticais" são preenchidos automaticamente, podendo posteriormente ser modificados.

4.3.2.2 -Entrada de dados de equações do modelo

Podem ser armazenadas quaisquer expressões regulares, com controle de precedência de operações passível de ser realizado por meio de parêntesis. Tais expressões têm o formato geral:

Expressão=[(](Equação)[),and, or,(] [Expressão]

Equação tem o formato geral:

Equação= [(](Membro1 [=, >,<,<=,>=, <>] Membro 2) [),and, or,(] [Equação]

Membro tem o formato geral:

Membro = [(] (Termo) [),+,-,,/,(] [Membro]*

Termo tem o formato geral:

Termo = [(] (Multiplicador [D_T ([Ponto,reta],[Ponto, reta])])*

onde Multiplicador é um número real e *D_T ([Ponto,reta],[Ponto, reta])* é uma função que retorna a distância táxi entre geometrias ponto e reta.

A Figura 72 exibe o formulário para entrada das equações do modelo a ser estudado. As grades 1º membro e 2º membro armazenam os termos de cada membro da equação relacionados por meio de um operador relacional(*=, >,<,<=,>=, <>*), selecionado no *combo-box* "Operador Relacional Entre Membros". A montagem dos termos de cada membro é feita pela digitação de um multiplicador (campo "Multip.", o *default* é 1), pela escolha de dois pontos entre os quais se deseja a distância (*combo-boxes "*Ponto 1" e "Ponto 2") e de um operador(+,-,*,/) que relaciona dois termos, se houver. Definido o termo, o botão "V" o insere na grade abaixo. Uma vez criada uma equação, ela deve ser armazenada. Isto é feito por meio do botão "Incluir Equação", que insere a equação na grade "Equações Incluídas" localizada na parte inferior da tela. O *combo-box* "Conexão com Próxima Equação" contém conectores lógicos (*or e and*) que relaciom equações entre si. Cada equação pode ser alterada ou excluída por meio dos respectivos botões "Alterar Equação" e "Excluir Equação". O botão *Exibir Resultado* exibe o sistema de coordenadas ortogonais contendo a topologia armazenada e o resultado da avaliação do conjunto de equações, conforme mostrado nos exemplos apresentados no próximo item.

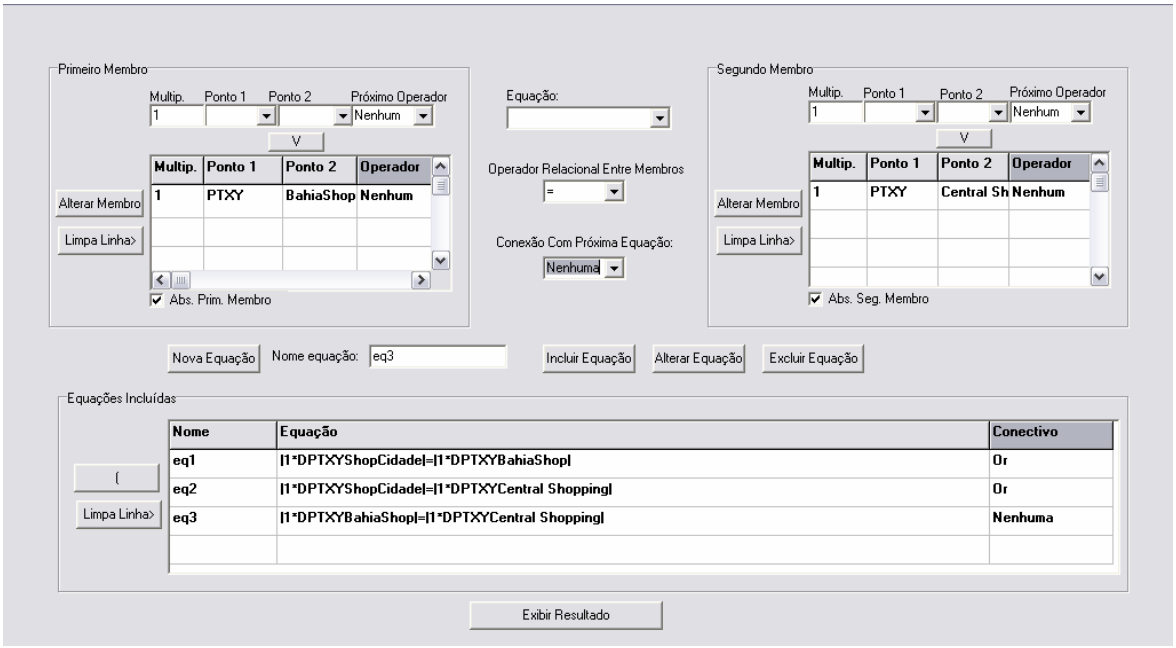


Figura 72- Formulário para registro de equações.

4.3.2.3 Exemplos de Aplicação

A seguir apresentamos alguns exemplos de utilização da geometria táxi para resolver alguns problemas de análise espacial. Utilizamos a região central de Belo Horizonte, que possui uma geometria aproximadamente reticular, com alguns ajustes (desprezando-se determinados objetos fora do padrão reticular) para adequá-la aos requisitos da geometria táxi. Os exemplos foram implementados utilizando-se o aplicativo desenvolvido nesta tese.

Consideramos quarteirões quadrados com 140 metros de lado. As avenidas estão representadas por retas diagonais. Consideramos três pontos de referência para análise: o Shopping Cidade (SC), o Central Shopping (CS) e o Bahia Shopping (BS). A Figura 73 exibe a região considerada.

Os resultados foram obtidos com um software desenvolvido para combinar equações e inequações da geometria táxi, relacionadas entre si por meio de operadores *or* ou *and.* As regiões/pontos-solução são demarcadas com círculos verdes.

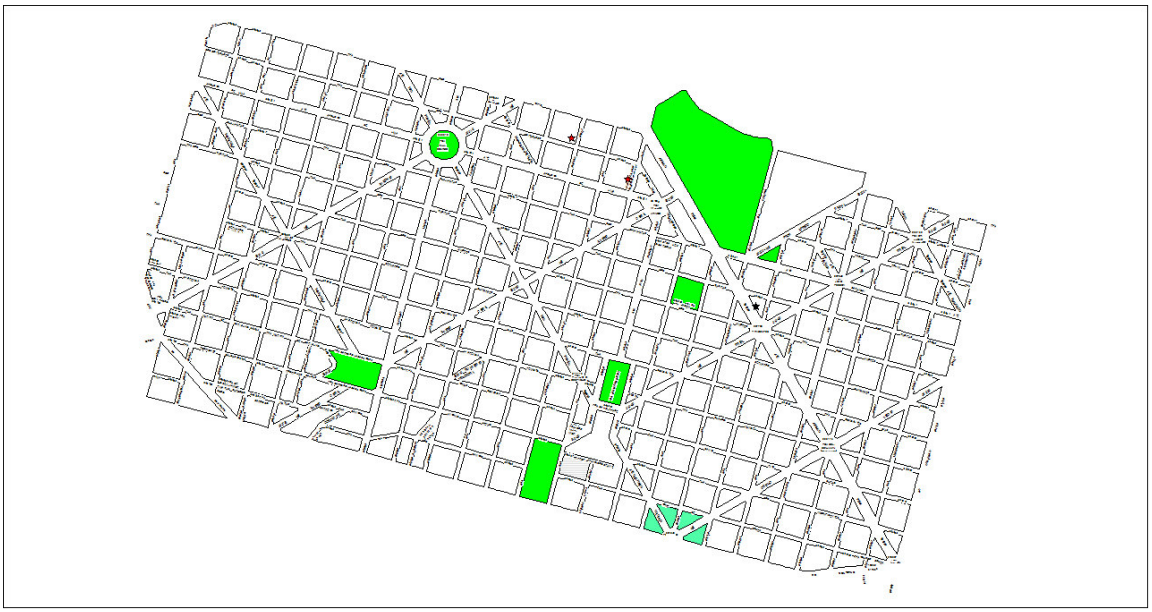


Figura 73 -Região Central de Belo Horizonte. Em verde visualizam-se parques e praças.

Exemplo 1-Determinar uma região possível de localização de um fornecedor, de forma que ela não esteja a mais de 300 m do Shopping Cidade e que razão entre as distâncias ao Shopping Cidade e ao Central Shopping esteja na razão inversa dos volumes de venda dos shoppings.

Consideramos o volume de vendas do Shopping Cidade como sendo o triplo do volume de vendas do Central Shopping.

O Modelo de equações e inequações que fornece a solução é o seguinte:

$D_T\ (P,SC) = D_T\ (P,CS)/3$

and

Dist(P,SC) <= 300

A Figura 74 a seguir exibe a região-solução.

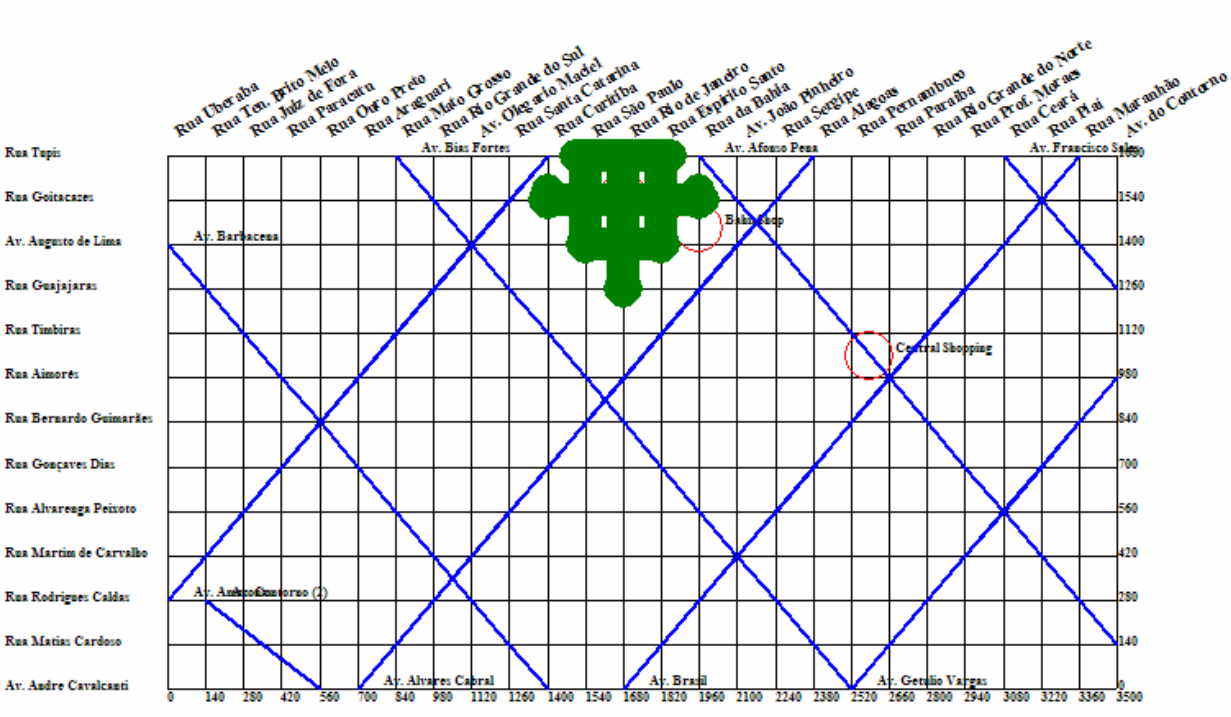


Figura 74-Pontos (círculos vermelhos) correspondentes aos shopping centers e regiões resultantes (círculos verdes superpostos) para o exemplo 1.

Exemplo 2- Encontrar limites entre os três shoppings de forma que cada consumidor dentro de tais limites esteja mais próximo de um dos três shoppings.

Neste exemplo, vamos definir 3 seções, correspondentes aos pontos eqüidistantes a cada par de shoppings. As equações que representam tais seções são:

$D_T (P,SC) = D_T (P,CS)$

or

$D_T (P,SC) = D_T (P,BS)$

or

$D_T (P,CS) = D_T (P,BS)$

A Figura 75 a seguir exibe o resultado da aplicação das equações acima. As linhas vermelha e verde delimitam as regiões de pontos mais próximos aos *shoppings,* quais sejam, mais próximos ao Shopping Cidade (esquerda da linha vermelha), mais próximos ao Bahia Shopping (entre as linhas vermelha e verde) e mais próximos ao Central Shopping (direita da linha verde).

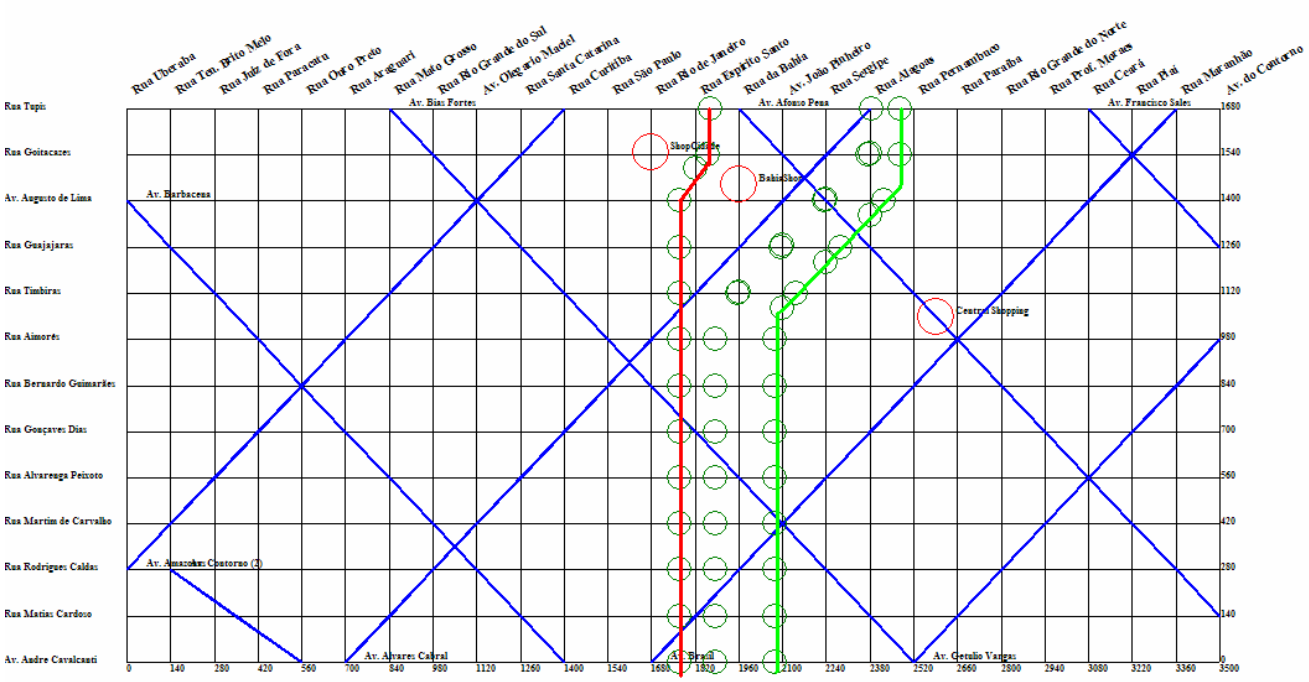


Figura 75- Pontos (círculos vermelhos) correspondentes aos shopping centers e regiões resultantes (círculos verdes com linhas vermelha e verde e a eles superpostas) para o exemplo 2.

Exemplo 3-Uma loja possui sua matriz no Shopping Cidade e filial no Central Shopping. Ela deseja instalar um depósito de forma que a distância para sair da matriz, passar pelo depósito e chegar à filial não consuma mais que 2000 m e que não esteja a mais que 1000 m da matriz.

Modelo de equações/inequações para o problema:

$D_T(P,SC) + D_T(P,CS) <= 2000$

and

$D_T(P,SC) <= 1000$

A Figura 76 exibe a região-solução.

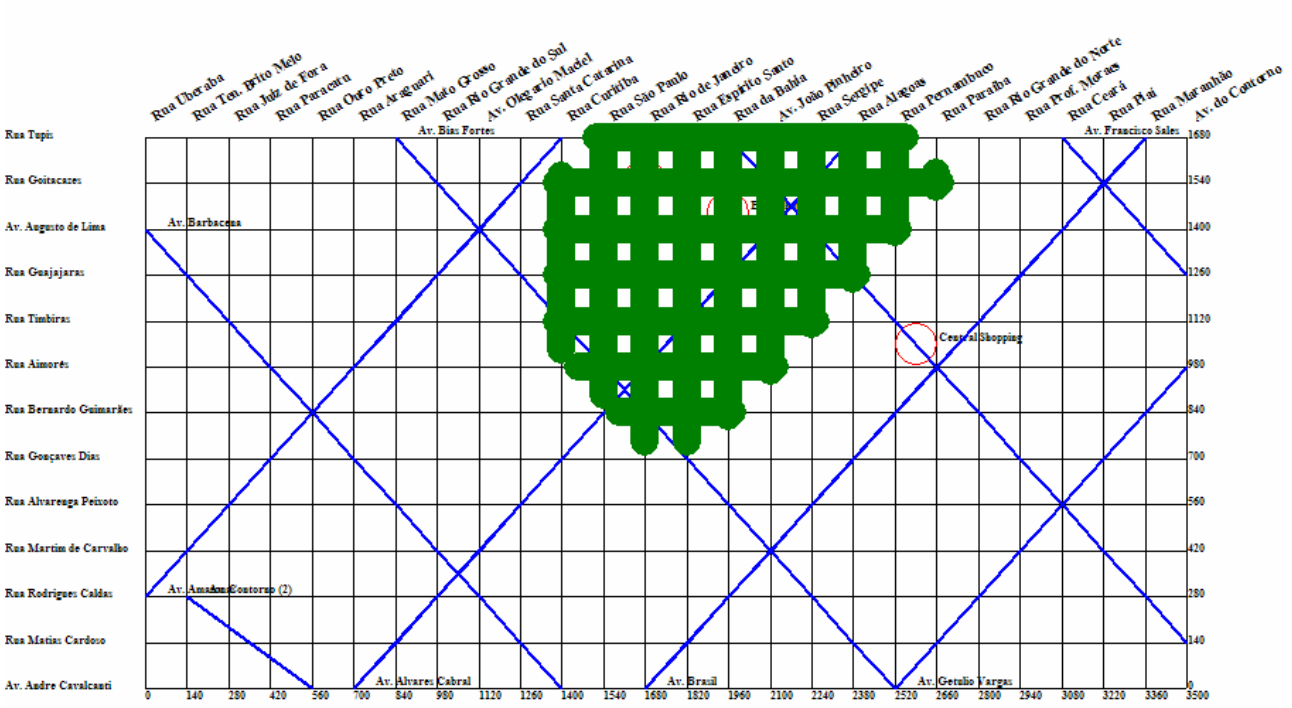


Figura 76- Pontos (círculos vermelhos) correspondentes aos shopping centers e regiões resultantes (círculos verdes superpostos) para o exemplo 3.

Exemplo 4-Uma loja com matriz no Shopping Cidade deseja escolher um ponto para instalar uma filial na Avenida Amazonas que não esteja a mais de 1600 metros da matriz.

Modelo de equações/inequações para o problema:

Sendo rAA a reta correspondente à Av. Amazonas

D_T *(P,rAA) = 0* (Pontos sobre a Av. Amazonas)

and

D_T *(P,SC) <= 1600*

A Figura 77 exibe a região-solução.

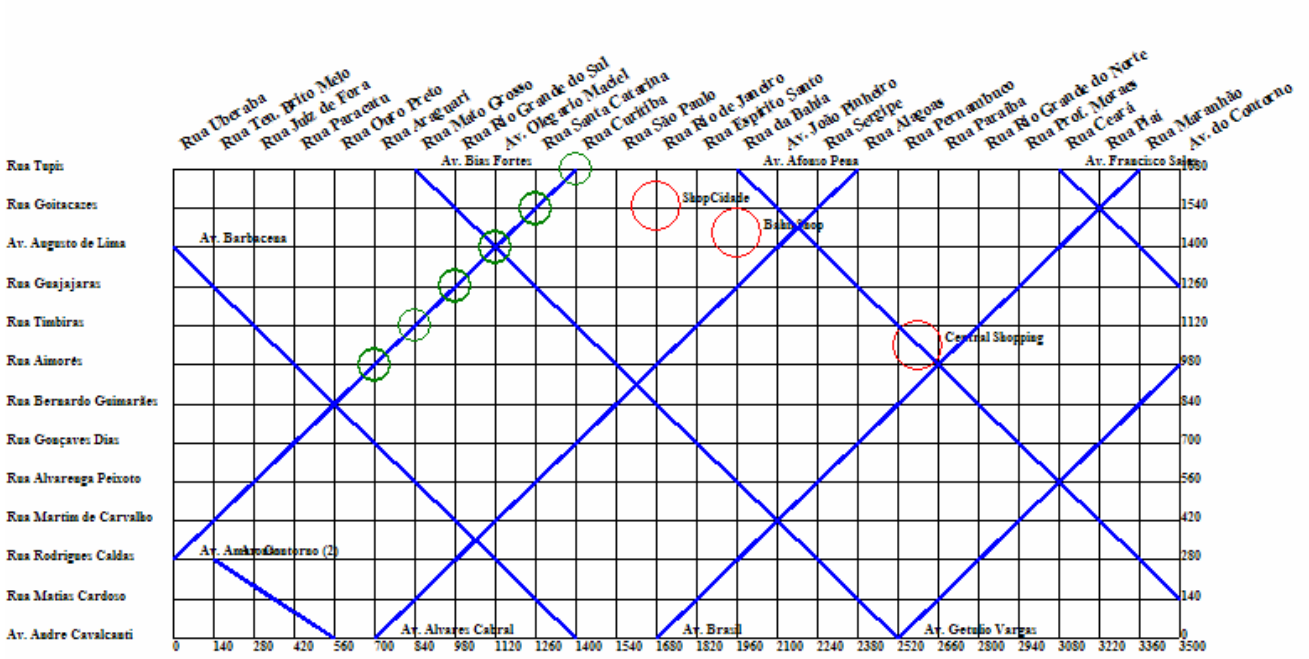


Figura 77- Pontos (círculos vermelhos) correspondentes aos shopping centers e regiões resultantes (círculos verdes) para o exemplo 4.

Exemplo 5- Uma loja do Shopping Cidade deseja definir uma região para implantação de uma filial que seja eqüidistante dela e da Av. Amazonas, mas que esteja a menos de 700 m do Shopping Cidade.

Modelo de equações/inequações para o problema:

Sendo rAA a reta correspondente à Av. Amazonas

D_T *(P,rAA) =* D_T *(P,SC) and*

D_T *(P,SC) <= 700*

A Figura 78 exibe a região-solução.

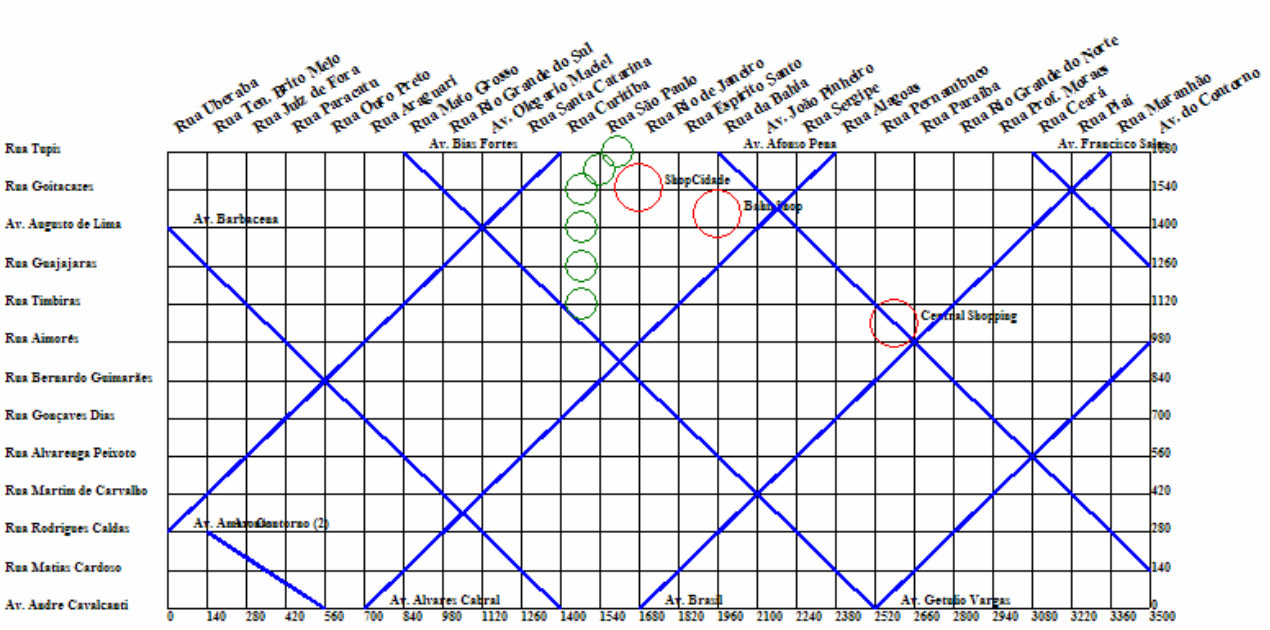


Figura 78- Pontos (círculos vermelhos) correspondentes aos shopping centers e regiões resultantes (círculos verdes) para o exemplo 5.

Exemplo 6-Deseja-se implantar um novo Shopping que esteja a mais de 2000 m dos três Shoppings.

Modelo de equações/inequações para o problema:

D_T *(P,SC) >= 2000*

And

D_T *(P,BS) >= 2000*

And

D_T *(P,CS) >= 2000*

A Figura 79 exibe a região-solução.

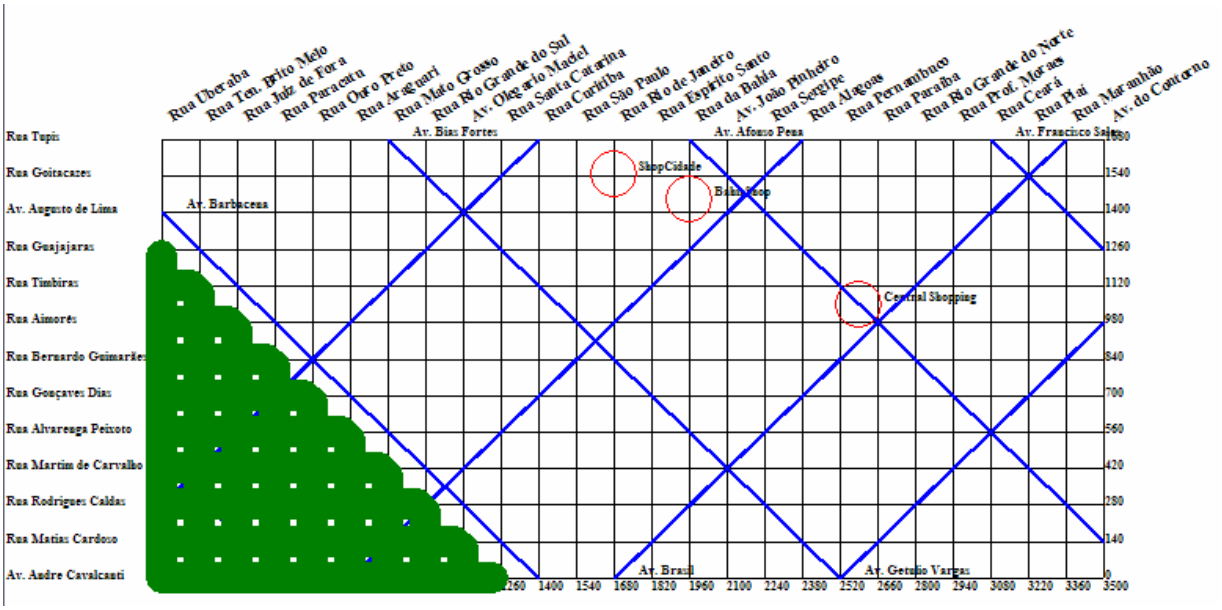


Figura 79- Pontos (círculos vermelhos) correspondentes aos shopping centers e regiões resultantes (círculos verdes superpostos)para o exemplo 6.

Exemplo 7-Idem anterior, mas eqüidistante das avenidas Amazonas e Álvares Cabral.

Modelo de equações/inequações para o problema:

Sendo rAA a reta correspondente à Av. Amazonas e rAC a reta correspondente à Av. Álvares Cabral.

$D_T (P,SC) >= 2000$

And

$D_T (P,BS) >= 2000$

And

$D_T (P,rAA) = D_T (P,rAC)$

A Figura 80 exibe a região-solução.

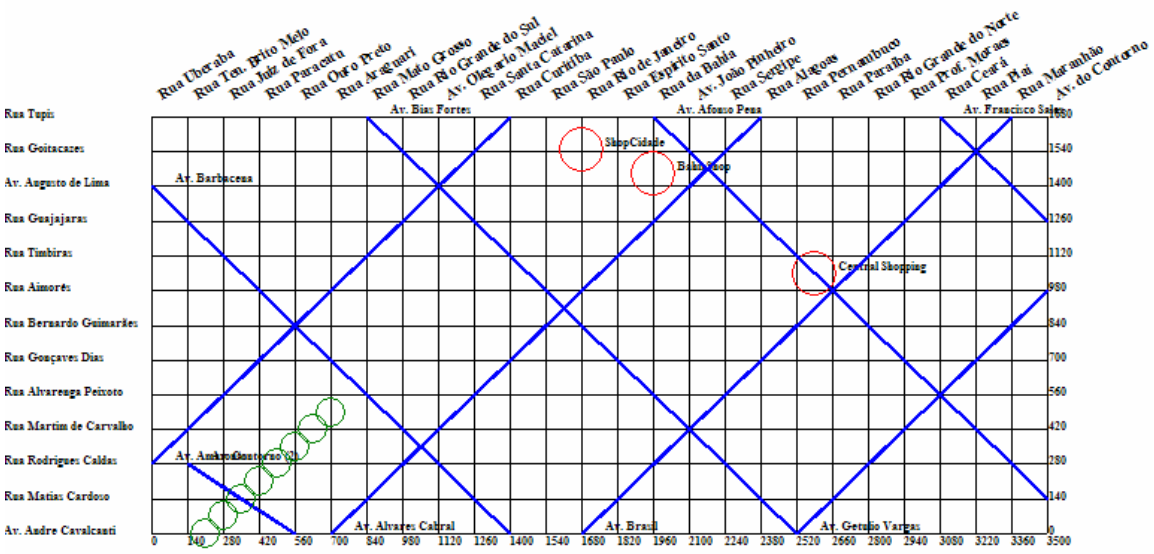


Figura 80 - Pontos (círculos vermelhos) correspondentes aos shopping centers e regiões resultantes (círculos verdes) para o exemplo 7.

Exemplo 8-Uma loja com matriz no Shopping Cidade e filial no Central Shopping deseja implantar uma segunda filial que esteja 400 m mais próxima da matriz do que da filial.

Modelo equações/inequações para o problema:

$D_T(P,CS) - D_T(P,CS) = 400$

A Figura 81 exibe a região-solução.

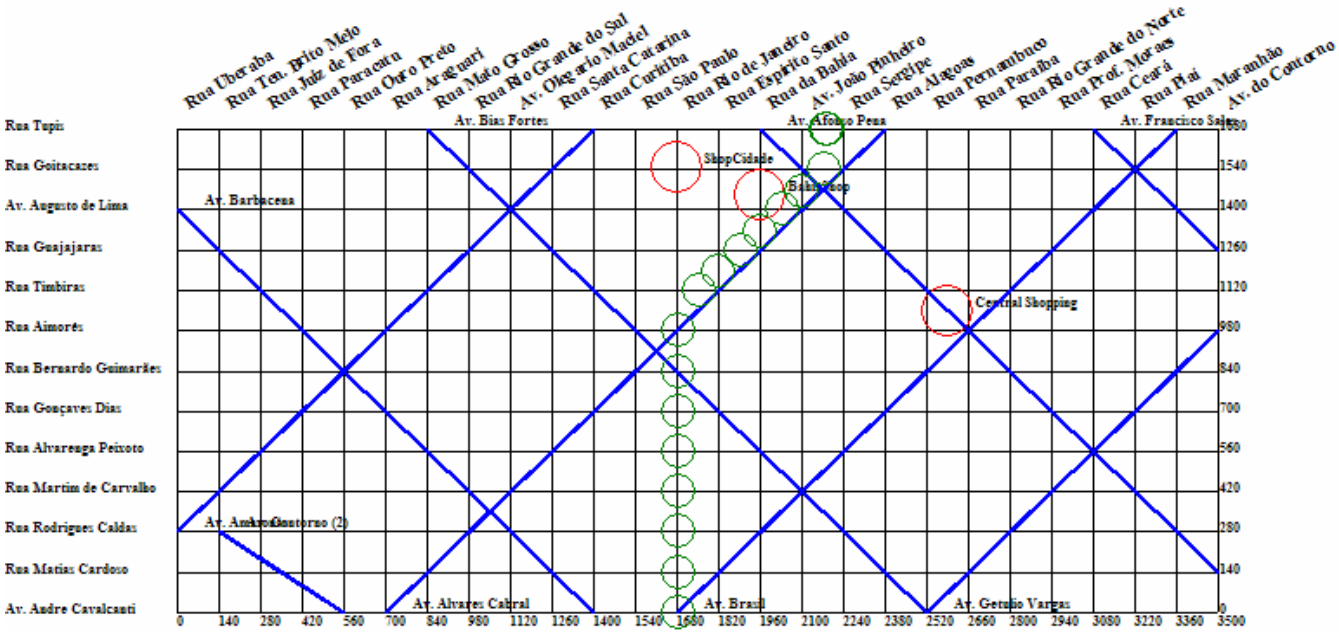


Figura 81- Pontos (círculos vermelhos) correspondentes aos shopping centers e regiões resultantes (círculos verdes) para o exemplo 8

4.4 –Aplicativo para Classificação de Dados por Meio de Redes Neurais *Multi Layer Perceptron*-MLP

4.4.1 Redes Neurais Artificiais

4.4.1.1 Definição

Segundo (Pádua, 2000), Redes Neurais Artificiais (RNAs) são sistemas paralelos distribuídos, constituídos por unidades (nós) de processamento simples que modelam um neurônio biológico, o *neurônio artificial*, capazes de aplicar funções matemáticas simples a dados recebidos em sua entrada. Este modelo é denominado MCP, por ter sido proposto por McCulloch e Pitts, em 1943.

A Figura 82 exibe um esquema do neurônio MCP. Ele é constituído por um vetor *x* de entradas $x_1, x_2, ..., x_n$ (que representam os dendritos de um neurônio biológico) e de uma saída *y* (que representa o axônio de um neurônio biológico). As sinapses de um neurônio biológico são representadas por um vetor *w* de pesos $w_1, w_2, ..., w_n$, acopladas às entradas do neurônio. A saída *y* de um neurônio MCP é ativada através da aplicação de uma *função de ativação f*, que recebe usualmente como parâmetro uma combinação linear *in* dos sinais de entrada x_i, da seguinte maneira:

$$in = \sum_{i=1}^{n} w_i x_i \quad (3)$$

A saída y é obtida então por

$$y=f(in) \quad (4)$$

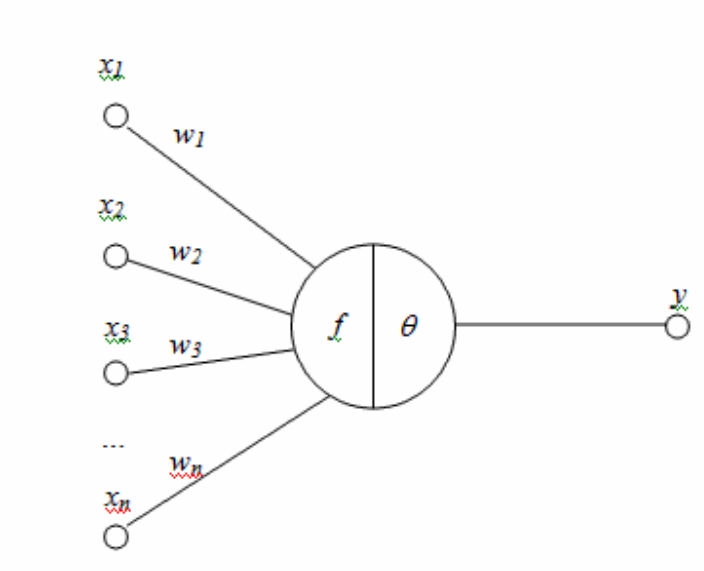


Figura 82-Esquema de um neurônio artificial

Podem ser utilizadas diferentes funções de ativação, mas as mais comuns são as seguintes:

- A função degrau, $\begin{cases} 1 & \sum w_i x_i \geq \theta \\ 0 & \sum w_i x_i < \theta \end{cases}$ onde θ é o limiar a partir do qual o neurônio dispara.

- A função sigmóide $f(x) = \frac{1}{1+e^{-\alpha x}}$

Ao utilizar-se a função degrau, a condição de disparo de um nodo MCP é

$$\sum_{i=1}^{n} w_i x_i = \theta \quad (5)$$

Para efeito de simplificação pode-se fazer

$$\sum_{i=1}^{n} w_i x_i - \theta = 0 \quad (6)$$

Isto equivale a adicionar às entradas da rede uma entrada com valor fixo $x_i = 1$ com um peso de valor $-\theta$. Assim tem-se

$w=\{ -\theta, w_1, w_2, ..., w_n \}$

$x=\{ 1, x_1, x_2, ..., x_n \}$

$$\sum_{i=1}^{n} w_i x_i = 0 \quad (7)$$

Segundo (Pádua, 2000), a função degrau restringe o uso do neurônio MCP à solução de problemas ditos *linearmente separáveis,* ou seja, aqueles cuja solução é obtida pela separação do espaço em duas regiões por meio de uma reta (se houver apenas duas entradas) ou um hiperplano (para n entradas). Se considerarmos apenas duas entradas x_1 e x_2 com pesos w_1 e w_2 e limiar θ, a condição de disparo do neurônio é $w_1 x_1 + w_2 x_2 = \theta$, que pode ser descrita no formato de uma equação de reta $x_2 = -(w_1 / w_2) x_1 + (\theta / w_2)$. A Figura 83 abaixo exibe a solução para os problemas das portas lógicas E e OU, com as respectivas retas-solução. Conforme pode ser visto, tais retas dividem o plano em duas regiões, evidenciando o caráter linearmente separável dos problemas. Entretanto, a Figura 84 mostra que o problema do OU-exclusivo (XOU) não é linearmente separável, não havendo portanto, solução para a equação acima nessas condições.

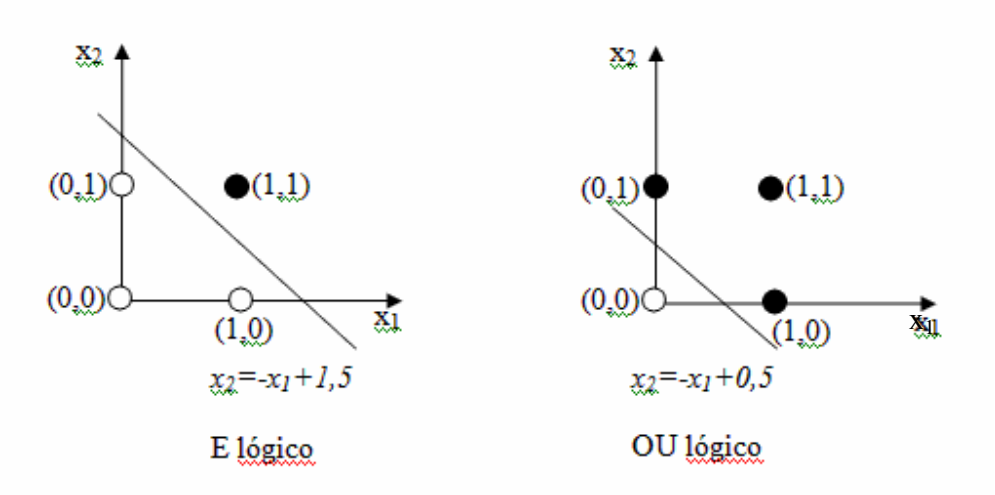


Figura 83– E e OU lógicos são linearmente separáveis-Fonte: (Russel, 2004)

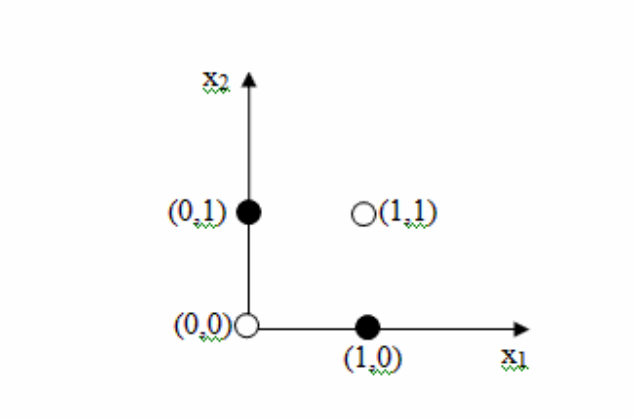


Figura 84-OU-exclusivo não permite a colocação de uma reta separando regiões de 1 e 0- Fonte: (Russel, 2004)

RNAs são muito utilizadas para classificação de um conjunto de dados fornecidos em sua entrada. O chamado *treinamento,* realizado por meio de *algoritmos de aprendizado,* permite ajustar os pesos das conexões, de forma que os dados de entrada sejam classificados discretamente na saída . Os algoritmos para aprendizado de redes neurais são classificados em dois tipos:

-*Algoritmos para treinamento supervisionado:* o treinamento é feito por meio de conjuntos de treinamento, ou seja, exemplos de entradas/saídas fornecidos à rede, que ajusta seus pesos com base neles;

-*Algoritmos de treinamento não-supervisionado:* tais algoritmos não demandam conjuntos de treinamento para aprendizado.

4.4.1.2 O Perceptron

Segundo (Pádua,2000) o trabalho de McCulloch e Pitts focou-se no modelamento matemático do neurônio biológico. Somente com o trabalho de Frank Rosemblatt, em 1958, o conceito de aprendizado de redes neurais foi introduzido. Rosemblatt propôs o denominado modelo *perceptron,* uma estrutura de rede contendo nós MCP e com um algoritmo de aprendizado.

O perceptron é composto por unidades de entrada (sensores, receptores de dados), unidades de associação (nós MCP com pesos fixos pré-determinados) e unidades de resposta (uma única camada de nós MCP a serem devidamente treinados).

(Pádua, 2000) demonstra que algoritmos de treinamento de perceptrons sempre convergem em tempo finito, para problemas linearmente separáveis. O algoritmo de treinamento para um nodo de um perceptron simples é descrito a seguir (Pádua, 2000).

O algoritmo de treinamento do perceptron

Algoritmos de treinamento supervisionado de RNAs normalmente se focam no ajuste de valores de *w* que permitam a classificação proposta no conjunto de treinamento constituído por pares de vetores entrada-saída (x^i, y^i). Em cada

iteração *t* do algoritmo obtém-se um incremento *Δw,* adicionado ao vetor *w* obtido na interação anterior: *w(t+1)=w(t)+ Δw.* Este *Δw* pode ser obtido de forma a se reduzir o chamado *erro quadrático médio* $E = \frac{1}{2}(Err)^2$, onde *Err= y_d - y_a* , y_d é a saída desejada e $y_a = f(\sum_{i=1}^{n} w_i x_i)$, a saída atual do perceptron.

Para reduzir o erro quadrático médio, utiliza-se o declínio de gradiante de *E*, que consiste em se calcular a derivada parcial de *E* em relação a cada peso. Assim tem-se (Russel, 2004):

$$\frac{\partial E}{\partial w_i} = \frac{\partial E}{\partial Err}\frac{\partial Err}{\partial w_i}$$

$$\frac{\partial E}{\partial w_i} = Err\frac{\partial Err}{\partial w_i}$$

$$\frac{\partial E}{\partial w_i} = Err\frac{\partial}{\partial w_i}(y_d - f(\sum_{i=1}^{n} w_i x_i))$$

Finalmente:

$$\frac{\partial E}{\partial w_i} = -Err \quad f'(\sum_{i=1}^{n} w_i x_i) \quad x_i \text{ (8)}$$

Entretanto, a regra de declínio de gradiente determina que o ajuste *Δw* deva ser feito na direção contrária à de $\frac{\partial E}{\partial w_i}$ ou seja , $\Delta w \quad \alpha - \frac{\partial E}{\partial w_i}$,podendo ser escrito como $\Delta w = -\eta\frac{\partial E}{\partial w_i}$, onde a constante de proporcionalidade η é denominada *taxa de*

aprendizado, determinante da rapidez com que os pesos são atualizados. Assim, obtemos finalmente a expressão de atualização de pesos:

$$w(t+1)=w(t)+\eta \quad Err \quad f'(\sum_{i=1}^{n} w_i x_i) \quad x_i \quad (9)$$

Se *f* for a função degrau, o termo $f'(\sum_{i=1}^{n} w_i x_i)$ pode ser omitido, pois é o mesmo para todos os pesos e sua omissão somente altera a magnitude, e não a orientação da atualização global de pesos para cada exemplo (Russel, 2004).

O algoritmo de treinamento pode ser assim formalizado (inspirado em (Pádua, 2000)):

1-Iniciar o vetor de pesos *w* com valores entre -1 e +1 e a taxa de aprendizado *η*;

2-Repetir

2.1-Para cada par entrada-saída desejada (x^j, y^j) de um conjunto de treinamento com *p* elementos $T=\{ (x^1, y^1), (x^2, y^2)... (x^p, y^p) \}$ faça:

2.1.1-Atualizar o vetor de pesos para cada um dos nodos da rede da seguinte forma:

$$y_a = f(\sum_{i=1}^{n} w_i x_i);$$

$$Err = y_d - y_a;$$

$$w(t+1)=w(t)+\eta \quad Err \quad f'(\sum_{i=1}^{n} w_i x_i) \quad x_i;$$

Até que *Err*=0 (ou algum outro critério de parada), para todos os *p* elementos do conjunto de treinamento e todos os nodos da rede.

4.4.1.3 Redes Multicamadas (MLP- *Multi Layer Perceptron)*

Conforme visto anteriormente, perceptrons de uma única camada somente resolvem problemas linearmente separáveis. (Pádua, 2000) afirma que a solução de problemas não linearmente separáveis somente é obtida com o uso de redes com uma ou mais camadas intermediárias ou escondidas. (Cybenko, 1988, 1989) demonstra que uma rede com uma única camada escondida pode implementar qualquer função contínua e que duas camadas escondidas permitem a aproximação de qualquer função. As redes do tipo *perceptron multicamadas* ou MLP (*multilayer perceptron)* caracterizam-se por apresentar a seguinte disposição de camadas (Figura 85):

- Camada de Entrada: onde os padrões de entrada são apresentados à rede;
- Camadas Intermediárias ou Escondidas: segundo (Russel, 2004), responsável pelo aumento do espaço de hipóteses que a rede é capaz de representar. Essencialmente são detectores de características (Pádua, 2000);
- Camada de Saída: onde o resultado do processamento é apresentado.

Em uma rede MLP, o processamento realizado por cada nodo é definido pela combinação linear dos processamentos realizados pelos nodos da camada anterior que estão conectados a ele (Pádua, 2000).

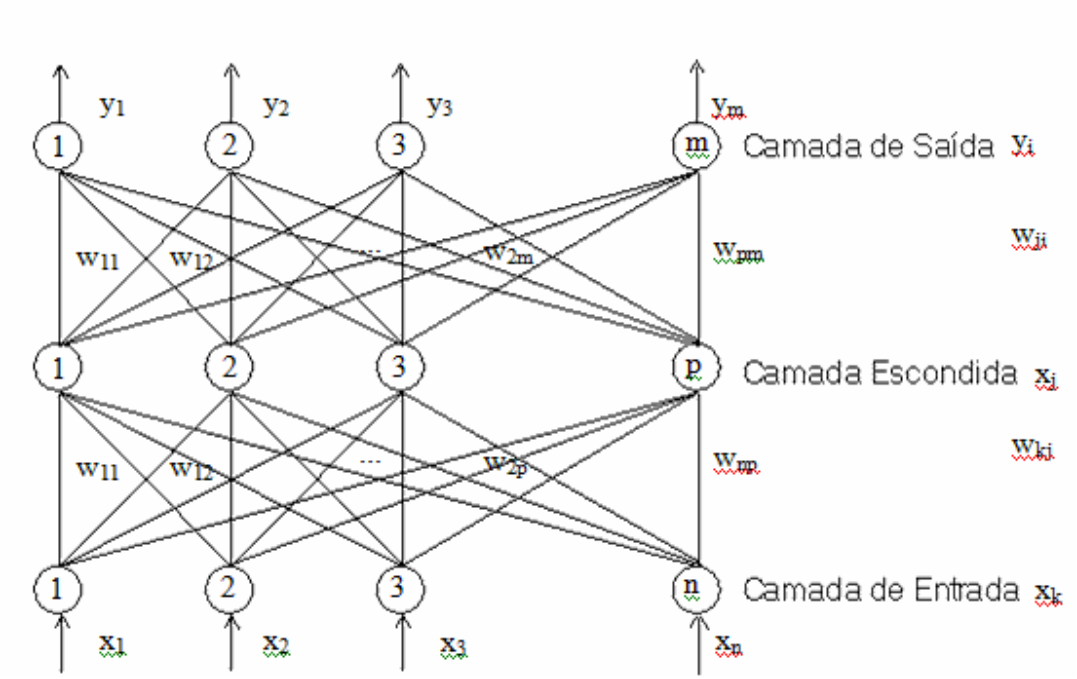


Figura 85- As várias camadas de uma rede MLP

Determinação da topologia de redes MLP

Na camada de entrada recomenda-se que o número de nodos MCP seja igual ao número de variáveis que produzem as saídas da rede MLP. Para determinar o número de neurônios na camada de saída, deve-se considerar o uso pretendido da rede. Se este for, por exemplo, classificar itens em grupos, é recomendável que haja um nodo MCP para cada grupo. Para modelagem de funções matemáticas, um neurônio é suficiente. Já para filtragem de ruídos, o ideal é que o número de neurônios na camada de saída seja igual ao da camada de entrada.

Sabemos que uma camada intermediária é suficiente para modelar qualquer função contínua e que duas camadas intermediárias modelam qualquer função. Entretanto, observa-se que problemas que requeiram duas camadas intermediárias são raros e que simplesmente não existe qualquer razão teórica para se utilizar mais que duas camadas.

A determinação do número de neurônios MCP nas camadas intermediárias é uma importante questão no projeto da arquitetura de uma RNA. Subestimar esse número pode resultar no chamado *underfitting:* a estrutura é insuficiente para manipular as entradas, dificultando sobremaneira a convergência dos algoritmos de treinamento. Por outro lado, um número excessivo de neurônios na camada intermediária usualmente gera *overfitting,* ou seja, a diminuição da capacidade de generalização. A determinação desse número depende de vários fatores, tais como:

- Número de neurônios de entrada e saída
- Número de exemplos de treinamento
- Quantidade de ruído presente nos exemplos
- Função de avaliação utilizada
- Complexidade da classificação ou da função a ser aprendida

Algumas sugestões para um ponto de partida na determinação do número de neurônios na camada intermediária são as seguintes:

- Um número intermediário entre o número de neurônios na camada de entrada e o número de neurônios na camada de saída
- 2/3 do número de neurônios na entrada mais o número de neurônios na saída
- Um número menor que o dobro de neurônios na camada de entrada (por exemplo, 2n - 1, onde n é o número de neurônios na entrada);
- Dez por cento do número de exemplos;

Entretanto, a experiência tem mostrado que a determinação do número de neurônios na camada intermediária é em essência um processo de tentativa e erro. Felizmente, existem refinamentos que podem ser aplicados à RNA, de forma a melhorar sua topologia para que possa modelar mais precisamente a classificação ou função embutida nos exemplos, mas que a torne também apta a generalizar,

quando necessário, considerando-se também, inclusive, o ruído possivelmente contido nos dados, que se deseja, não seja modelado.

Segundo (Reed, 1993) o *overfitting* pode ser diminuído utilizando-se um conjunto de treinamento para modificar os pesos mediante um algoritmo de aprendizado e um *conjunto de validação* para verificar a capacidade de generalização apreendida durante o aprendizado: interrompe-se o treinamento quando o erro produzido pelo conjunto de validação começar a subir.

As denominadas técnicas de *poda (pruning)* envolvem a eliminação de nodos e pesos da estrutura da rede. Uma primeira abordagem propõe a exclusão sistemática de nodos da rede, avaliando-se em seguida o impacto no erro de saída. Se este for pequeno, mantém-se a exclusão, pois sua ausência não afeta a capacidade da mesma em modelar os dados (Reed, 1993). Uma segunda abordagem, apresentada em (Pádua, 2000), consiste em relacionar o erro quadrático médio com a norma do vetor *w,* de forma a se obter soluções com pesos de norma mínima. Durante o treinamento, começa-se com uma rede superdimensionada e removem-se os pesos irrelevantes. (Hinton, 1987) propõe uma alteração na expressão do erro quadrático médio (exibida anteriormente), que incorpora o termo correspondente à norma do vetor *w:*

$$E = \frac{1}{2}(Err)^2 + \frac{1}{2}\lambda\|w\|^2 \quad (10)$$

Essa expressão é então minimizada, obtendo-se uma solução com pesos de norma mínima, na qual aqueles muito pequenos são eliminados. O parâmetro de regularização λ deve ser ajustado: se for muito grande, a solução para o vetor de pesos tende para w=0, se for nulo, somente a soma dos erros quadráticos é minimizada. Portanto, o desafio consiste em obter um valor intermediário de λ que seja satisfatório, entre esses extremos.

O algoritmo de treinamento backpropagation

É o algoritmo mais conhecido para treinamento de redes MLP. Trata-se de um algoritmo supervisionado que utiliza pares (entrada, saída desejada) para, por meio de um mecanismo de correção de erros, ajustar os pesos da rede (Pádua, 2000). O treinamento ocorre em duas fases, cada uma delas percorre a rede em um sentido: para frente (*forward*) em que a saída da rede é definida para determinado padrão de entrada, e para trás (*backward*), em que utiliza-se uma propagação do erro (diferença entre as saídas desejada e produzida pela rede) da camada de saída para as camadas ocultas de forma a atualizar os pesos das conexões.

Em redes MLP tem-se um vetor de saídas y_i, assim, o erro quadrático médio deve ser expresso como um somatório dos erros quadráticos médios de cada saída y_i:

$E = \frac{1}{2}\sum_{i=1}^{m}(Err)^2$ (11) onde $Err = yd_i - y_i$, yd_i é a saída desejada e $y_i = f(\sum_{j=1}^{p} w_{ji} x_j)$.

Vamos considerar inicialmente a correção dos pesos w_{ji} que conectam a camada escondida à camada de saída, ou seja, vamos obter o gradiente de E em relação aos pesos w_{ji}. Nesse caso específico, expandimos somente yi, pois todos os outros termos do somatório não são afetados por w_{ji} (Russel, 2004). Assim, nesse caso tem-se $E = \frac{1}{2}(Err)^2$.

$$\frac{\partial E}{\partial w_{ji}} = \frac{\partial E}{\partial Err}\frac{\partial Err}{\partial w_{ji}}$$

$$\frac{\partial E}{\partial w_{ji}} = Err\frac{\partial Err}{\partial w_{ji}}$$

$$\frac{\partial E}{\partial w_{ji}} = Err\frac{\partial}{\partial w_{ji}}(yd_i - f(\sum_{j=1}^{p} w_{ji}x_j))$$

$$\frac{\partial E}{\partial w_{ji}} = -(yd_i - f(\sum_{j=1}^{p} w_{ji}x_j)) \quad f'(\sum_{j=1}^{p} w_{ji}x_j) \quad x_j$$

Escrevemos finalmente:

$$\frac{\partial E}{\partial w_{ji}} = -x_j \Delta_i \text{, onde } \Delta_i = (yd_i - f(\sum_{j=1}^{p} w_{ji} x_j)) \quad f'(\sum_{j=1}^{p} w_{ji} x_j) \text{ (12)}$$

Analogamente ao que se considerou para o perceptron, podemos escrever então:

$$\Delta w_{ji} = -\eta \frac{\partial E}{\partial w_{ji}} = \eta x_j \Delta_i \text{ (13), onde } \eta \text{ é a taxa de aprendizado.}$$

Consideremos agora a correção de pesos w_{kj} que conectam a camada escondida à camada de entrada. Nesse caso não podemos ignorar o somatório, pois cada saída y_i pode ser afetada por w_{kj}.

$$\frac{\partial E}{\partial w_{kj}} = \sum_{i=1}^{m} \frac{\partial E}{\partial Err} \frac{\partial Err}{\partial w_{kj}}$$

$$\frac{\partial E}{\partial w_{kj}} = \sum_{i=1}^{m} Err \frac{\partial Err}{\partial w_{kj}}$$

$$\frac{\partial E}{\partial w_{kj}} = \sum_{i=1}^{m} ((yd_{\mathrm{i}} - f(\sum_{j=1}^{p} w_{ji} x_j)) \frac{\partial}{\partial w_{kj}} ((yd_i - f(\sum_{j=1}^{p} w_{ji} x_j))))$$

$$\frac{\partial E}{\partial w_{kj}} = -\sum_{i=1}^{m} ((yd_{\mathrm{i}} - f(\sum_{j=1}^{p} w_{ji} x_j)) f'(\sum_{j=1}^{p} w_{ji} x_j) \frac{\partial}{\partial w_{kj}} (f(\sum_{j=1}^{p} w_{ji} x_j)))$$

Mas, $\Delta_i = (yd_i - f(\sum_{j=1}^{p} w_{ji} x_j)) \quad f'(\sum_{j=1}^{p} w_{ji} x_j)$

$$\frac{\partial E}{\partial w_{kj}} = -\sum_{i=1}^{m} (\Delta_i \frac{\partial}{\partial w_{kj}} (f(\sum_{j=1}^{p} w_{ji} x_j)))$$

$$\frac{\partial E}{\partial w_{kj}} = -\sum_{i=1}^{m} (\Delta_i w_{ji} \frac{\partial}{\partial w_{kj}} x_j)$$

Mas $x_j = f(\sum_{k=1}^{n} w_{kj} x_k)$

$$\frac{\partial E}{\partial w_{kj}} = -\sum_{i=1}^{m} (\Delta_i w_{ji} \frac{\partial}{\partial w_{kj}} f(\sum_{k=1}^{n} w_{kj} x_k))$$

$$\frac{\partial E}{\partial w_{kj}} = -\sum_{i=1}^{m} (\Delta_i w_{ji} f'(\sum_{k=1}^{n} w_{kj} x_k) x_k)$$

Finalmente

$\frac{\partial E}{\partial w_{kj}} = -x_k \Delta_j$, onde $\Delta_j = f'(\sum_{k=1}^{n} w_{kj} x_k) \sum_{i=1}^{m} (\Delta_i w_{ji})$ (14)

E neste caso, também podemos fazer:

$\Delta w_{kj} = -\eta \frac{\partial E}{\partial w_{kj}} = \eta x_k \Delta_j$ (15), onde η é a taxa de aprendizado.

Uma vez derivadas as correções dos pesos entre as camadas, vamos utilizá-las para descrever detalhadamente o algoritmo *backpropagation:*

1-Iniciar pesos e parâmetros;

2-Repetir até o erro ser mínimo ou até a realização de um dado número de ciclos:

2.1-Para cada padrão de treinamento X

2.1.1-Definir a saída da rede através da fase *forward* (descrita abaixo);

2.1.2-Atualizar pesos dos nodos através da fase *backward* (descrita abaixo).

Considerando-se:

M= número de camadas da RNA

x_j^l = entrada dos nós na camada *l*,

y_j^l = saída dos nós na camada *l*,

w_{ij}^l = pesos na camada *l*,

Δ_j^l é o erro nas saídas da camada *l*

Fase *forward:*

Para *l=2* até *M* faça

p = número de nós da camada l;

q= número de nós da camada (l-1);

Para *j = 1 até p* faça

$$y_j^l = f(\sum_{i=1}^{q} w_{ij}^l \; x^l{}_i);$$

Se (*l=M*) faça

$\Delta_j^M = (yd_j - y_j^M) \quad f'(\sum_{i=1}^{q} w_{ij}^l \; x^M{}_i)$;// Calcula o erro na camada de saída

Fase *backward*

Para *l=M* até 2 faça

p = número de nós da camada l;

q= número de nós da camada (l-1);

r= número de nós da camada (l-2);

Para *i = 1 até q* faça

Para *j = 1 até p* faça

$$w_{ij}^{l} = w_{ij}^{l} + \eta x_i^{l} \Delta_j^{l};$$

Se *l* $\neq$ 2 então

Para *i = 1 até q* faça

$$\Delta_i^{l-1} = f'(\sum_{j=1}^{r} w_{ji}^{l-1} x_j^{l-1}) \sum_{j=1}^{p} (\Delta_j^{l} w_{ij}^{l});$$

Melhorias no treinamento

O principal problema no treinamento de RNAs pelo algoritmo *backpropagation* é a possibilidade de ocorrência de mínimos locais na superfície de erro, que provê uma solução incorreta. Para reduzir a incidência deste problema, bem como acelerar a convergência do algoritmo, pode-se fazer o seguinte (Pádua, 2000):

- Diminuir a taxa de aprendizado;
- Adicionar mais nós nas camadas intermediárias;
- Adicionar ruído aos dados;
- Acrescentar o chamado fator *momentum* à atualização dos pesos:

$w_{ij}(t) = w_{ij}(t-1) + \eta x_i \Delta_j + \alpha(w_{ij}(t-1) - w_{ij}(t-2))$ (16), onde *t* é a iteração atual do algoritmo, e α ajusta a intensidade do fator.

Quando houver a possibilidade de todas as entradas em uma RNA serem nulas, é fácil perceber o risco de não haver a atualização dos pesos. Esse problema pode ser evitado colocando-se uma entrada de valor unitário, denominada *bias,* nas

camadas onde houver o referido risco. Esta entrada produz pesos adicionais à camada onde está conectada e é tratada como se fosse uma entrada comum.

4.4.2 O Aplicativo

O aplicativo desenvolvido implementa a interface ISBMLP que especifica métodos para acesso a funcionalidades relacionadas a redes neurais MLP. Ele permite a definição, o treinamento e o teste de redes MLP, bem como a associação de classificações resultantes a objetos geográficos, que podem ser exibidas em mapas.

4.4.2.1 A Entrada de Dados do Aplicativo

A Figura 85 exibe a entrada de dados do aplicativo. Ele está associado a tabelas de fatos definidas no aplicativo para consultas em bases de dados multidimensionais (item 4.5). As variáveis de entrada e saída podem ser obtidas de duas formas:

- Por meio da seleção tabelas de fatos, fatos de entrada e de saída (correspondentes à classificação desejada), respectivamento no *combo box tabela de fatos* , no *list box Variáveis* e no *list box Saída.* Os botões *inserir* e *remover* permitem incluir e excluir variáveis de entrada e saída, que são exibidas na tabela *Conjunto de Treinamento* logo abaixo.

- Preenchendo-se a tabela de entrada de dados, quando se seleciona o *check box* “Usar somente tabela de entrada de dados abaixo”. A tabela *Conjunto Treinamento* contém tantas colunas quantos sejam os números de neurônios de entrada e saída definidos.

Pressionando-se o botão *Gerar Conjunto* a tabela *Conjunto Treinamento* é populada com dados do banco de dados. Esta tabela pode ser também

manualmente preenchida e seus dados salvos e recuperados em arquivos. A tabela mais abaixo permite entrar dados de entrada e saída e testar a rede treinada.

Os dados classificados podem ser vinculados a campos geográficos (*combo Campo Dimensão Espacial)* e exibidos em mapa, cuja legenda pode ser definida na tabela *Legenda.* A Figura 87 exibe o mapa resultante para uma classificação de municípios de Minas Gerais com base nas variáveis RENDAPERCAPITA (renda per capita), TOTALPOPALF (população alfabetizada),IDHMLONG (IDH-M longevidade), IDHMEDU (IDH-M educação),IDHMREN (IDH-M renda),POPRUR (população rural),POPURB (população urbana). Este trabalho foi publicado e encontra-se detalhado em (Miranda Jr. & Abreu, 2008).

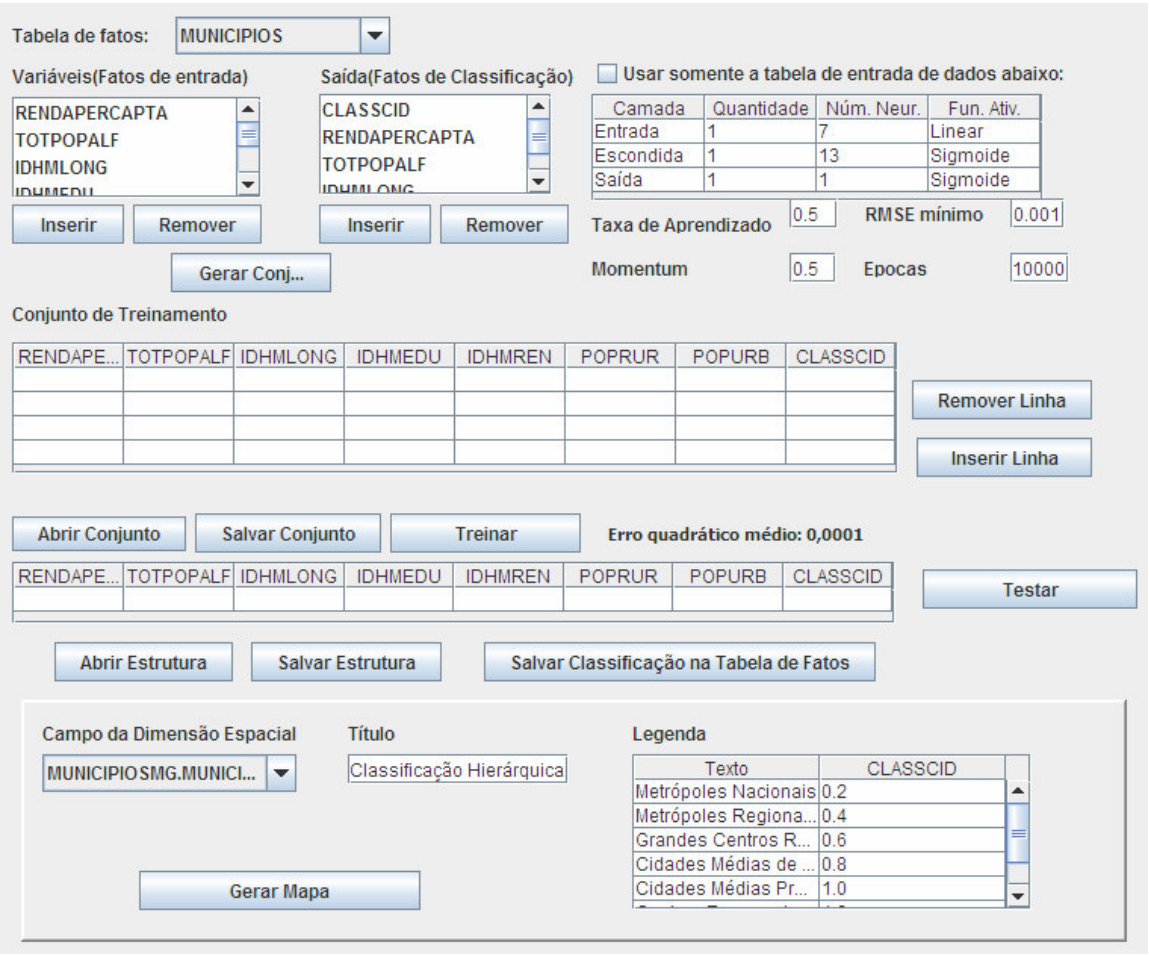


Figura 86-Entrada de dados do aplicativo para classificação por redes neurais MLP

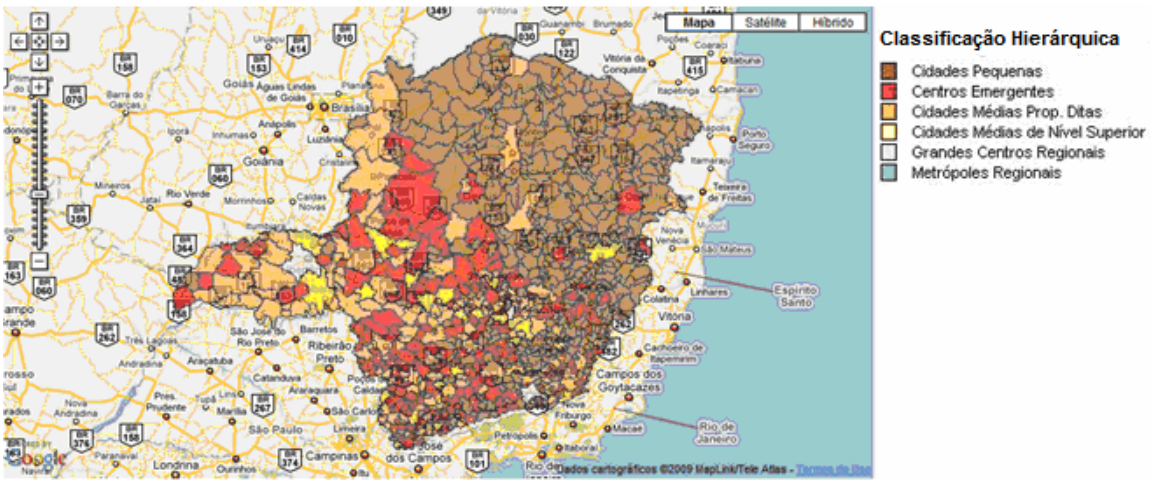


Figura 87-Classificação hierárquica das cidades de MG obtidas com o aplicativo Ranking das Universidades da Shanghai.

4.5 -Aplicativo para Consultas Em Bases de Dados Multidimensionais

Esta ferramenta foi desenvolvida com os propósitos de permitir a execução de consultas OLAP em bases de dados multidimensionais convencionais e espaciais e gerar facilmente aplicações conforme o layout e estrutura definidos na proposta.

4.5.1 - Informações Exibidas na Ferramenta

A Figura 88 exibe o modelo de dados da ferramenta e é, em essência, um metamodelo simplificado do modelo dimensional proposto neste trabalho. A tabela *FATO* armazena os fatos/medidas contidos nas tabelas de fatos, armazenadas em *TABELA_FATOS*. Igualmente, a tabela *DIMENSAO* armazena informações sobre dimensões e os campos dessas são armazenados em DIM_CAMPOS. *TABFATDIM* armazena relações entre tabelas de fatos e suas respectivas dimensões.

A Figura 89 exibe a interface de uma aplicação desenvolvida para possibilitar a população eficiente das tabelas supra citadas. Ela permite a seleção de tabelas de fatos e de dimensões no *combo box Tabelas.* As grades *Tabelas de Fatos Selecionadas* e *Tabelas de Dimensões Selecionadas* são editadas para armazenamento, respectivamente, dos nomes real e amigável (nome inteligível pelo usuário) das tabelas de fatos do modelo e do nome real, nome amigável e tipo (E, espacial, C, Convencional e T, temporal) das tabelas de dimensões. A grade *Medidas* permite a edição de informação sobre medidas(fatos) existentes na tabela de fatos: nome da medida, nome amigável e *usa,* campo booleano que informa se aquela medida vai ser exibida ou não na ferramenta OLAP. A grade *Campos ,* edita informação sobre campos da tabela de dimensões: nome do campo, nome amigável, *Chave Prim., booleano* que informa se é chave primária, *Cód. chave*, que armazena o código da chave estrangeira caso o campo em questão esteja vinculado a outra tabela de dimensões (em um modelo *snow flake)* e *usa,* campo booleano que informa se aquele campo vai ser exibido ou não na ferramenta OLAP. Os botões > e < colocam e retiram tabelas de fatos e dimensões das grades, conforme a tabela selecionada no *combo box Tabelas.*

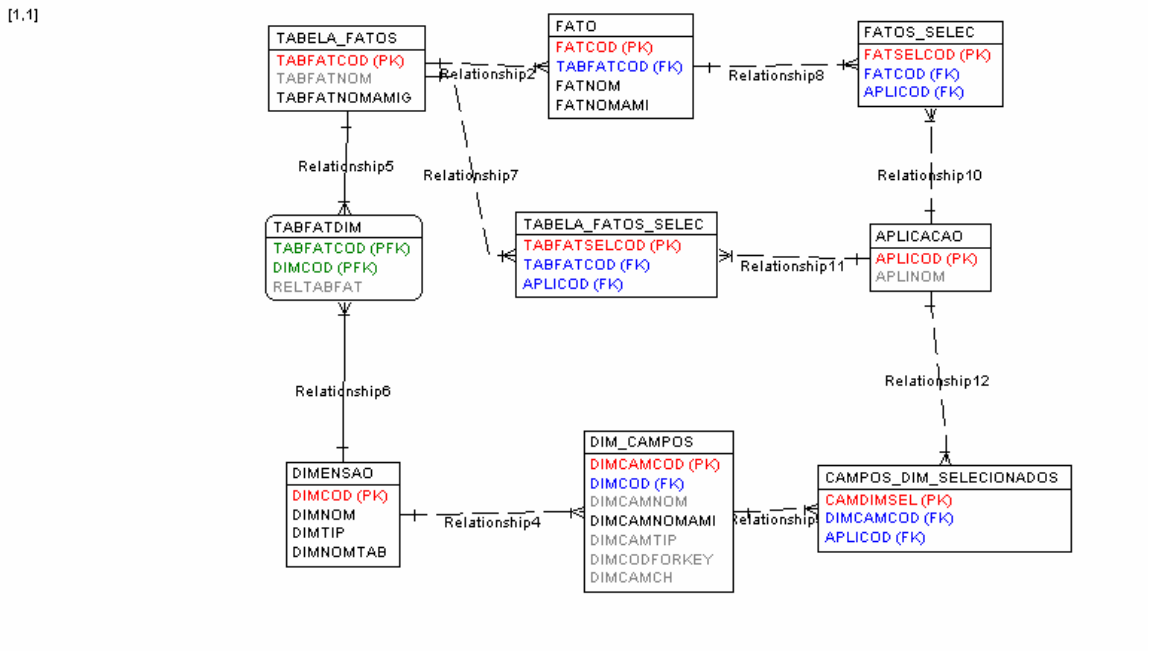


Figura 88-Modelo de dados da Ferramenta

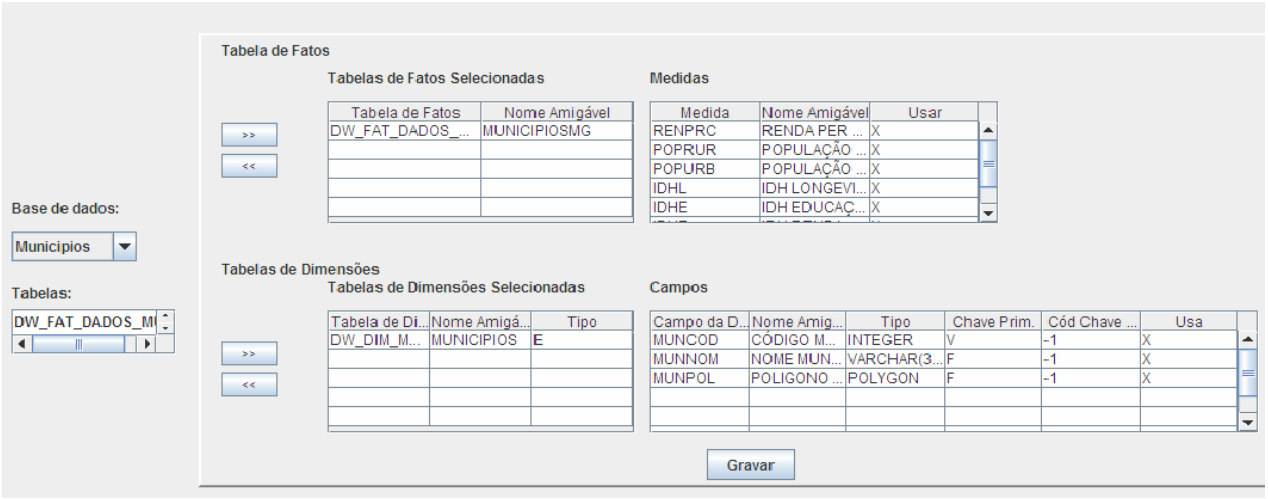


Figura 89-Interface para inserção de dados nas tabelas exibidas na Figura 88

4.5.2 -A Interface da Ferramenta

A Figura 90 exibe a interface para geração de consultas em bases multidimensionais. A geração das consultas é feita pela seleção de fatos, dimensões e campos das dimensões, nos *combo boxes* da interface, na seguinte seqüência:

- *Tabela de Fatos:* permite selecionar as tabelas de fatos a serem analisadas. Na Figura 90, a tabela de fatos *Rotas* foi selecionada. Tão logo uma tabela seja selecionada, os *combo boxes Fatos* e *Dimensões* são populados com os fatos e dimensões vinculados a esta tabela selecionada.

- *Fatos:* selecionam-se os fatos que comporão as células do cubo. Na Figura 90, foram selecionadas as medidas *Quilômetros Rodados* e *Velocidade Máxima*.

- *Dimensões: combo box* para seleção das dimensões que comporão o cubo. No exemplo, selecionaram-se as dimensões *Tempo, Motorista* e *Veículo.*

- *Campos das dimensões:* correspondem aos campos das dimensões selecionadas que serão exibidos. No exemplo, foram selecionados os campos, *Dia, Hora, Nome Motorista e Placa.*

Cada informação selecionada nos *combo boxes* é exibida na lista à direita dos mesmos. O botão *Remover* permite remover uma seleção na lista. O *combo box média,* permite fazer agregações de fatos e dimensões: média, máximo, mínimo, soma e contagem para campos numéricos, área, centróide, intersecção e união para campos espaciais. O botão *Exportar Estrutura* permite exportar a estrutura de fatos e dimensões em um arquivo XML que será utilizado para geração de uma aplicação específica, conforme *layout* exibido na Figura 39.

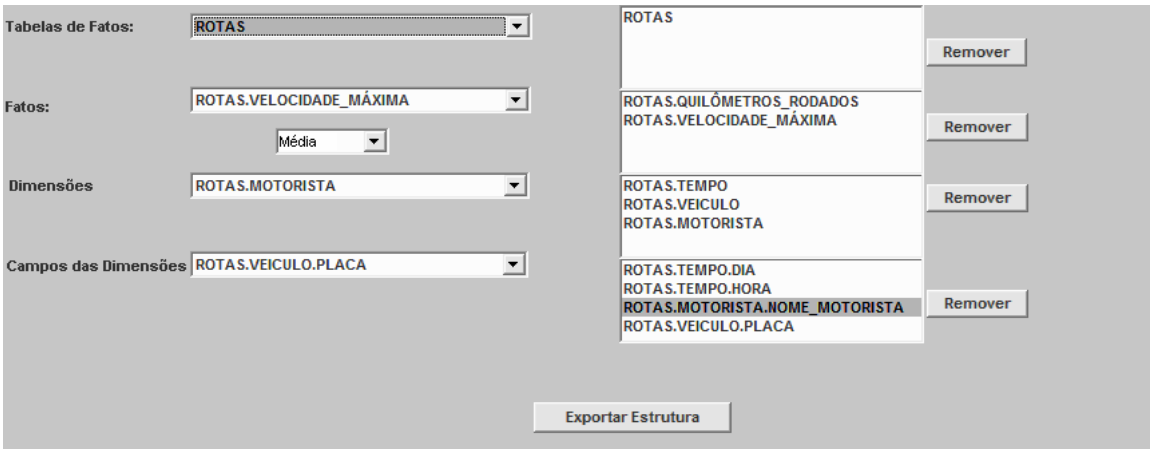


Figura 90-Interface para seleção de dados para consulta OLAP

Na Figura 91, é exibida a tela para realização de filtragens, seleções e operações espaciais. As linhas da grade abaixo armazenam consultas (*queries)* a serem realizadas sobre a base multidimensional, segundo a seguinte sintaxe:

Query=[()]Clásula[),and, or,(] [query]

Cláusula = [()][Campo Dimensão | Fato|Função] [operador relacional| operador espacial] [valor numérico| Função| Campo Dimensão | Fato]

[Operador relacional] = [>|<|>=| <=|<>]

[Operador espacial] = [contém | disjunto | igual a | intersecciona | sobrepõe| toca| está dentro de| distância| cruza]

Função= [média| máximo| mínimo| soma| contagem| área| centróide| intersecção| união]([Parâmetros])

Parâmetros= Parâmetro [,Parâmetro]

Parâmetro=[Fato|Campo Dimensão|valor numérico]

Cada linha da grade contém uma cláusula que comporá a *query* e é preenchida através de seleções nos *combo boxes,* que preenchem as seguintes colunas:

- *(:* abre parênteses na *query*

- *Fato/Dim:* esta coluna é preenchida selecionando-se campos das dimensões ou fatos nos *combo boxes Dimensões Espaciais, Dimensões Temporais* e *Dimensões Comuns.* Podem também ser utilizadas funções contidas no *combo box Funções,* conforme descrito na sintaxe da *query* supra descrita.

- *Oper. Rel.* corresponde aos operadores utilizados na cláusula que, conforme visto acima, podem ser relacionais ou espaciais

- *Valor:* nesta coluna são colocados os valores com os quais são comparados os fatos ou dimensões da cláusula. Os valores numéricos podem ser digitados ou selecionados no *combo box* valores, que é preenchido com valores presentes na base de dados, quando fatos ou dimensões são selecionados nos *combo boxes* correspondentes. Podem também ser utilizadas funções contidas no *combo box Funções,* conforme descrito na sintaxe da *query* supra descrita.

- *Oper Lóg.:* operadores lógicos *e* e *ou* que relacionam cláusulas.

- *):* fecha parênteses na *query*

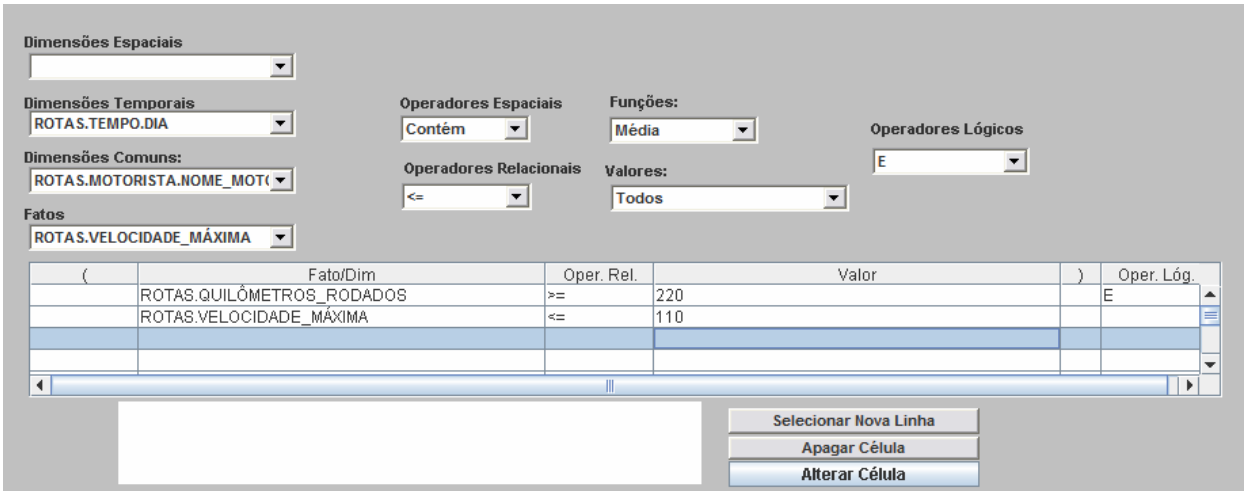


Figura 91-Interface para definição de filtros, seleção de dados por meio de operadores espaciais e convencionais e funções

A Figura 92 exibe a tela para definição de agregações (agrupamentos) de dimensões (*drill up)* e a ordenação do relatório final.

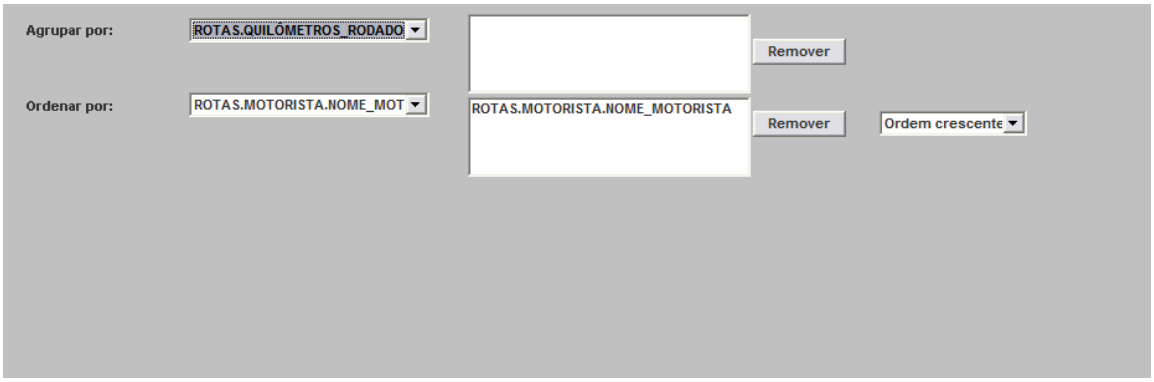


Figura 92-Definição de campos de agrupamento e ordenação

Finalmente, para emissão do relatório contendo os dados do cubo selecionado, utiliza-se a interface exibida na Figura 93. O relatório resultante é exibido na Figura 94.

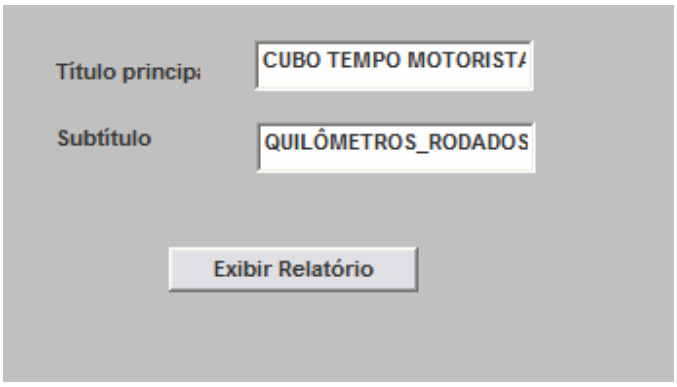
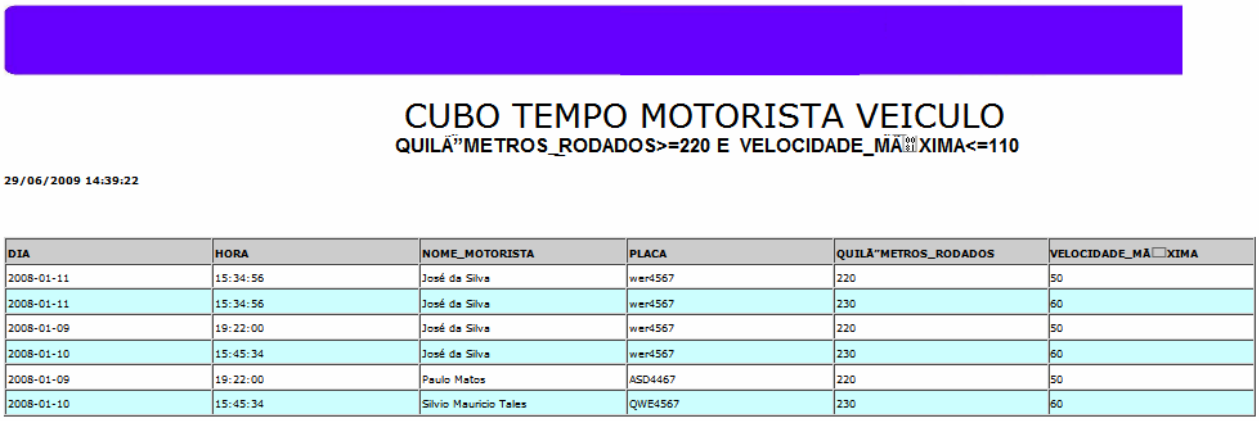


Figura 93-Interface para exibição de relatórios



CUBO TEMPO MOTORISTA VEICULO

QUILÃ"METROS_RODADOS>=220 E VELOCIDADE_MÃXIMA<=110

29/06/2009 14:39:22

DIA	HORA	NOME_MOTORISTA	PLACA	QUILÃ"METROS_RODADOS	VELOCIDADE_MÃXIMA
2008-01-11	15:34:56	José da Silva	wer4567	220	50
2008-01-11	15:34:56	José da Silva	wer4567	230	60
2008-01-09	19:22:00	José da Silva	wer4567	220	50
2008-01-10	15:45:34	José da Silva	wer4567	230	60
2008-01-09	19:22:00	Paulo Matos	ASD4467	220	50
2008-01-10	15:45:34	Silvio Mauricio Tales	QWE4567	230	60

Figura 94- Relatório de Consulta Gerado

A ferramenta permite a exibição de gráficos de barras, setores (*pie chart)* ou linhas. A Figura 95 exibe o tela para geração dos gráficos. Permite a seleção dos fatos e campos das dimensões a serem exibidos nos eixos vertical e horizontal do gráfico. As Figuras 96 e 97 exibem gráficos produzidos com a ferramenta.

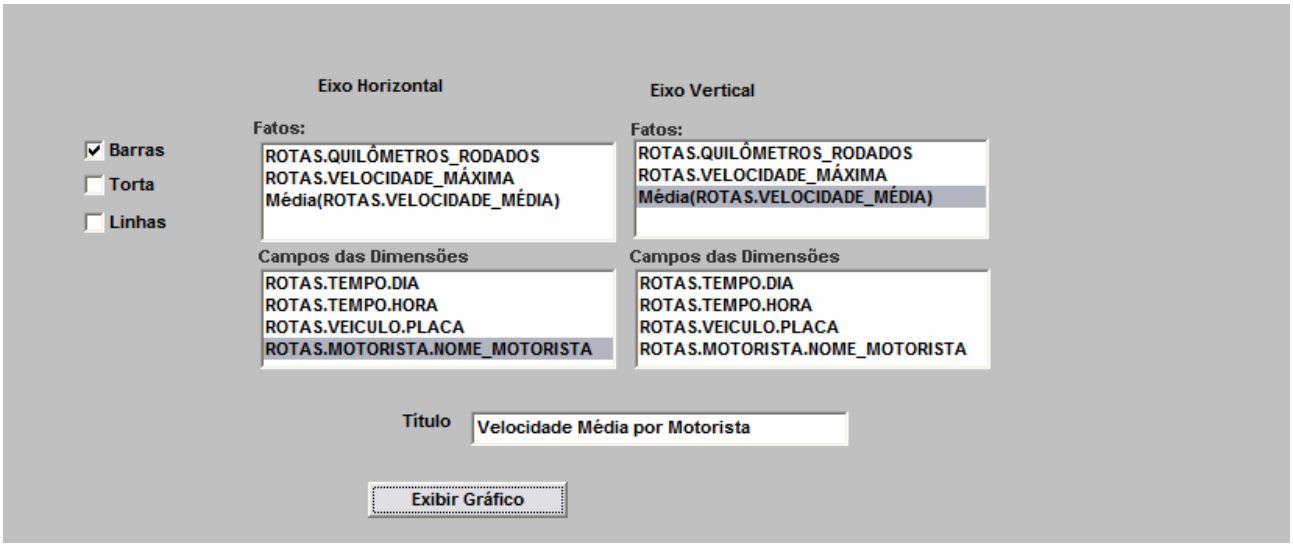


Figura 95-Interface para exibição de gráficos

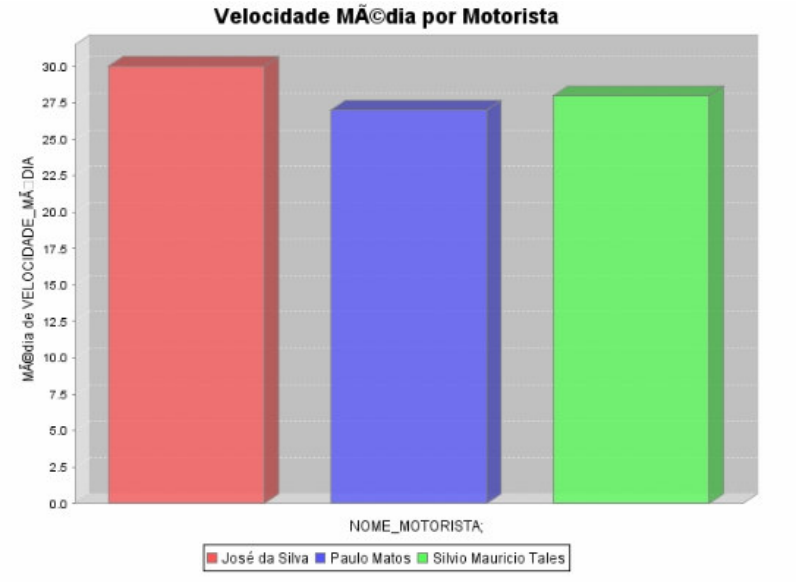


Figura 96-Gráfico de Barras Gerado

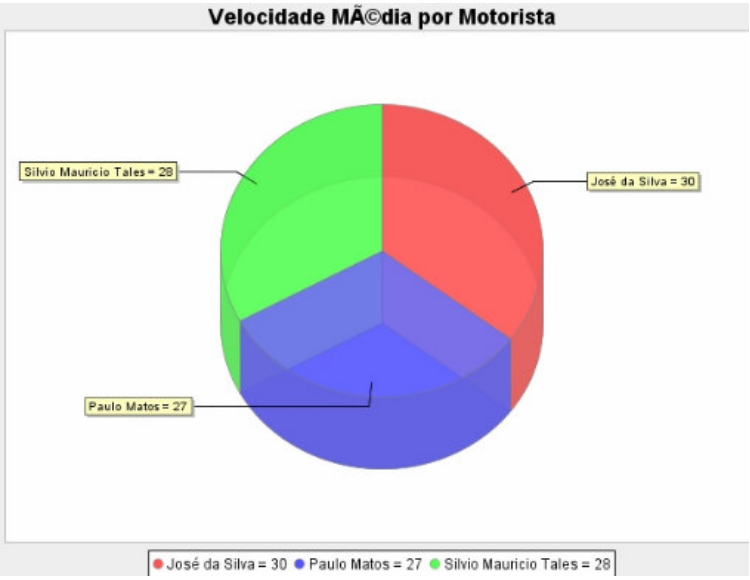


Figura 97-Gráfico de setores gerado

Quando dimensões ou medidas forem espaciais, mapas podem ser gerados a partir da interface exibida na Figura 98, na qual seleciona-se o fato a ser exibido ou objetos (quando se desejar objetos resultantes de uma análise espacial). Para

produção da legenda do mapa, pode-se definir o número de faixas. O botão *Redistribuir* recalcula as faixas em função do número digitado no campo *Faixas,* à esquerda. No *list box* logo abaixo, as faixas podem ser modificadas: ao se selecionar uma delas, seus limites superior e inferior podem ser alterados nos campos à direita do *list box.*

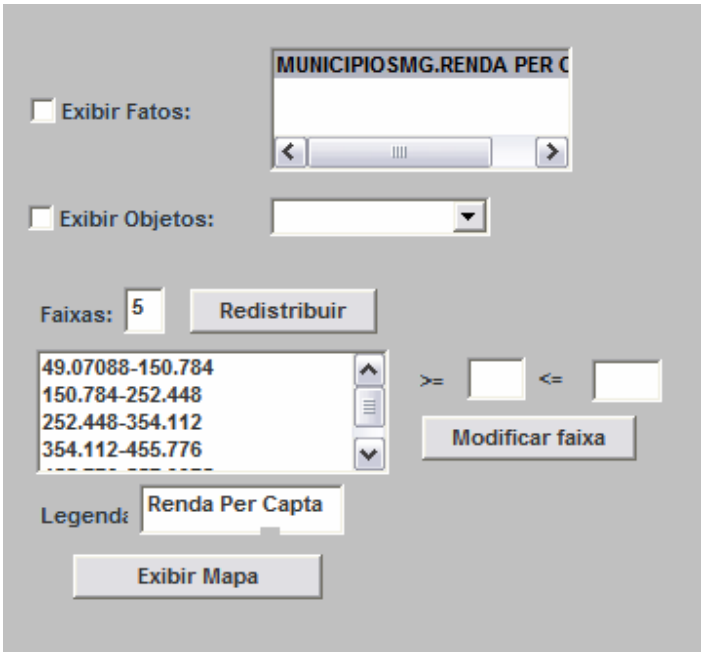


Figura 98-Interface para exibição de mapas

A Figura 99 exibe o mapa resultante. A Figura 100 mostra um mapa em que foram selecionados somente municípios de Minas Gerais com população acima de 20.000 habitantes.

Utilizando-se a interface mostrada na Figura 91 podem-se realizar análises espaciais complexas, utilizando-se operadores e funções espaciais relacionados mais acima.

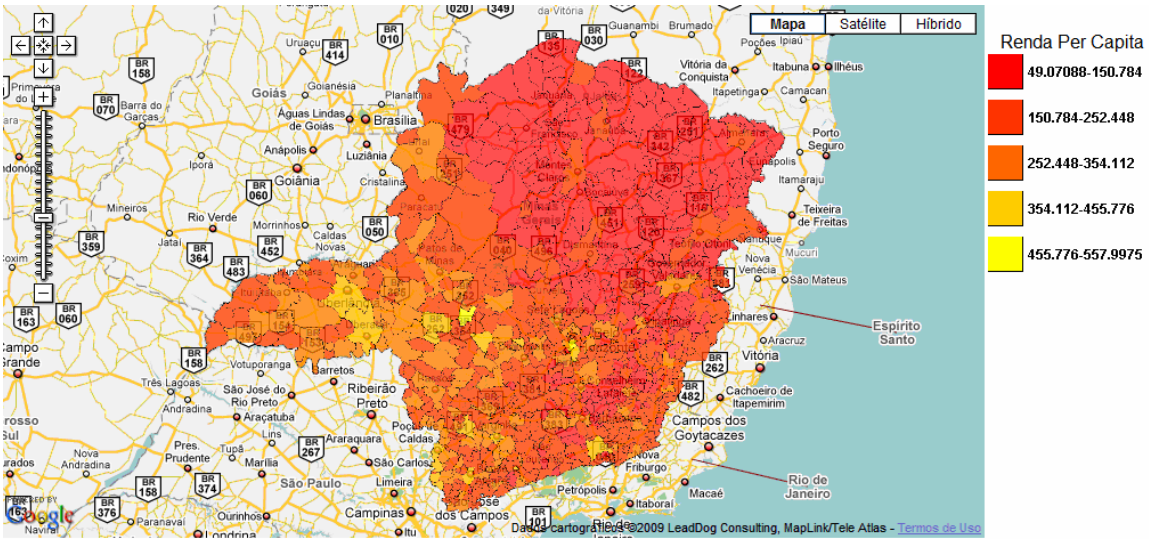


Figura 99- Mapa de municípios de Minas Gerais exibido sobre o Google Maps

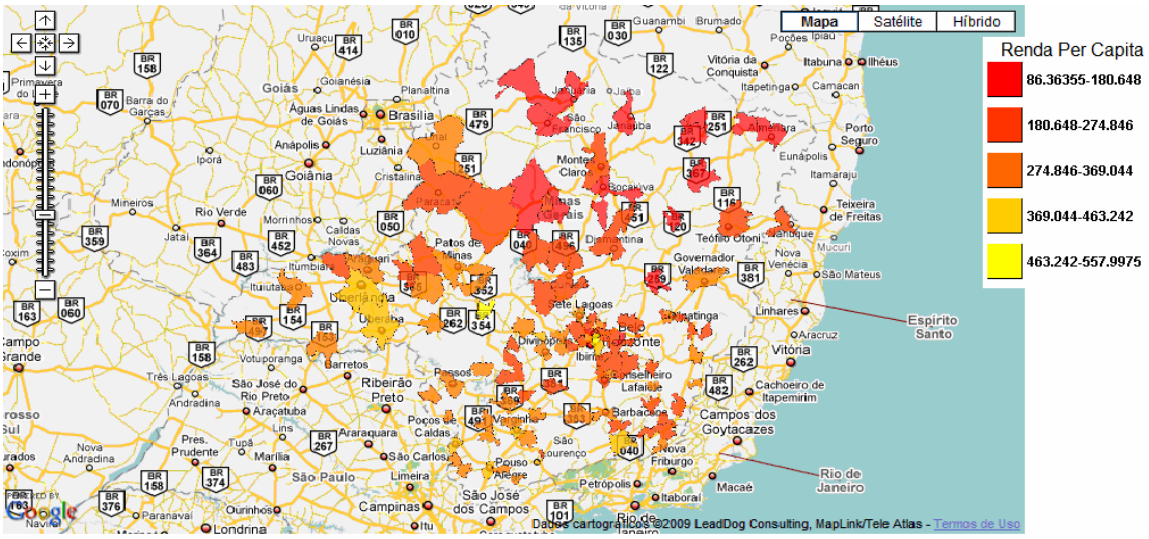


Figura 100- Mapa exibido para municípios com população acima de 20000 habitantes.

4.6 Mapeamento do Ranking Acadêmico Mundial de Universidades

Este aplicativo-piloto foi criado para teste da infraestrutura proposta e é apresentado em (Abreu & Miranda Jr, 2007b e 2009). Inicialmente gerou-se um *data mart* com dados provenientes do *Ranking* Acadêmico Mundial de Universidades, produzidos pela Universidade de Shanghai. Os dados foram extraídos de planilhas Excell, georeferenciados e associados a objetos polígono extraídos de mapas de países em formato MIF (Mapinfo) e armazenados em um banco de dados MySQL com extensões espaciais.

Sobre este *data mart* desenvolveu-se uma aplicação *web* segundo a infraestrutura arquitetural de software proposta neste trabalho, para consulta sobre informações do *ranking* em mapas, relatórios e gráficos. O *layout* geral da aplicação segue o padrão exibido na Figura 39.

A Figura 101 exibe o modelo dimensional utilizado na aplicação, constituído por uma tabela de fatos sobre o ranking (*dw_fat_rank_instituicoes)* e as tabelas de dimensões *dw_dim_tempo (*dimensão tempo), *dw_dim_regiao* (dimensão continente), *dw_dim_pais (*dimensão país), *dw_dim_cid (*dimensão cidade) e *dw_dim_instituicao* (dimensão instituição de ensino superior).

[1,1]

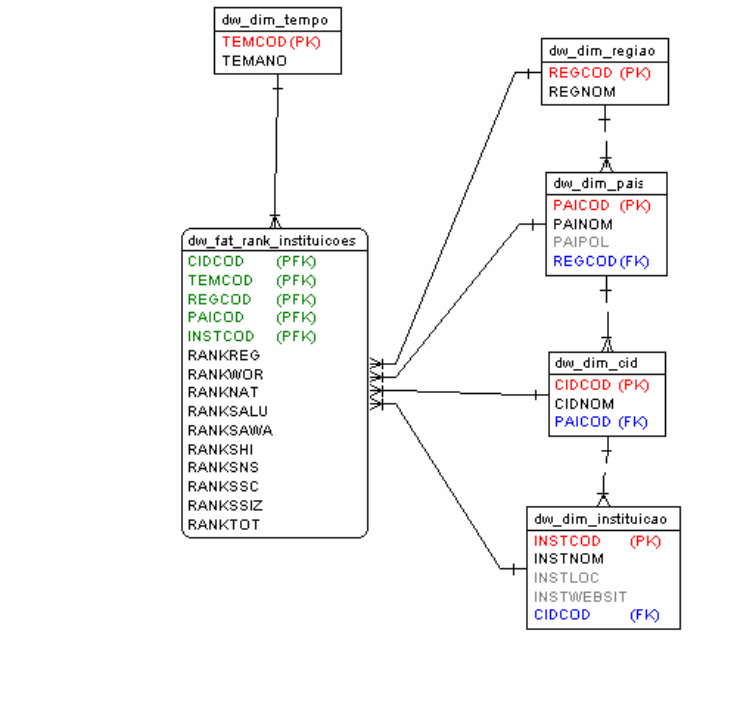
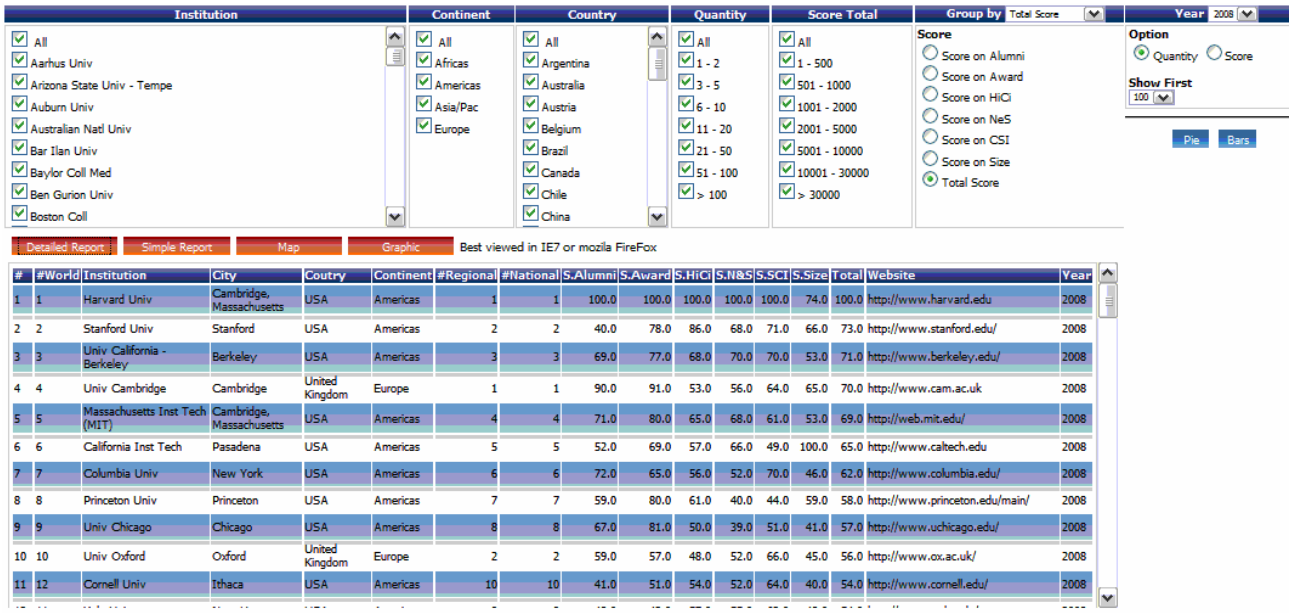


Figura 101-Modelo Multidimensional da Aplicação

A tela principal da aplicação *web* está exibida na Figura 102.



#	#World	Institution	City	Coutry	Continent	#Regional	#National	S.Alumni	S.Award	S.HiCi	S.N&S	S.SCI	S.Size	Total	Website	Year
1	1	Harvard Univ	Cambridge, Massachusetts	USA	Americas	1	1	100.0	100.0	100.0	100.0	100.0	74.0	100.0	http://www.harvard.edu	2008
2	2	Stanford Univ	Stanford	USA	Americas	2	2	40.0	78.0	86.0	68.0	71.0	66.0	73.0	http://www.stanford.edu/	2008
3	3	Univ California - Berkeley	Berkeley	USA	Americas	3	3	69.0	77.0	68.0	70.0	70.0	53.0	71.0	http://www.berkeley.edu/	2008
4	4	Univ Cambridge	Cambridge	United Kingdom	Europe	1	1	90.0	91.0	53.0	56.0	64.0	65.0	70.0	http://www.cam.ac.uk	2008
5	5	Massachusetts Inst Tech (MIT)	Cambridge, Massachusetts	USA	Americas	4	4	71.0	80.0	65.0	68.0	61.0	53.0	69.0	http://web.mit.edu/	2008
6	6	California Inst Tech	Pasadena	USA	Americas	5	5	52.0	69.0	57.0	66.0	49.0	100.0	65.0	http://www.caltech.edu	2008
7	7	Columbia Univ	New York	USA	Americas	6	6	72.0	65.0	56.0	52.0	70.0	46.0	62.0	http://www.columbia.edu/	2008
8	8	Princeton Univ	Princeton	USA	Americas	7	7	59.0	80.0	61.0	40.0	44.0	59.0	58.0	http://www.princeton.edu/main/	2008
9	9	Univ Chicago	Chicago	USA	Americas	8	8	67.0	81.0	50.0	39.0	51.0	41.0	57.0	http://www.uchicago.edu/	2008
10	10	Univ Oxford	Oxford	United Kingdom	Europe	2	2	59.0	57.0	48.0	52.0	66.0	45.0	56.0	http://www.ox.ac.uk/	2008
11	12	Cornell Univ	Ithaca	USA	Americas	10	10	41.0	51.0	54.0	52.0	64.0	40.0	54.0	http://www.cornell.edu/	2008

Figura 102-Tela inicial da aplicação web. Fonte: (Abreu & Miranda Jr, 2007b e 2009)

As seções *Institution, Continent* e *Country* permitem a seleção respectivamente de instituições, continentes e países a serem exibidos. O *combo-box Group By* permite fazer agregações e desagragações (*Drill up e Drill Down)* do nível mais alto ao mais baixo: continente, país, instituição. A Figura 103 exibe agregações obtidas em níveis de continente, país e instituição, respectivamente.

Na Figura 102, a visualização de gráficos *torta* ou *barras* é obtida pressionando-se o botão *Mapa.* A Figura 104 exibe um gráfico obtido para o *score* total por continente.

O botão *Map* exibe o mapa cloroplético de quaisquer seleções efetuadas. A Figura 105 exibe o mapa coroplético do *score* total por país obtido pela aplicação.

Os fatos (medidas) a serem exibidos em gráficos e no mapa são selecionados na seção *Score.* No canto superior direito a dimensão *tempo (*ano neste caso) pode ser selecionado de forma a exibir o *score* em um ano específico (de 2003 a 2008).

As seções *Quantity* e *Score Total* permitem a seleção de países ou continentes por *quantidade* de instituições existentes e por faixa de *score.* A Figura 106 exibe um mapa em que se destacam somente países cujos *scores* totais estejam entre 501 e

1000 no ano de 2007. Estes países estão também exibidos no relatório da Figura 107.

Total Score	Score
Americas	4179.00
Europe	3450.00
Asia/Pac	1399.00
Africas	36.00

Total Score	Score
USA	3665.00
United Kingdom	866.00
Germany	628.00
Japan	471.00
Canada	384.00
France	352.00
Italy	279.00
Australia	251.00
Netherlands	222.00
China	199.00
Sweden	192.00
Switzerland	182.00
Belgium	122.00
Israel	116.00
Spain	107.00
South Korea	97.00
Denmark	88.00
Austria	84.00
China-tw	79.00

Total Score	Score
Harvard Univ	100.00
Stanford Univ	73.00
Univ California - Berkeley	71.00
Univ Cambridge	70.00
Massachusetts Inst Tech (MIT)	69.00
California Inst Tech	65.00
Columbia Univ	62.00
Princeton Univ	58.00
Univ Chicago	57.00
Univ Oxford	56.00
Cornell Univ	54.00
Yale Univ	54.00
Univ California - Los Angeles	52.00
Univ California - San Diego	50.00
Univ Pennsylvania	49.00
Univ Washington - Seattle	48.00
Univ Wisconsin - Madison	47.00
Univ California - San Francisco	46.00
Tokyo Univ	46.00

Figura 103-Três níveis de agregação obtidos na aplicação: continente, país e instituição respectivamente. Fonte: (Abreu & Miranda Jr, 2007b e 2009)

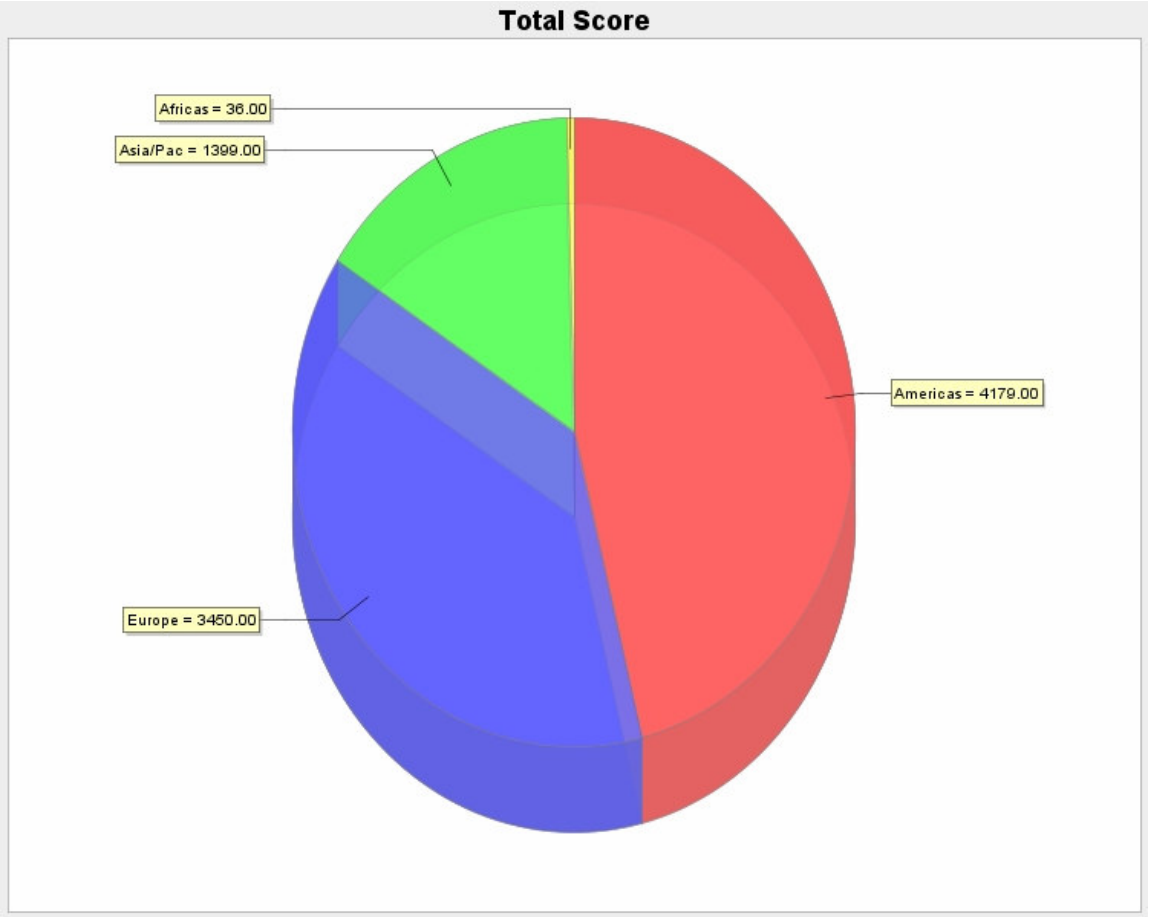


Figura 104 – Gráfico de setores-Escore total por continente. Fonte: (Abreu & Miranda Jr, 2007b e 2009)

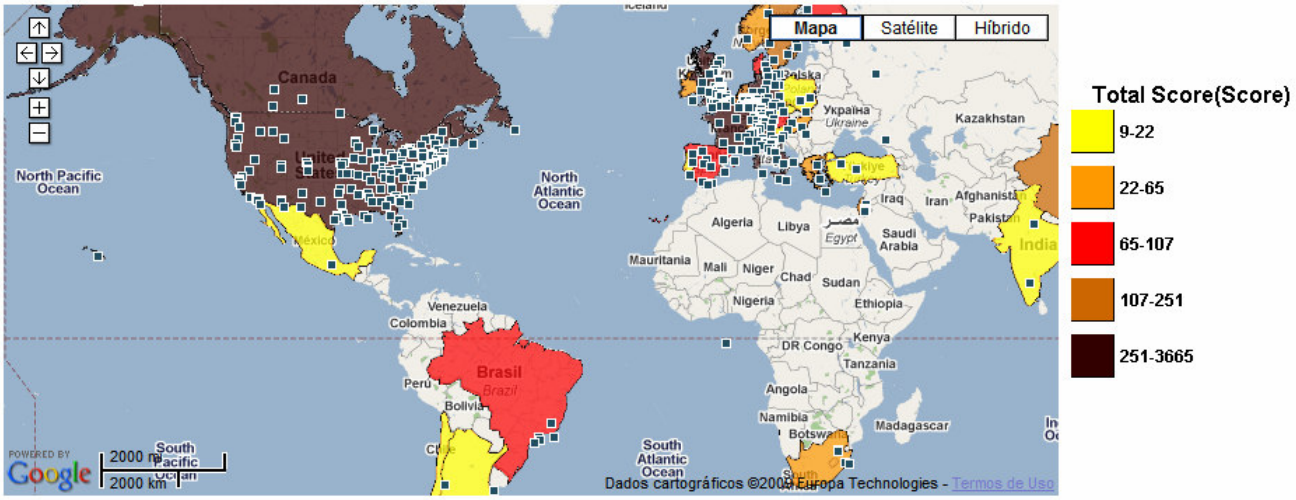


Figura 105-Mapa coroplético – agregação de score total por país. Fonte: (Abreu & Miranda Jr, 2007b e 2009)

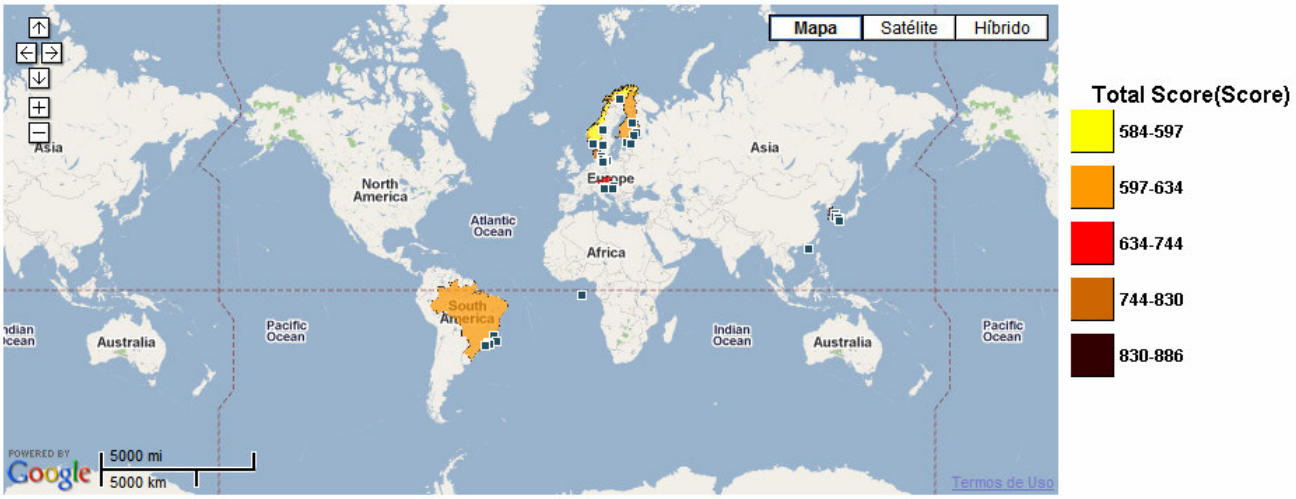


Figura 106-Mapa coroplético – países com score entre 501 e 1000 em 2007 Fonte: (Abreu & Miranda Jr, 2007b e 2009)

Total Score	Score
South Korea	886.20
Denmark	830.10
Austria	743.60
Finland	634.30
Brazil	615.20
China-hk	597.50
Norway	584.40

Figura 107- Países com score total na faixa de 501 a 1000 em 2007. Fonte: (Abreu & Miranda Jr, 2007b e 2009)

5 Conclusões

Informações espaciais têm se tornado de grande utilidade à análise de dados e auxiliam significativamente na tomada de decisões em organizações. Este trabalho procura mostrar que técnicas e ferramentas adaptadas às particularidades de tais informações são necessárias para tal. Neste sentido, contribui através de uma proposta metodológica aderente a tais particularidades, que integra um processo de desenvolvimento, uma infraestrutura arquitetural e um conjunto de APIs que especificam interfaces para funcionalidades ligadas à visualização , tratamento e análise de informações espaciais. Os aplicativos desenvolvidos demonstram a utilização de diversas estruturas e recursos da proposta, evidenciando-se também a importância do uso de técnicas específicas capazes de extrair toda a potencialidade proporcionada pelo uso de informações espaciais.

As funcionalidades e a estrutura de um *data warehouse* são bastante adequadas à estruturação, tratamento e análise de informações espaciais, pois usualmente tais informações necessitam de um grande volume de armazenagem, demandando, por conseguinte, um alto consumo de processamento. Assim sendo, o caráter "desnormalizado" de um *data warehouse*, obtido principalmente pela utilização de modelos multidimendionais em formato estrela, melhora a velocidade de recuperação de informações. Isto pode ser evidenciado na aplicação-piloto desenvolvida (*Ranking* Acadêmico Mundial de Universidades), que demonstram uma boa velocidade de recuperação de informação, inclusive em ambiente web, no qual naturalmente a velocidade é limitada pelas características da conexão internet. Além disso, operações usuais em cubos de *data marts*, como *roll up, drill down, slice* e *dice* são bastante adequadas à navegação entre os diversos níveis de agregação espacial.

Os aplicativos de análise espacial desenvolvidos deixam claro como técnicas de análise espacial contribuem na extração de fatos relevantes a partir de dados espaciais. Alguns exemplos que evidenciam isto são os seguintes:

- O uso da análise de autocorrelação espacial mostra que as regiões de *hot spots* (*High-High*), realmente correspondem às regiões mais ricas do estado de Minas Gerais(região central, ao redor de Belo Horizonte e Triângulo Mineiro), e que as regiões *low-low*, como era de se esperar, correspondem ao norte-nordeste de minas, mais pobres.

- Já o uso da análise de reticulados por meio da geometria táxi permite a visualização clara de regiões que atendem a critérios de distância, algo que mesmo intuitivamente torna-se difícil quando as relações a serem consideradas se tornam complexas.

- Por sua vez, o modelo tempo-espacial de Hägerstrand aumenta a capacidade de visualização de seqüências de eventos no espaço, vislumbrando-se com isso um bom campo de aplicação;

Conforme observado acima, o aplicativo para análise espacial em reticulados revela-se bastante útil, ao permitir a visualização de pontos e regiões no mapa que atendem a restrições de distância e proximidade, as quais podem se tornar complexas. Neste caso, o uso do construtor de expressões permite representar tal complexidade de maneira intuitiva, facilitando o trabalho do analista de dados. Como um número significativo de cidades possui topologia reticular em sua totalidade ou em boa parte de sua superfície, evidencia-se um bom campo de aplicação para este modelo. Uma limitação do aplicativo está na entrada de dados, que deve ser feita pela introdução das ruas por meio do tamanho dos quarteirões e das avenidas por meio das retas que a representam, processo que pode se tornar laborioso para uma área demasiadamente grande. Um importante melhoramento , portanto, consiste na utilização de mapas vetoriais para geração do reticulado, atualmente formado por quarteirões de dimensões fixas. Assim sendo, o usuário definiria uma área a ser importada ao aplicativo, a qual seria convertida nas estruturas de dados do mesmo.

O aplicativo para análise de autocorrelação pode ser estendido pela introdução de outros critérios para avaliação de vizinhança entre objetos geográficos. O critério atual considera a contigüidade física entre objetos geográficos. Podem-se

introduzir critérios de distância máxima ou mínima, em que objetos situados a uma distância maior ou menor que um raio pré-determinado a partir de um objeto específico, sejam considerados vizinhos.

Com relação ao aplicativo que implementa o modelo tempo-espacial de Hägerstrand, o mesmo realiza importação de dados em formato CSV/Excell, mas não exporta tais dados de sua representação interna para o referido formato, de forma que possam ser utilizados em outras aplicações.

O aplicativo para consulta em bases de dados multidimensionais revelou-se ser de boa usabilidade. Dois aspectos sobre isto devem ser ressaltados. Em primeiro lugar, a facilidade para a criação da estrutura de consultas que permite, uma vez criado o modelo multidimensional, rapidamente gerar os dados para população das tabelas. O desenvolvimento da aplicação-piloto (*ranking* da Universidade de Shanghai) foi facilitado com o uso desta ferramenta, a partir do modelo multidimensional da mesma. Em segundo lugar, deve-se destacar a facilidade para geração de consultas (*queries*) sobre o modelo multidimensional. O resultado de tais consultas, quando exibido na forma de mapas, muitas vezes evidencia fatos relevantes, os quais nem sempre são visualizáveis por meios convencionais. Dentre as possíveis limitações e possíveis melhoramentos a serem realizados, podemos citar:

- A introdução de recursos de *join* de forma a permitir a conexão entre tabelas de fatos e dimensões.
- A criação de novos critérios para definição de faixas de legenda. O critério atual implementa faixas de mesmo tamanho. Propõe-se a criação de critérios de contagem igual de instâncias, de divisões por desvios-padrões e de divisões por quartis.
- Recursos para formatação personalizada de gráficos e relatórios.

O desenvolvimento da aplicação-piloto demonstrou a utilização do aplicativo para importação de dados, em especial, importação de dados provenientes de planilhas

Excell e de mapas em formato MIF/Mapinfo. Observou-se que as APIs especificadas e desenvolvidas para tal (respectivamente ISBExtrairCSV, ISBExtrarirMIF) são bastante adequadas e permitem uma boa integração, rápida e precisa, entre dados convencionais (neste caso, provenientes das planilhas Excell) e dados espaciais (objetos geográficos provenientes de arquivos MIF/Mapinfo).

A aplicação-piloto tem sido continuamente utilizada e aprimorada e a alta usabilidade de sua interface tem sido evidenciada por avaliações positivas de usuários, confirmando a adequação do *layout* padrão de aplicações proposto neste trabalho.

O processo de desenvolvimento proposto aborda as principais disciplinas necessárias ao desenvolvimento de sistemas para visualização, tratamento e análise de dados espaciais, no qual tais disciplinas são descritas em termos de suas atividades, estas em tarefas, que por sua vez dividem-se em passos e possuem artefatos que necessitam e produzem, além de papéis que as realizam. Um aspecto interessante a ser ressaltado diz respeito ao ambiente onde foi formalizado, o EPF(*Eclipse Process Framework)*. Seus recursos de *plugin*, herança e extensão de tarefas e atividades são muito úteis a uma atividade bastante usual em Engenharia de Software, a denominada *adequação*. Esta atividade estabelece critérios financeiros, tecnológicos, de recursos ou de características desejáveis que serão a base para definição de quais atividades, tarefas, papéis ou artefatos devem ser mantidos ou excluídos do processo, de forma a adequá-lo às características de um projeto específico ou de um ambiente de desenvolvimento. Um notável melhoramento futuro, seria a adequação e adaptação do processo proposto ao MPS-Br, Modelo Processo de Software Brasileiro. O MPS-Br é um constituído por um conjunto de processos e níveis que revelam o grau de maturidade de um processo em um organização. Esta adaptação implica essencialmente na aderência das atividades e tarefas definidas nesta proposta aos resultados de processos do MPS-Br, de forma a criarem-se evidências de sua implementação, e na criação de novas disciplinas exigidas pelos diversos níveis de capacidade do MPS-Br.

Uma limitação da ferramenta EPF é que para o processo nela definido, pode-se gerar apenas o *web site* com informações do mesmo. Em muitas situações, um relatório em formato *Ms Word* pode ser necessário, como na documentação da metodologia utilizada por uma organização. Entretanto, a ferramenta EPF permite a exportação do processo em formato XML. Assim sendo, propõe-se a implementação de uma aplicativo para geração de relatórios em formato *Ms Word,* a partir do arquivo XML gerado.

A utilização da arquitetura de sofware pôde ser demostrada por meio do desenvolvimento dos aplicativos e sobre ela podemos destacar o seguinte:

- A facilidade de implementação das interfaces ISB, que se revelaram suficientes e adequadas aos modelos de domínio do problema.

- A constatação da eficiência do uso do padrão de projeto *adapter* (Gamma, 1998) nas ISBs, ou seja, a independência entre a interface e sua implementação, abrindo a possibilidade de uso de novas bibliotecas no futuro.

- O uso do arcabouço Ecore do EMF, em especial, aumenta a produtividade da implementação, por padronizar e consequentemente facilitar a codificação de rotinas para recuperação e armazenamento das classes pertencentes a estes modelos de domínio de problema.

- A topologia da proposta, em camadas, permite o desenvolvimento estruturado da aplicação, inclusive por equipes de desenvolvimento independentes, trabalhando em paralelo, quando houver recursos de pessoal suficientes para tal. Isto acontece porque tais camadas estão semanticamente bem definidas e funcionalmente delimitadas entre si.

- A camada de interface incorpora objetos de interface que se conectam a classes adaptadoras do tipo ISB (*Interface Session Bean)* por meio de *servlets* ou *web services.* Os ISBs foram desenvolvidos para incorporar somente as regras de negócio requeridas e sua interface conecta-se a classes de domínio de problema na camada de persistência.

- Pode-se inferir, por conseguinte, que proporciona alto grau de componentização e modularidade ao processo de desenvolvimento, o que se traduz também em maior nível de coesão de cada classe ou interface nele existente e, na prática, em melhor qualidade do código produzido, maior velocidade de desenvolvimento e maior liberdade de criação ao analista, que pode se concentrar em atividades de nível mais baixo de abstração, mais próximo ao mundo real.

- Sua flexibilidade, que permite a criação de novos componentes funcionais por meio de *session beans,* aumentando-se consequentemente a capacidade de extensão do mesmo em função de novas necessidades de análise de dados convencionais ou espaciais.

Algumas limitações, possíveis implementações e melhoramentos propostos para a arquitetura de software podem ser enumerados:

- Trata-se de uma estrutura desenvolvida para trabalhar apenas com tecnologia Java. Entretanto, sua modularidade e transparência de interfaces tornam a tarefa de adaptação a outras tecnologias (.net, por exemplo) bastante facilitada.

- No que tange à utilização da tecnologia AJAX, componentes de interface melhor adaptados à mesma podem ser desenvolvidos e uma melhor integração com a tecnologia JSF pode ser realizada. Para tal, sugere-se a utilização da tecnologia *Seam Remoting API* da IBM, que facilita a comunicação de componentes AJAX com os *backing beans* do JSF.

- Especificar interfaces para outros serviços OGC além de WMS, WFS e WCS.

- A instalação em produção dos diversos componentes de software das camadas arquiteturais é feita manualmente e é laboriosa. Propõe-se a implementação de uma ferramenta que automatize e facilite esta tarefa.

- Intensificar o uso da API do *Gooble Maps*. Atualmente ela é utilizada nesta infraestrutura para visualização de mapas, através de uma interface WMS. Podem-se desenvolver interfaces para acesso a suas funcionalidades de roteamento e georreferenciamento de endereços.

- Especificar operações e estruturas para dados *raster,* não explorados nesta tese.

- O EMF automatiza uma parte do processo proposto pela geração de código Java para modelos de domínio de problema. Entretanto a proposta carece de ferramentas que automatizem a integração entre as camadas de apresentação e de controle e desta com a de persistência.

- Propõe-se a especificação e o desenvolvimento de uma linguagem gráfica, possivelmente estendida a partir da UML, para especificação de domínios e metamodelos da arquitetura proposta.

- A infraestrutura não especifica ou implementa qualquer medida de segurança. Um melhoramento futuro deverá explorar serviços de segurança definidos pelo JEE.

- Especificar interfaces para métodos estatísticos e de análise multivariada.

- Criar extensões geográficas para outros pacotes do CWM. Foram realizadas extensões apenas para os pacotes *data mining* e OLAP, por serem as de maior utilização na estrutura proposta e nas APIs.

Finalmente, ressalta-se que alguns dos resultados desta tese foram publicados em anais de congressos internacionais recentes (Abreu e Miranda Jr, 2007a, 2007b, e 2009), (Miranda Jr. e Abreu, 2008).

Referências

ABREU, J.F; BARROSO, L.C.,. Alguns Aspectos da Geometria Táxi na Geografia. *In Revista Geografia e Ensino*. São Paulo: março de 1982.

ABREU, J.F & MUZZARELLI, A.. *Introduzioni ai Sistemi Informativi Geografici.* Bologna: Edilio, 1ª edição,2003.

ABREU, J.F, MIRANDA, D.F, BARROSO, L.C. Das Cidades à Geometria Táxi. *In Geografia, Modelos de Análise Espacial e GIS*. Belo Horizonte: Editora PUC Minas, 2004.

ABREU, J. F. 2005. *Notas de Aula da Disciplina Métodos de Análise Espacial.* Belo Horizonte:PUC Minas, 2005.

ABREU, J. F. ; MIRANDA JUNIOR,P.O . Mapping Geographical Voices,with an Application of Hägerstrand´s Space-Time Model. In: *2007 -AAG Meeting, 2007, S.Francisco. 2007 Annual Meeting*. Washington : AAG, 2007a. v. 1.

ABREU, J. F. ; MIRANDA JUNIOR,P.O . Mapping the World Rankings of Universities -Using Geographic Information Systems and Multi-Dimensional Scales. *In: IREG 3, 2007, SHANGHAI. Proceedings of the Third Meeting of the International Ranking Expert Group(IREG 3)*. Shanghai : Shanghai Jiao Tong University, 2007b. v. 1. p. 297-307.

ABREU, J. F. ; MIRANDA JUNIOR,P.O . Geo-Representation of The World Ranking Universities Using GIS and Multi Dimensional Scale. *In: AAG Meeeeting 2009, 2009*, Las Vegas. Annals 2009 AAG Annual Meeting. Washington DC : AG, 2009. v. 1.

ADAMS, P.C. *Application of a CAD-Based Accessibility Model.* New York, 1996.

AGRAWAL , R. Fast Algorithms for Mining Association Rules in *Proceedings of the 20th VLDB Conference,* Santiago: VLDB Conference, 1994.

AMORIM FILHO, Oswaldo Bueno; RIGOTTI, José Irineu Rangel; CAMPOS, Jarvis. *Os níveis hierárquicos das cidades médias de Minas Gerais.* Belo Horizonte.: Pontifícia Universidade Católica de Minas Gerais, Programa de Pós-Graduação em Tratamento da Informação Espacial, 21 p., 2º sem. 2006.

ANSELIN, L. Local Indicators of Spatial Association – LISA. *Geographical Analysis.* Vol. 27 No 2, april 1995.

ASSUNÇÃO, R.M.; LAGE, J.P.; A.REIS, E.; SILVA, P.L.N. *Análise de conglomerados espaciais via árvore geradora mínima*. Belo Horizonte: Departamento de Estatística da UFMG, 2000.

ASSUNÇÃO, R.M, et al*; Procedimentos Automáticos e semi-automáticos de Regionalização por Árvore Geradora Mínima;* Belo Horizonte: Departamento de Estatística da UFMG, 2003.

AZEVEDO,L.G. et al. Approximate Spatial Query Processing Using Raster Signatures. *In Braziliam Symposium of Geoinformatics*. São José dos Campos: InPE, 2005.

BAÇÃO, F. et ali The Self-Organizing Map and it's variants as tools for geodemographical data analysis: the case of Lisbon's Metropolitan Area. In *7th AGILE Conference on Geographic Information Science,* Heraklion, Grécia: 2004a.

BAÇÃO, F. et ali The Self-Organizing Map,The Geo-SOM, and relevant variants for geosciences.In *Computer Geosciences.* Berlin: Springer, 2004b.

BAÇÃO, F. et ali. *Geo-SOM and Its Integration with Geographic Information Systems*. In WSOM 2005, Paris: 2005.

BARQUIN, R. *Planning and Designing the Data Warehouse.* New Jersey: Prentice-Hall, 1996.

BENSON, A. et al. "Data Warehousing, Data Mining, and OLAP". San Francisco: McGraw-Hill, 1997.

BOGORNY, V. *Enhancing Spatial Association Rule Mining in Geographic Databases.* Tese de Doutorado. Porto Alegre: UFRGS, 2006.

BOOCH, G. et al. *UML – Guia do Usuário.* 2a Edição, Rio de Janeiro: Elsevier, 2003.

BRAGA, A. *Um Arcabouço para Implementação de Data Warehouses Estendidos com Tipos Abstratos de Dados.* Tese de Doutorado. Rio de Janeiro: PUC Rio, 2004.

BRINKHOFF, K. et al. Efficient Processing of Spatial Joins Using R-Trees. IN *In ACM Sigmoid International Conference on Management of Data*, Washington D.C.: 1994, p. 237-246

BUDINSKY, F. et al. *Eclipse Modeling Framework.* 1a edição. New York: Addison Wesley, 2003.

CÂMARA, G. et al. *Bancos de Dados Geográficos.* São José dos Campos: Publicação do INPE, 1999.

CÂMARA, G. et al. *Mineração de Dados em Grandes Bancos de Dados Geográficos.* Relatório Técnico. São José dos Campos: INPE, 2001.

CAMPOS, M.L. *Data warehouse.* Minicurso do Braziian Symposium on Databases. Brasília: 2000.

CANTELLI, R. *Tecnologias Aplicadas à Inteligência do Negócio-Notas de Aula.* Belo Horizonte: IEC-PUC Minas, 2006.

CAPEL, H. & URTEAGA,Luis. *Las Nuevas Geografias.* Madrid: Salvat, 1987.

CHRISTOFOLETTI, A. As Características da Nova Geografia. *In Geografia,* 1(1). Rio de Janeiro:1985.

CYBENKO, G. *Continuous Valued Neural Networks with Two Hidden Layers are Suficient.* Technical Report, Department of Computer Science, Boston: Tufts University, 1988.

CYBENKO, G. Aproximation by Superpositions of a Sigmoid Function. In *Mathematics of Control, Signals and Systems.* New York: *1989*

DAVIS, C. *Arquitetura de Bancos de Dados Geográficos.* São José dos Campos: InPE, 2000.

DAVIS, C. et al. *O Open Geospatial Consortium.* São José dos Campos: Inpe, 2005.

DEMPSTER, A.P. – Maximum Likelihood From Incomplete Data via The EM Algorithm – *Journal of the Royal Statistical Society .* Vol.39, No 1, 1977, pp 1-38

DESCARTES, R. *O Discurso do Método*. São Paulo: Editora Abril, 1972.

DEVLIN, B. *Data Warehouse: from architecture to implementation*. New York:Addison Wesley Longman,Inc., 1997.

FAN, B. et al. Spatial Data Mining Method For Costumer Intelligence. in *Proceedings of the Second International Conference on Machine Learning and Cybernetics.* IEEE, 2003.

FIDALGO, R. *Uma Infraestrutura Para Integração de Modelos, Esquemas e Serviços Multidimensionais e Geográficos.* Tese de Doutorado. Recife, UFPE, 2005.

FISHER, H.F. - Knowledge Acquisition Via Incremental Conceptual Clustering- *Machine Learning.* Boston: Kluwer Academic Publihsers, 1987.

FORNARI, M.R. *Análise e Desenvolvimento de Um Novo Algoritmo de Junção Espacial para SGBDs Geográficos.* Tese de Doutorado. Porto Alegre UFRGS, 2006.

GAMMA, E. et Al. *Design Patterns: Elements of Reusable Object-Oriented Software.* New York, Addison-Wesley, 1998.

GORDON, A.D. *Classification Methods for the Exploratory Analysis of Multivariate Data.* 1ed. London: Chapmam and Hall, 1996

GOULD, P. *The Geographer at Work.* London: Routledge & Keegan Paul,1985.

HARINARAYAN, V. et al. Implementing Data Cubes Efficiently. In. *Proceeding s of ACM SIGMOD'96.* Montreal:1996.

HARTSHORN, R. Perspective on Nature of Geography. In: *Monograph Series of Association of American Geographers*. Chicago: Rand Mcnally, 1992

HINTON, E. & STEPHEN, J. How Learning Can Guide Evolution. In: *Complex Systems.* Vol. 1, 495-502, 1987.

IBM-International Businness Machines. *The Unified Process.* Disponível em http://www.wthreex.com/rup/portugues/index.htm., 2003.

IBM-International Businness Machines. *The Eclipse Process Framework.* Disponível em http://www.eclipse.org/epf/., 2006.

IBM-International Businness Machines. *The Eclipse GUI.* Disponível em http://www.eclipse.org, 2009.

IBM-International Businness Machines. *The Eclipse Modeling Framework.* Disponível em http://www.eclipse.org/modeling/emf/?project=teneo, 2009b.

ICHIKI, H. et al. In *IJCNN'91, International Conference on Neural Networks*. Houston: pp. 357-360. 1991.

INMON W.H. *Building the Data Warehouse.* New York: John Wiley & Sons; 1992.

JOHNSTON, D.J. *Geografia e Geógrafos*. São Paulo: DIFEL, 1986.

JONG, K. Adaptative Systems Design. In *IEEE Transactions On Systems, Man and Cybernetics.* Dez,1980

KANGAS, J. A. Variants of Self-Organizing Maps.In *IEEE Transactions on Neural Networks* .1(1): 93-99, april, 1990.

KAPLAN, R. & NORTON, R. *A Estratégia em Ação-Balanced Scoregard.* New York: Wiley,2003

KAUFMAN,L. & P.J. ROUSUEEUW. *Finding Groups in Data: an Introduction to Cluster Analysis.* New York: John Wiley & Sons,1990.

KIMBALL, R. *The Data Warehouse Lifecycle Toolkit.* New York: Wiley, 1998.

KWAN, M. *Human Extensibility and Individual Hybrid-Accessibility in Space-Time: a Multi-Scale Representation Using GIS.*.Boston: 1995.

KIRKPATRICK, D. and SNOEYINK, J. Tentative Prune-and-Search for Computing Fixed-Points with Applications to Geometric Computation, in *Proc. Ninth ACM Symp. Computational Geometry*. Los Angeles: 1993.

KOHONEN, T. *Artificial Neural Networks*. Boston: Elsevier Science Publishers, 1991.

KOHONEN, T. *Sef-Organizing Maps.* Berlin: Springer-Verlag, 2001.

KOPERSKI, K. and HAN, J. Discovery of Spatial Association Rules in Geographic Information Databases In: *International Symposium on Large Geographical Databases.* Portland: 1995.

KRIEGEL, H.P. Spatial Query Processing for High Resolution. In *International Conference of Database Systems for Advanced Applications*. Tokyo: IEEE Press, 2003. p.17-26

KROLL, P. *OpenUp in a NutShell.* Disponível em http://www.ibm.com/developerworks/rational/library/sep07/kroll/, 2007.

KWAN, M. *Human Extensibility and Individual Hybrid-Accessibility in Space-Time: a Multi-Scale Representation Using GIS.* Boston *1995.*

LAKATOS, I. *Criticism and the Growth of Knowledge*. Cambridge: Cambridge University Press, 1970.

MIRANDA JUNIOR,P.O & ABREU, J.F.. Urban Hierarchization with Classification using GIS, Multidimensional Scaling and Neural Network Analysis. In: *Proceedings of RSAI World Congress 2008,* S.Paulo : FIPE/USP, 2008.

NG, R.T.; HAN, J. Efficient and effective clustering methods for Spatial Data Mining. *In:Twentieth International Conference on Very Large Data Base,* Santiago: 1994..

NG, R.T.; HAN, J. CLARANS: A Method for Clustering Objects for Spatial Data Mining *in IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, VOL. 14, NO. 5, 2002.

OGC-Open Geospatial Consortium.*OGC Reference Model.* 2003.

OGC-Open Geospatial Consortium. *Web Map Service Interface.* Versão 1.3. 2004.

OGC-Open Geospatial Consortium.*OGC Web Feature Service Implementation Specification.* Versão 1.1.0.OGC, 2005.

OGC-Open Geospatial Consortium.*The OpenGIS Abstract Specification – Topic 6: The Coverage Type and its Subtypes*. 2006a.

OGC-Open Geospatial Consortium.*OGC Web Map Service Implementation Specification.* Versão 1.3.0.OGC, 2006b.

OMG-Object Management Group. *Common Warehouse Metamodel (CWM) Specification.* Version 1.1, Volume 1, 2003.

OPENSHAW, S. Developing appropriate spatial analysis methods for GIS, In: *Geographical Information Systems Vol 1 Principles*. Harlow: Longman Scientific & Technical, pp 389-402, 1991.

ORENSTEIN, J.A. Spatial Query Processing. in Object Oriented Database Systems. *In ACM Sigmoid International Conference on Management of Data*, New York: 1996, p. 326-336

PADUA, A. et al. Redes Neurais Artificiais-Teoria e Aplicações. Rio de Janeiro, LTC, 2000.

PASQUIER, N *et al*. Efficient Mining of Association Rules Using Closed Itemsets Lattices. *Information Systems,* v.24, n.1, p.25-46 , Mar, 1999.

POOLE, J. et al. *Commom Warehouse Metamodel Developers Guide.*New York: Wiley Publishing Inc. , 1a edição, 2003.

POOLE, J. et al. *Common Warehouse Metamodel -An Introduction to the Standard For Data Warehouse Integration.* 1a edição. New York: Wiley, 2002.

REED, R. Pruning Algorithms- A Survey . *in IEEE Transactions on Neural Networks.* Vol 4, pp 740-746, 1993.

RICHARDS, J.A*. Remote Sensing Digital Image Analysis - An Introduction.* 3 ed. Berlin:Springer-Verlag, 1995.

RUSSEL, S. & NORVIG, P. *Inteligência Artificial.* Rio de Janeiro: Elsevier, 2ª edição*,* 2004.

SAMPAIO, M.C. et al. Towards a Logical Multidimensional Model for Spatial Data Warehousing Using OLAP. In *Proceedings od DOLAP'06.* Arlington, USA: 2006.

SCHALKOFF, R. *Artificial Neural Networks.* 1a Edição*.* San Francisco: Mc Graw-Hill, 1997.

STEFANOVIC, N. *Design and Implementation of On-Line Analytical Processing of Spatial Data.* Dissertação de Mestrado. Belgrado: Universidade de Belgrado, 1997.

Sun Microsystems. *The Java EE 5 Tutorial.* 1a edição, Sunny Valle, USA: 2007.

TOBLER, W. A Computer Model Simulating Urban Growth in the Detroit Region, in *Economic Geography*, vol. 46, pp. 234-240, 1970.

WISE, S.; HAINING, R.; MA, J. Regionalization Tools for The Exploratory Spatial Analysis od Health Data. In: FISCHER, M.M.; GETIS, A., Eds., *Recent Developments in Spatial Analysis: Spatial Statistics, Behavioural Modelling, and Computational Intelligence*. Berlin: Springer, 1997.

WITTEN, I.H & FRANK, E. *Data Mining – Practical Machine Learning Tools and Techniques.* San Franscisco: Elsevier, 2a Edição, 2005.

APÊNDICE A – Detalhamento do Processo de Desenvolvimento

Neste apêndice apresentam-se telas do *web site* gerado pela ferramenta EPF. As atividades de cada disciplina do processo PADE são detalhadas. Para cada disciplina apresenta-se seu diagrama de atividades e para cada atividade exibe-se um diagrama contendo todas as suas tarefas, com todos os artefatos de entrada e saída da mesma e os papéis responsáveis por sua realização. Em cada tarefa exibem-se também os passos que a implementam.

A.1-Etapa de Concepção

O principal objetivo da fase de concepção é atingir o consenso entre todos os envolvidos sobre os objetivos do ciclo de vida do projeto (IBM, 2003). Esta fase concentra tarefas das disciplinas *Planejamento* e *Requisitos,* descritas a seguir.

A.1.1-A disciplina Planejamento

Esta disciplina envolve atividades e tarefas relacionadas ao planejamento do projeto em todas as suas etapas.

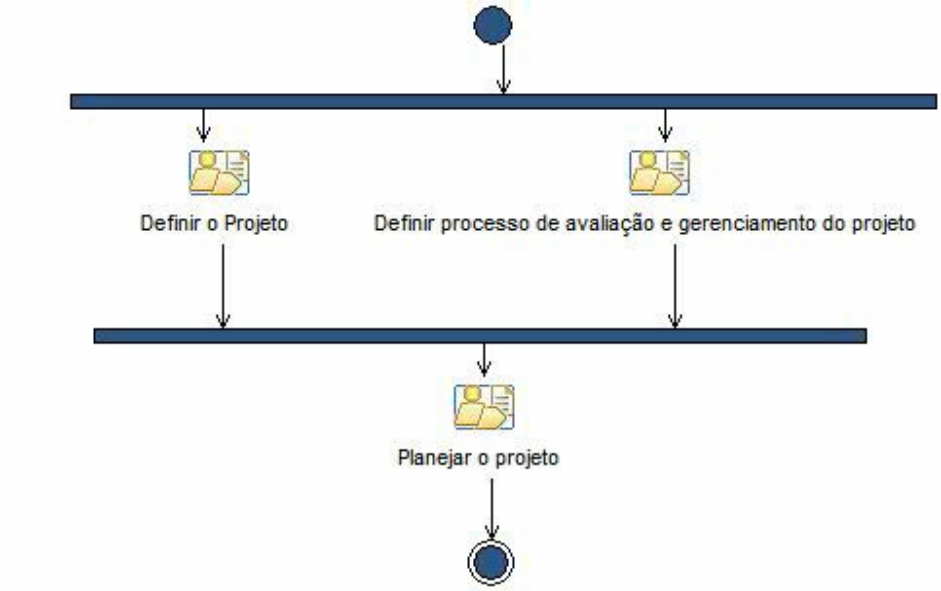


Figura 108-Diagrama de atividades da disciplina Planejamento

A.1.1.1-Atividade definir o projeto

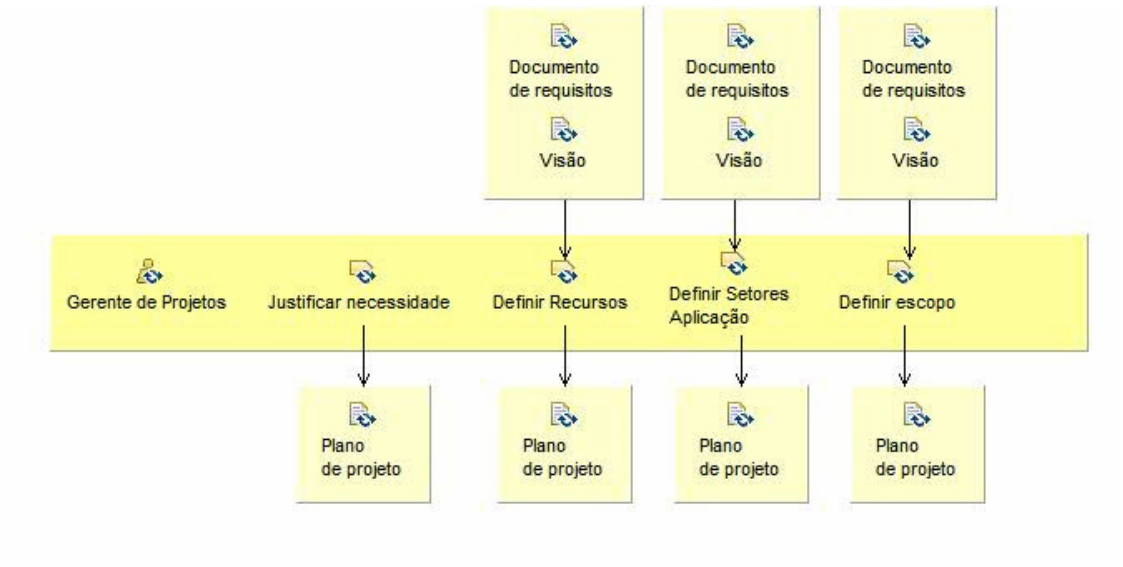


Figura 109-Atividade definir projeto

Justificar necessidade

⊟ Justificar necessidade de desenvolvimento
Justificar junto à direção, a necessidade de desenvolvimento do presente proje

Definir recursos

⊟ Definir recursos de pessoal
Definir todos os recursos de pessoal necessários à realização do proj
⊟ Definir recursos de hardware
Definir todos os recursos de hardware necessários à realização do proj
⊟ Definir recursos de software
Definir todos os recursos de software necessários à realização do proj
⊟ Definir recursos financeiros
Definir todos os recursos financeiros totais necessários à realização do proje

Definir setores aplicação
⊟ Definir setores aplicação
Definir todos os setores que serão afetados pela realização do proje

Definir escopo
⊟ Estabelecer domínio do projeto
Estabelecer o domínio de abrangência do que vai ser desenvolvido de forma a

atingir os objetivos definidos do projeto.
⊟ Identificar produtos de trabalho
Identificar produtos de trabalho que serão requerid
⊟ Identificar produtos de trabalho que serão reutilizados
Identificar produtos de trabalho que serão reutilizados, provenientes de projetos anteriores.

A.1.1.2-Atividade planejar o projeto

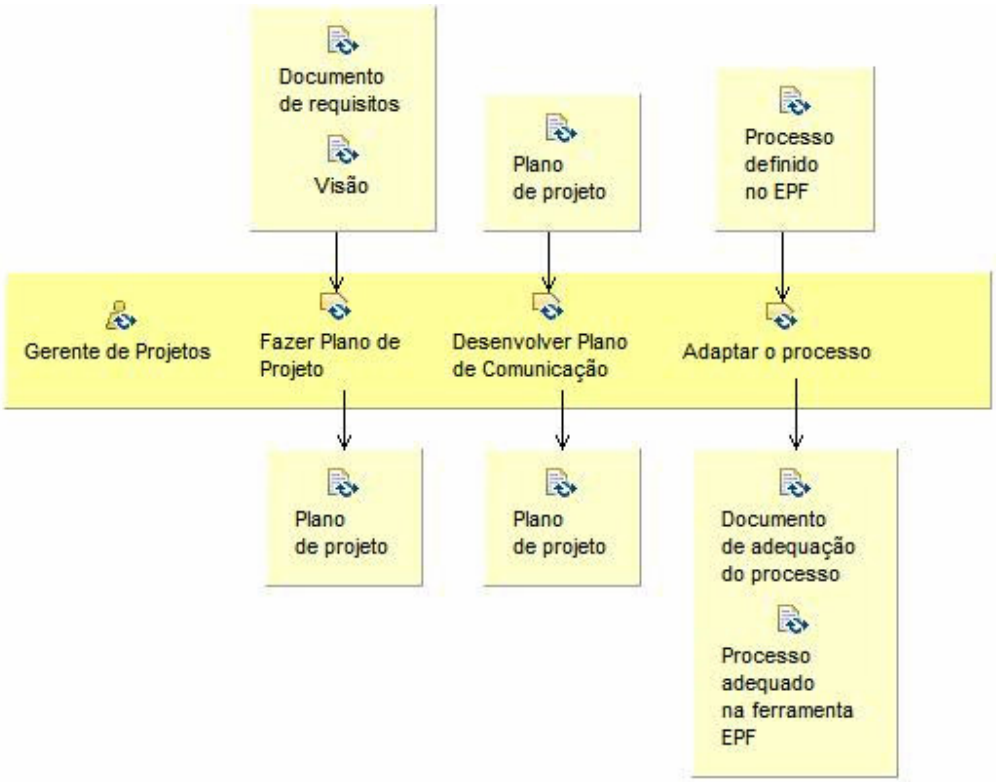


Figura 110-- Atividade planejar o projeto

Fazer plano de projeto

⊟ Estabelecer uma equipe coesa
Determinar cada papel em cada atividade do proje
⊟ Estimar tamanho
Produzir estimativas de tamanho para cada produto de trabal
⊟ Avaliar riscos
A equipe identifica os riscos do projeto e realiza uma análise quantitativa de risco para avaliar sua ordem de magnitude.
⊟ Prever duração do projeto
Definir o tamanho da iteração e usá-la para estimar a velocidade de

desenvolvimento.
Definir iterações
Definir iterações em um conjunto de fas
Estimar custos
Estimar todos os custos dos recursos envolvidos no proje
Planejar instalação e entrega
Planejar instalação e entrega do software no ambiente de produç

Desenvolver plano de comunicação

Estabelecer sequência de comunincação
Estabelecer sequência hierárquica de comunicação entre os envolvidos no projeto.

Estabelecer meios de comunicação
Estabelecer as formas pelas quais a comunicação será efetuada.

Adaptar o Processo

Definir disciplinas a serem utilizadas
Definir disciplinas a serem mantidas e excluídas no processo.

Definir atividades a serem utilizadas
Definir atividades a serem mantidas ou excluídas no projeto.

DefInir tarefas a serem utilizadas
Defnir tarefas a serem mantidas ou excluídas no projeto
Definir passos a serem utilizados
Definir passos a serem mantidos ou excluídos do projeto.

Definir papéis a serem utilizados
Definir papéis a serem mantidos e excluídos no projeto.
Definir produtos de trabalho a serem utilizados
Definir produtos de trabalho a serem mantidos ou excluídos do projeto.

Fazer adaptação via EPF
Utilizar os recursos da ferramenta EPF para definir o processo adaptado, com os componentes de processo mantidos nos passos anteriores.

A.1.1.3-Atividade definir processo de avaliação e gerenciamento do projeto

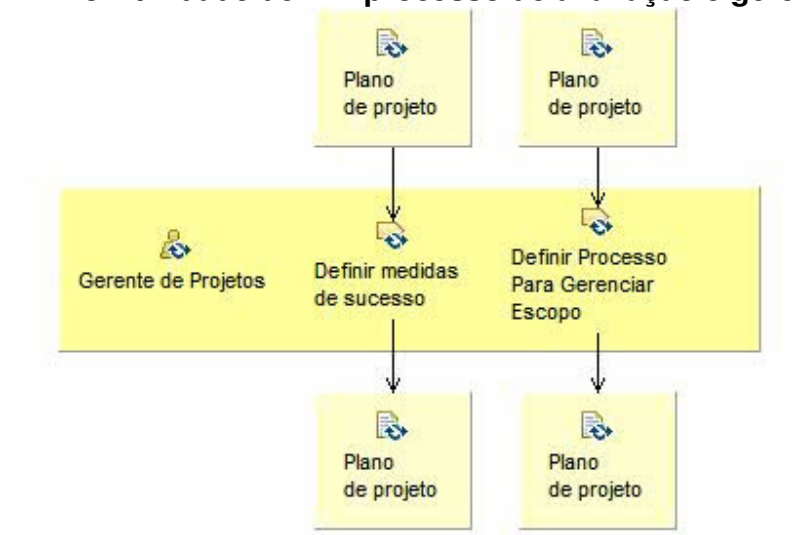


Figura 111- Atvidade definir processo de avaliação e gerenciamento do projeto

Definir medidas de sucesso

⊟ Estabelecer medidas de sucesso
Estabelecer as medidas pelas quais o sucesso do projeto será auferido.

Definir processo para gerenciar escopo

⊟ Definir envolvidos no processo
Definir envolvidos no processo de gerenciamento de escopo.
⊟ Definir o processo de gerenciamento de escopo
Definir as formas pelas quais o gerenciamento do escopo será efetuado.

A.1.2-A disciplina Requisitos

Esta disciplina envolve atividades e tarefas para elicitar, analisar,especificar, validar e gerenciar requisitos.

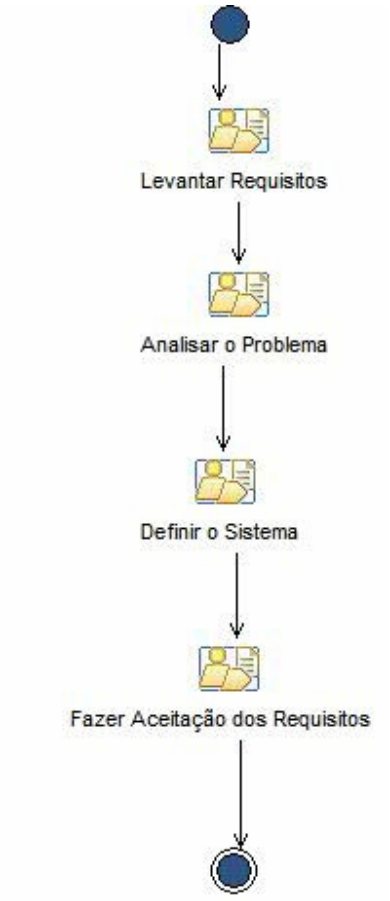


Figura 112-Diagrama de atividades da disciplina Requisitos

A.1.2.1-Atividade Analisar o problema

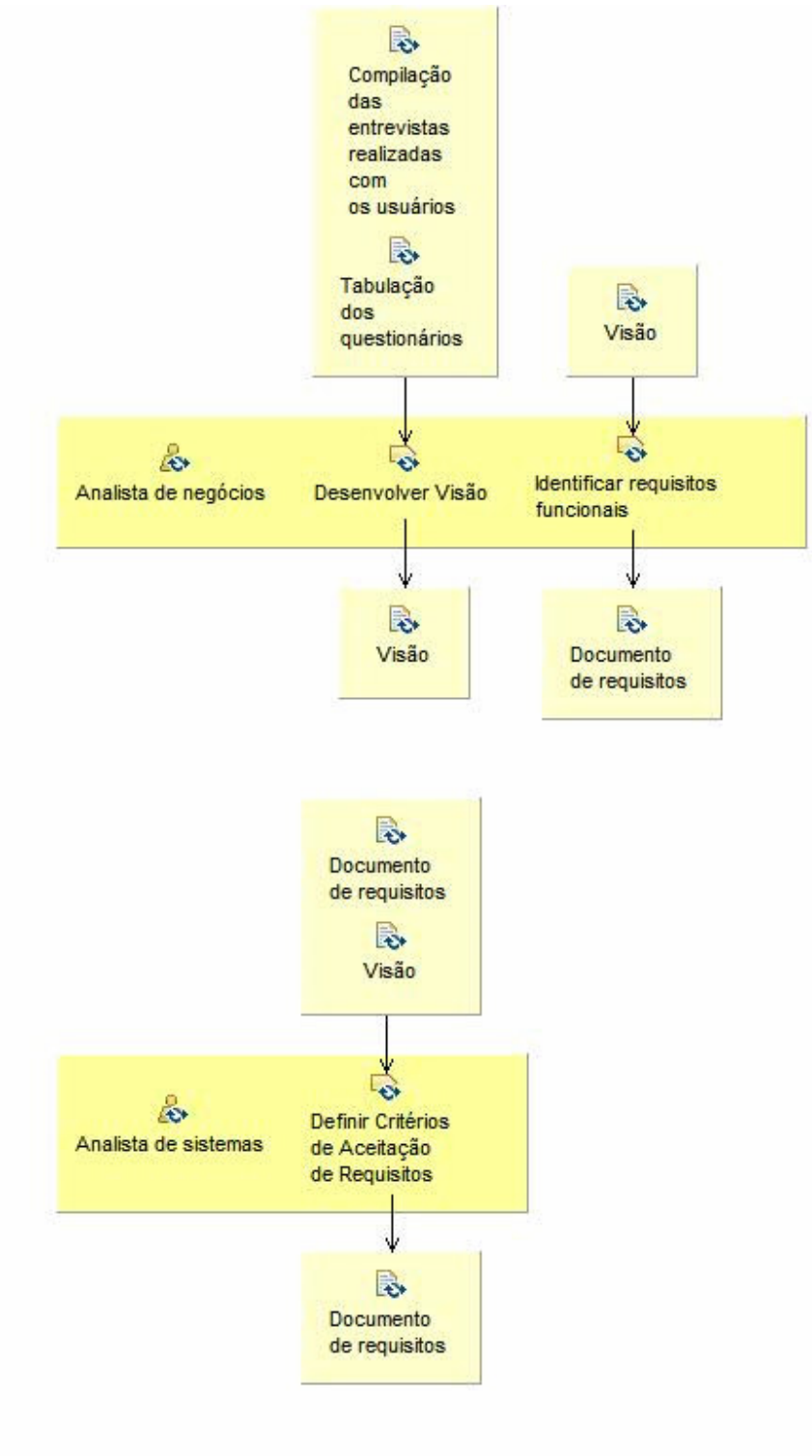


Figura 113- Atividade Analisar o problema

Desenvolver visão

- Estabelecer Objetivos de Análise

Esta tarefa consiste em determinar os objetivos da gerência em relação às análises a serem feitas pelo sistema de datawarehouse a ser implementado.

- Identificar restrições

Definem-se restrições a serem consideradas, como por exemplo, de ordem econômica, orçamentária, jurídicas, estatutárias, de mão-de-obra, gargalos na obtenção de informações, dentre outras

- Identificar capacidades analíticas desejáveis

Identificar as capacidades de análise desejadas de forma a se atingirem os objetivos de análise.

Identificar requisitos funcionais

- Identificar e capturar termos do domínio

Trabalhar próximo aos envolvidos para capturar termos usuais ao domínio do sistema.

- Identificar objetivos do negócio

Identificar os objetivos do negócio, que na realidade são a base da determinação das medidas realmente relevantes à análise do negócio.

- Identificar tipos de requisitos relevantes ao domínio do datawarehouse.

Identificar junto aos provedores de requisitos, requisitos de análise do negócio (por exemplo, fatos que auxiliem na análise de desempenho ou de produtividade), requisitos informacionais (requisitos acerca de dados relevantes ao sistema),

- Identificar possíveis associações de requisitos a feições espaciais

Este passo procura identificar como informações espaciais podem melhorar a análise de dados do negócio

- Identificar critérios de sucesso

Identificar critérios de sucesso, capazes de avaliar o sucesso do projeto, como por exemplo, métricas de implementação, métricas de nível de utilização, métricas de nível d serviço, métricas de impacto no negócio, métricas de desempenho antes e depois da implantação do *datawarehouse.*

- Definir necessidades de análise convencional e espacial

Com base nas necessidades e metas de gestão do negócio, definem-se os objetivos da análise de dados a ser feita (espaciais ou convencionais), de forma que sejam explicitadas todas as necessidades de análise (que posteriormente serão implementadas por métodos e ferramentas específicas).

Definir Critérios de Aceitação de Requisitos

- Definir critérios para aceitação de requsitos

Exemplos de critérios de aceitação: clareza , completude, consistência, unicidade,

adequabilidade à implementação, verificável, rastreável.

A.1.2.2-Atividade Definir o sistema

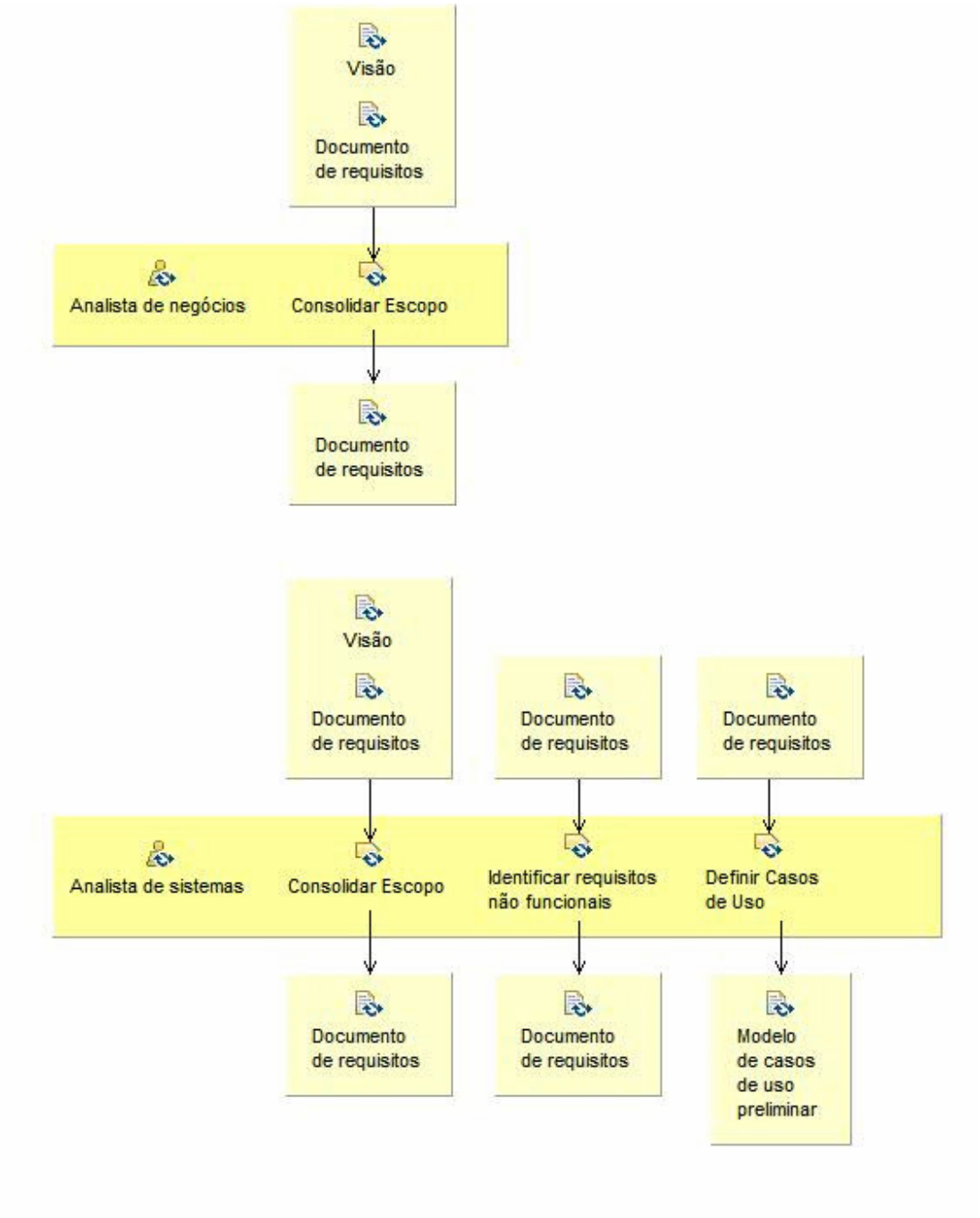


Figura 114- Atividade Definir o sistema

Consolidar escopo

⊟ Priorizar casos de uso e cenários
Definir a ordem das iterações sucessivas de casos de u
⊟ Documentar visão de casos de uso
Produzir uma listagem de casos de uso e cenários, documentando suas propriedades significativas, descrições e relacionamentos.
⊟ Fazer uma verificação do escopo
O escopo deve ser verificado de forma a certificar o bom andamento do trabal

Identificar requisitos não funcionais

⊟ Identificar requisitos de desempenho
Identificar requisitos ao desempenho da aplicação, como tempo de inicialização, tempo de resposta máximo de cada relatório, capacidades de armazenamento, dentre outros
⊟ Identificar requisitos de segurança
Identificar requisitos de segurança: restrições de acesso às informações, senhas, criptografia necessária, acesso via internet, dentre outros.
⊟ Identificar requisitos de recuperação
Identificar os tempos de resposta necessários à recuperação em caso de queda ou quebra do sistema.

Definir casos de uso

⊟ Determinar casos de uso
Casos de uso em sistemas *de* datawarehouse consistem essencialmente em respostas necessidades de análise do negócio definidas na etapa de requisitos. Normalmente, refletem-se em funcionalidades relacionadas à emissão de relatórios analíticos, realizaçã de operações de OLAP e ainda de manutenção e administração. Um arcabouço comum casos de uso analíticos, consiste em casos de uso relacionados a fatos presentes nos relatórios estendidos por casos de uso que refletem a análise sob o ponto de vista de determinada dimensão.
⊟ Desmembrar ou agregar casos de uso
Desmembrar casos de uso com alto grau de complexidade, ou agregar casos de uso que sejam simples e vinculáveis a outros.
⊟ Criar associações entre casos de uso
Criam-se, onde necessário, generalizações <EXTENDS> ou reutilizações <INCLUDE> de forma a tornar o diagrama mais robusto.
⊟ Criar descrições de alto nível.
Descrições de alto nível, contêm um texto narrativo sucinto que descreve o processo qual se referem. Constituem uma primeira abordagem acerca da funcionalidade do caso de uso, que servirá como base para refinamentos efetuados em etapas posteriores (casos de uso expandidos e reais).
⊟ Definir glossário
Definir junto com os envolvidos, o vocabulário utilizado no domínio sob análise.

Verificar a ocorrência de termos ambíguos ou impróprios.

A.1.2.3-Atividade Levantar requisitos

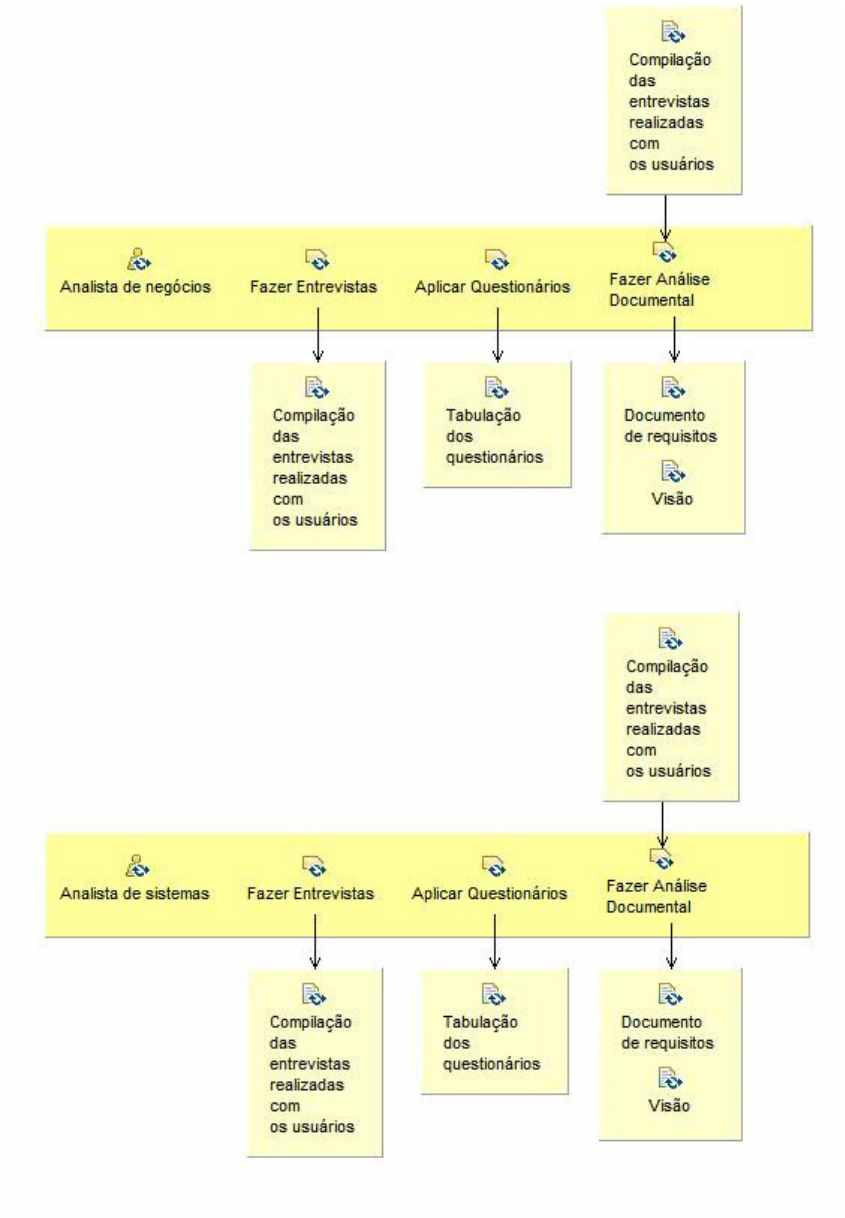


Figura 115- Atividade Levantar requisitos

Fazer entrevistas

⊟ Preparar as entrevistas
Definir os entrevistados (executivos, diretores e gerentes do negócio) , os entrevistadores, locais e tempo de duração.

⊟ Realizar as entrevistas
Realizar a entrevista abordando os seguintes tópicos:
Definir o objetivos da organização
Definir medidas de sucesso

Desafios do negócio
Como alavancar a qualidade das informações executivas na organização
Tipos de rotina de análise, como os dados para ela são obtidos e como são utilizados
Quais capacidades analíticas gostaria de possuir
Existência de gargalos na obtenção da informacão
Quanta informação histórica é necessária
Quais oportunidades existem para melhorar o negócio se houver melhoria nc acesso à informação
Quais relatórios são mais usualmente utilizados.
Como informações geográficas podem auxiliar na melhoria da qualidade das informações obtidas.

⊟ Compilar as entrevistas
Fazer um relatório final, destancando os pontos chaves e principais obtidos na entrevista:

Objetivos do negócio
Requisitos analíticos
Requisitos informacionais
Destaque para informações que possam ser melhor visualizadas com o uso feições geográficas.
Critérios de sucesso

Aplicar questionários

⊟ Preparar questionário
Preparar questionário para obter informações relevantes dos envolvidos: que tipo de trabalho faz, informações que produz e recebe, o que usa para tomar decisões, quais a pessoas envolvidas em seu trabalho, informações que não tem mas que julga importante, dentre outras.

Fazer análise documental

⊟ Identificar documentos importantes
Identificar documentos relevantes ao estudo do ambiente .

Identificar formulários
Identificar todos os formulários utilizados no ambiente.

A.1.2.4-Atividade fazer aceitação dos requisitos

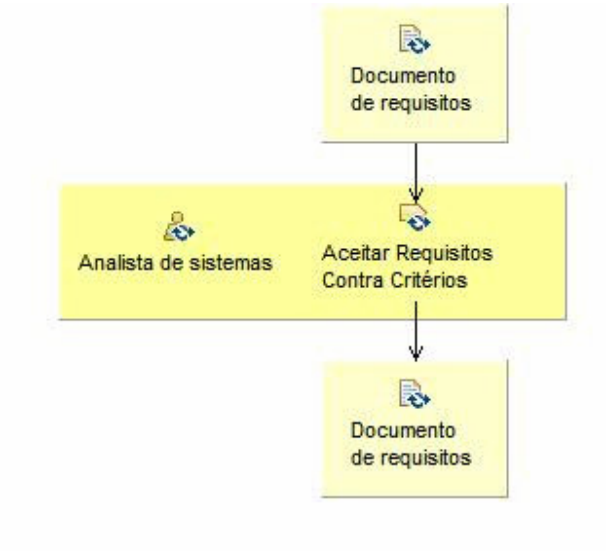


Figura 116- Atividade fazer aceitação dos requisitos

Aceitar Requisitos contra critérios

⊟ Estabelecer importância aos critérios
Determinar o grau de importância de cada critério, de forma que, quando da aceitação dos requisitos com o cliente, maior atenção seja dada aos mais

relevantes.

⊟ Verificar requisitos
Fazer a veriticação dos requisitos com o cliente, tendo como base os critérios de aceitação. Isto deve ser feito em tantas reuniões quantas sejam necessária para t
⊟ Formalizar aceitação de requisitos
Protocolar a aceitação de todos os requisitos pelo cliente em um documento de requisitos.

A.2-Etapa de elaboração

A etapa de elaboração envolve disciplinas relacionadas a análise de requisitos e projeto de toda a infraestrutura da aplicação.

A.2.1-A disciplina Análise e Especificação

A disciplina análise e especificação envolve atividades e tarefas de análise de requisitos, de forma a produzir uma especificação formal da aplicação.

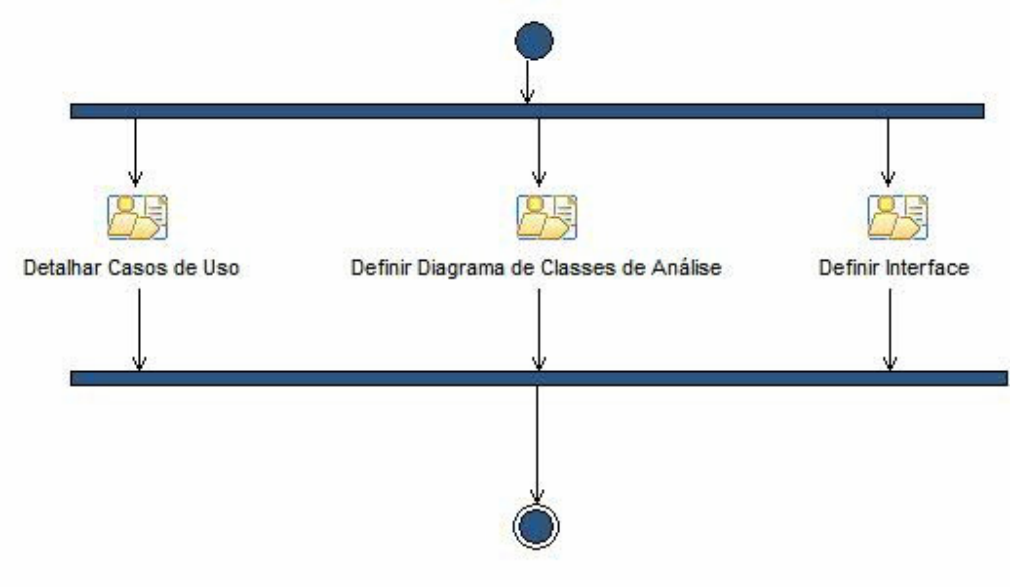


Figura 117-Diagrama de atividades da disciplina Análise e Especificação

A.2.1.1-Atividade definir interface

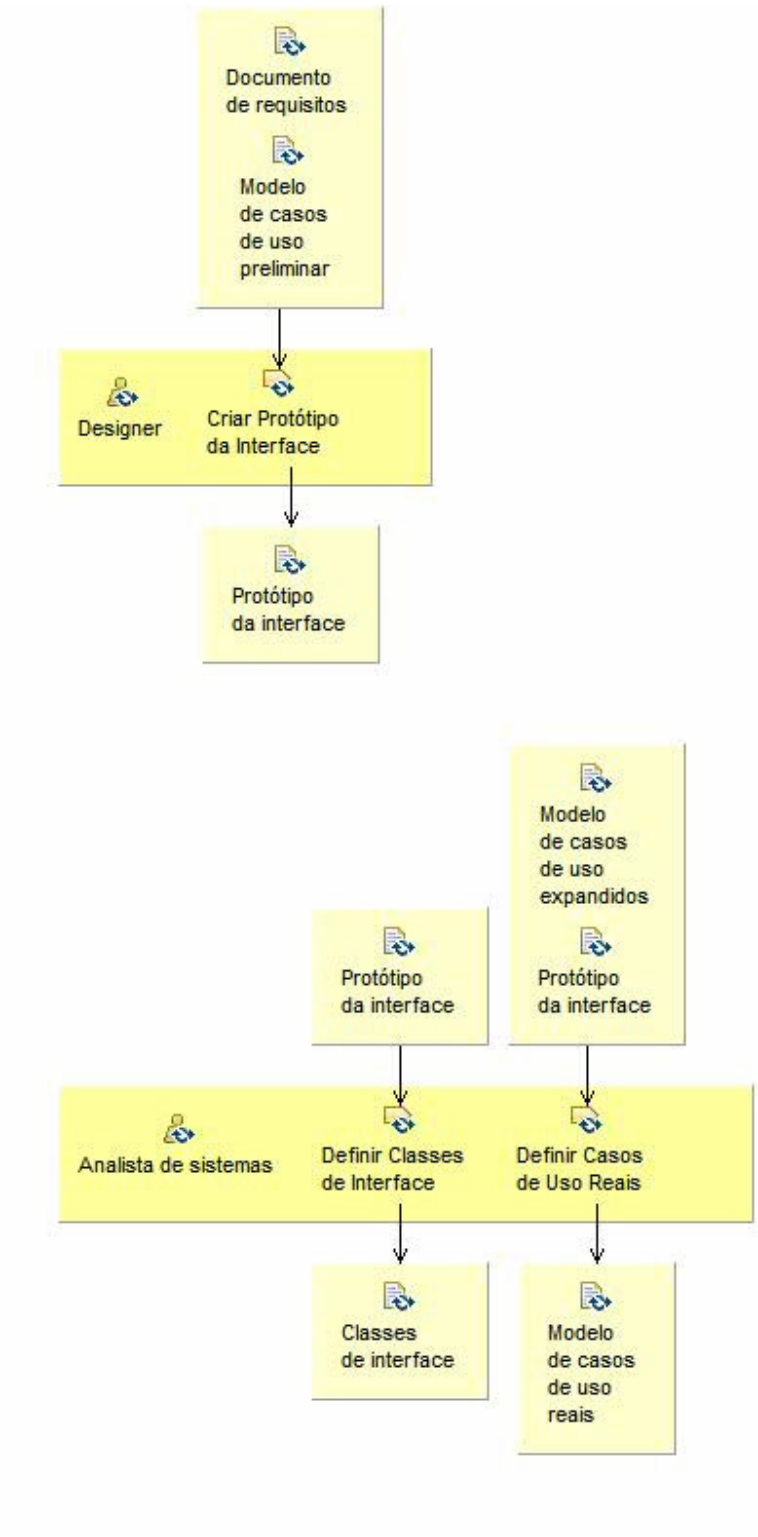


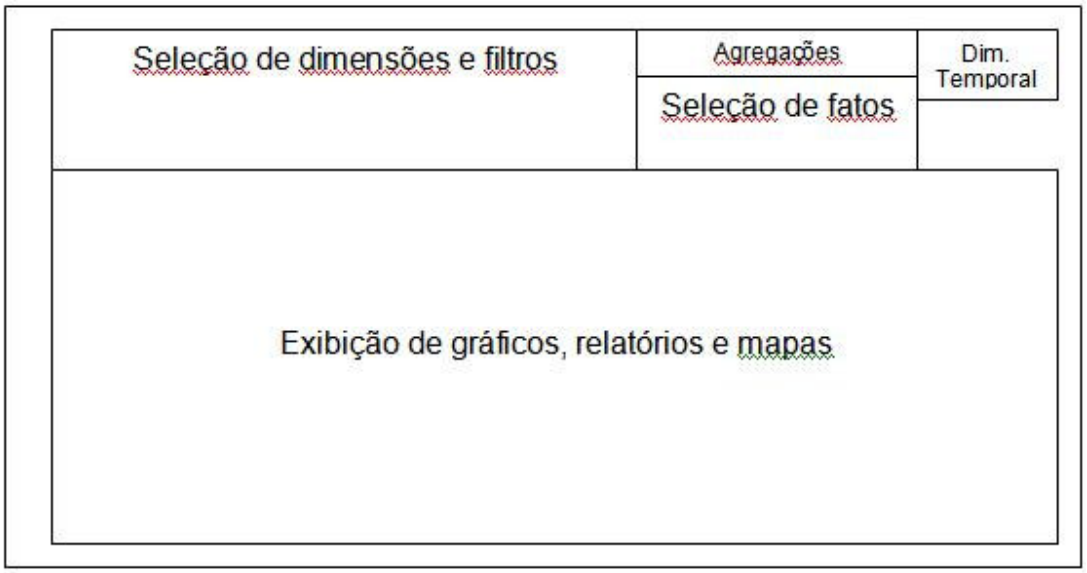
Figura 118- Atividade definir interface

Criar protótipo da interface

⊟ Definir medidas a serem incluídas
Definir todas as medidas, convencionais ou espaciais, que farão parte da interface
aplicação a ser desenvolvida. Essas medidas estarão localizadas em combo-boxes
de forma a serem selecionadas para análise.

⊟ Definir dimensões a serem incluídas
Definir dimensões a serem incluídas em list-boxes nas interfaces.

⊟ Fazer design da interface
O design da interface é feito segundo o seguinte padrão:



O layout desta interface contém as seguintes áreas:

A área definição de filtros e dimensões contém list-boxes para obtenção de filt
aplicáveis às dimensões.

A área seleção de dimensões contém um combo-box para seleção da dimens
a ser visualizada.

A área seleção de fatos contém um combo-box para seleção do fato a
visualizado.

A área exibição de gráficos, relatórios e mapas permite a visualização d
informações na forma de gráficos torta ou barras , de relatórios sintéticos ou analíticos e
mapas (quando aplicável).

Definir classes de interface

⊟ Aplicar princípios de projeto de classes de interface

Detalhes de implementação são completamente omitidos do desenvolvedor, que tem ciên apenas de métodos e atributos para exibição e coleta de dados da interface.

As classes de interface devem ser hierarquicamente estruturadas por meio de relações de herança. Em aplicações web especificamente, diversas páginas têm partes em comum (cabeçalho e rodapé, por exemplo), variando em outras (as partes centrais e laterais). Estas partes em comum são alocadas a classes mães e os detalhes específicos a classes derivadas.

⊟ Criar classes de interface

Definir as classes de interface com os objetos de interface a serem vinculados às mesmas: combo boxes, list-boxes, gráficos, relatórios e mapas.

Definir casos de uso reais

⊟ Definir associação com a interface

Definir interfaces associadas ao caso de u

⊟ Descrever casos de uso reais

Cada caso de uso terá sua descrição feita por meio da interação com as interface a ele associadas, tal qual um manual do usuário. Descrever eventos de interação com a interface e as respostas do sistema aos mesmos.

A.2.1.2-Atividade detalhar casos de uso

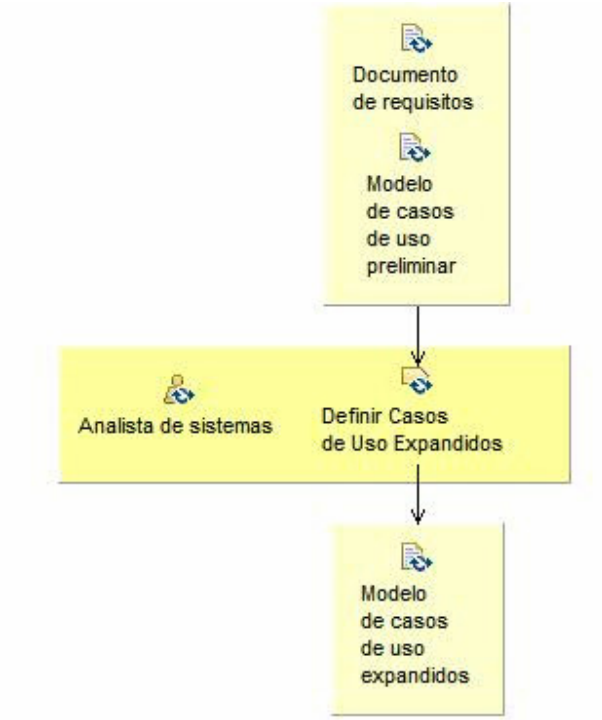


Figura 119- Atividade detalhar casos de uso

Definir casos de uso expandidos

⊟ Detalhar descrição
Casos de uso expandidos contêm uma descrição minuciosa das regras de negócio. Uma forma usual, estabelecer ações dos atores e as respostas do sistema às

mesmas.

A.2.1.3-Atividade definir diagramas de classes de análise

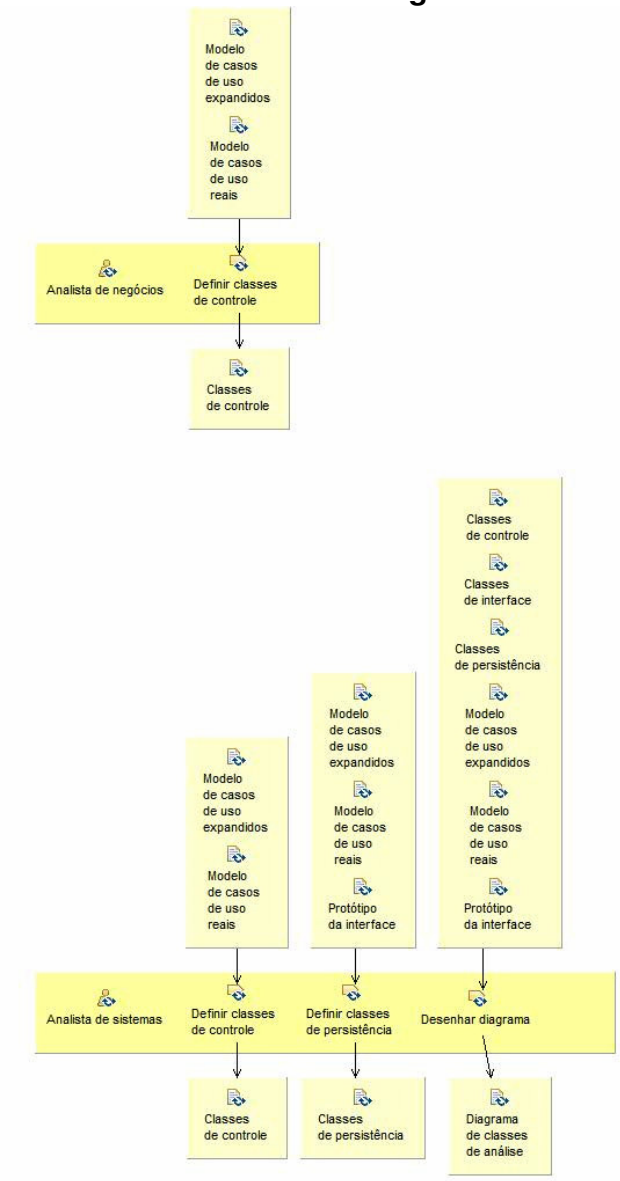


Figura 120- Atividade definir diagramas de classes de análise

Definir classes de controle

⊟ Definir classes de controle
Cada classe de controle é associada um caso de uso, que neste caso será relacionado a funcionalidades do *data warehouse.*

Definir classes de persistência

⊟ Definir classes de persistência
Definir todas as classes de presistência associadas ao domínio do problema.
⊟ Definir atributos das classes de persistência
Definir todos os atributos necessários à descrição da classe de persistência.

Desenhar diagrama

☐ Desenhar classes de persistência
Desenhar no diagrama todas as classes de persistên
⊟ Desenhar classes de controle
Desenhar no diagrama todas as classes de contr
⊟ Desenhar classes de interface
Desenhar no diagrama as classes de interface mais importantes (desenhar todas pode sobrecarregar visualmente o dagrama)
⊟ Desenhar associações
Classes de controle usalmente se associam a uma ou mais classes de persistência e podem ou não se associar a classes de interface.

A.2.2-A disciplina Análise Multidimensional

A disciplina análise e especificação envolve atividades e tarefas necessários à criação do modelo multidimensional, como definição de dimensões, medidas, tabelas e das fontes de dados, em um *data warehouse* convencional ou espacial.

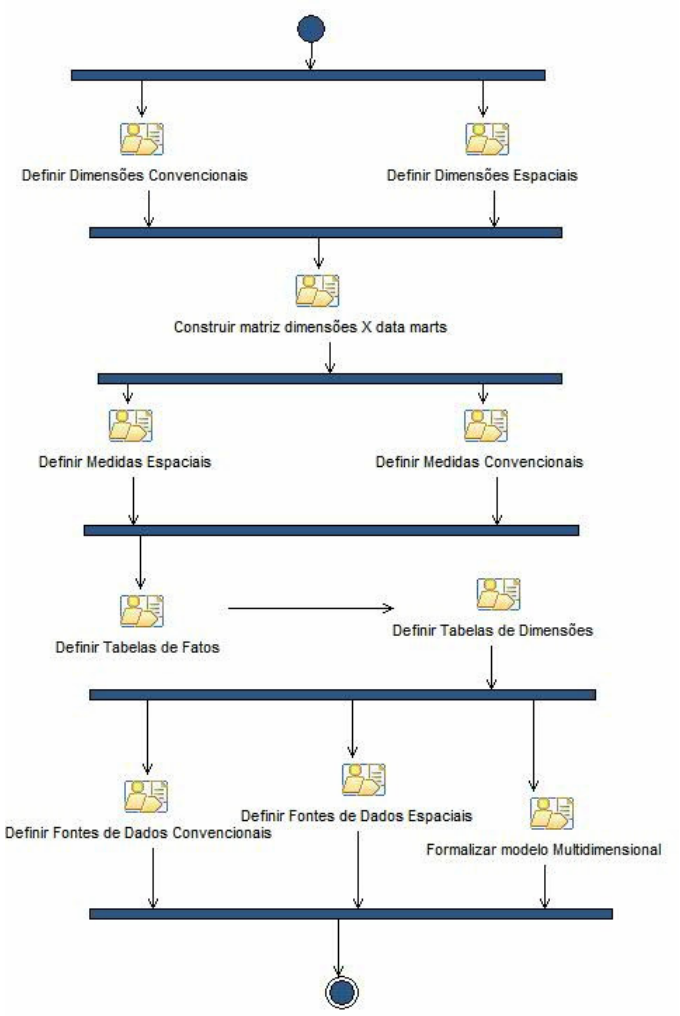


Figura 121-Diagrama de atividades da disciplina Análise Multidimensional

A.2.2.1-Atividade definir dimensões convencionais

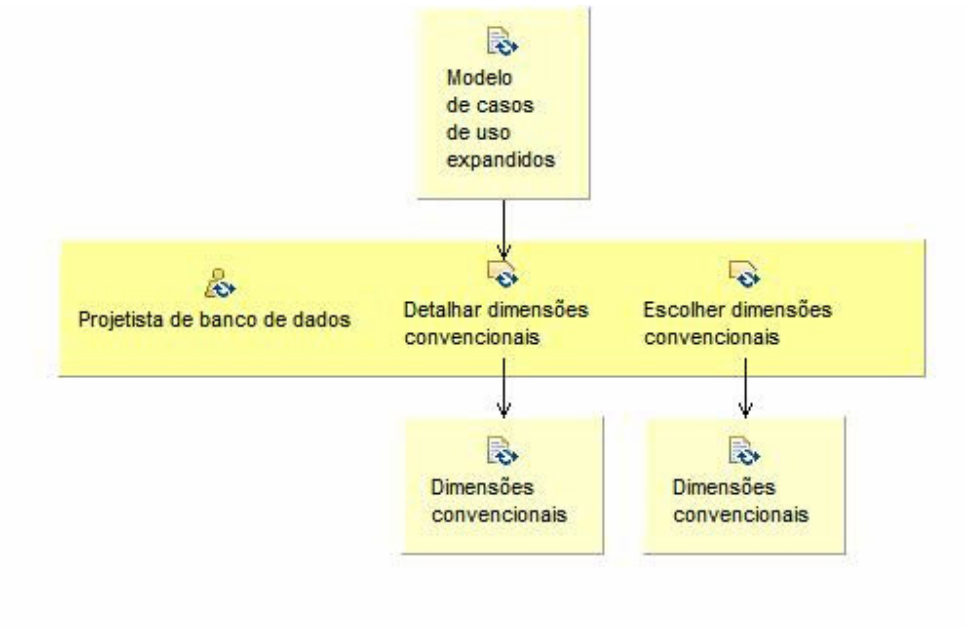


Figura 122- Atividade definir dimensões convencionais

Detalhar dimensões convencionais

⊟ Fazer descrição
Fazer uma descrição detalhada da dimensão, enfatizando sua aplicação no negócio.
⊟ Definir estrutura
Definir a estrutura hierárquica das dimensões, se houver: *snow flake* ou *estrela.*

Escolher dimensões convencionais

⊟ Analisar requisitos para capturar dimensões
As necessidades de relatórios e informações capturadas nos requisitos e na análise de requisitos são a fonte primária para obtenção das dimensões do modelo.
⊟ Listar dimensões
Nessa tarefa é feita uma lista de dimensões do modelo, sem muita preocupação com com nomes ou restrições.

A.2.2.2-Atividade definir medidas espaciais

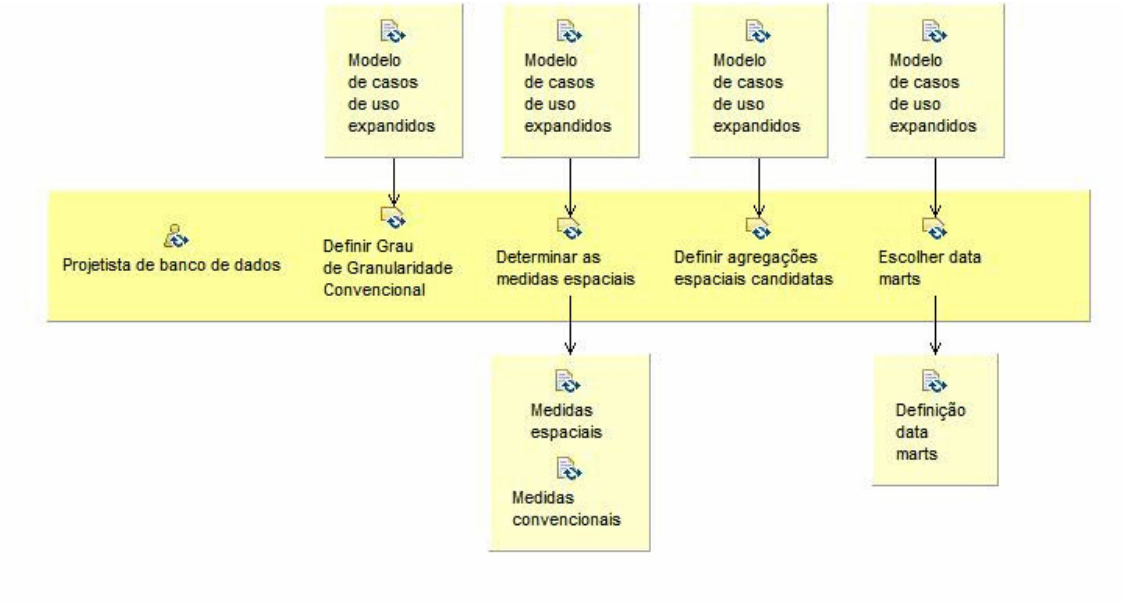


Figura 123- Atividade definir medidas espaciais

Definir grau de granularidade convencional

⊟ Avaliar relatórios definidos nos requisitos.
Este passo verifica os relatórios descritos na etapa de requisitos para definir o menor grau de granularidade requerido para as tabelas de fatos.
⊟ Avaliar particularidades da granularidade espacial
De forma a permitir consultas sobre áreas ad hoc, recomenda-se que a menor granularidade espacial seja a de objetos do tipo ponto. Se não se considerar este nível de granularidade, distâncias, por exemplo, entre polígonos, devem ser estimadas (Fidalgo, 2005).
⊟ Determinar o grau de granularidade
Uma vez avaliados os requisitos , a granularidade é determinada em função da necessidade mínima de análise, em nível de tomada de decisões. Por exemplo, e uma gerência de produção, pode-se desejar a informação diária acerca da produção. Já para uma gerência geral, basta a informação de produção semanal.

Determinar as medidas espaciais

⊟ Escolher medidas
Para cada *data mart, d*efinir tantas medidas quantas sejam necessárias ao grau d granularidade definido na tarefa anterior e considerando-se as necessidades definidas nos requisitos.
⊟ Determinar variáveis
Determinar variáveis que terão valores distintos alocados a objetos geográficos

distintos. Por exemplo, variáveis *temperatura e umidade* podem possuir faixas de valores alocadas a conjuntos de regiões que as possuam.

⊟ Determinar feição geográfica associada às variáveis
Determinar que tipo de feições geográficas vão estar vinculadas à variáveis definidas no passo anterior. Por exemplo, temperatura e umidade com mesmas faixas estarão associadas a listas de polígonos que possuem estas faixas.
⊟ Criar a medida espacial
Criar a dimensão espacial, associando-se variáveis a vetores de feições geográficas.
⊟ Determinar chaves
Neste nível, determinar apenas chaves temporárias, que serão refinadas posteriormente.
⊟ Definir medidas derivadas
Definir medidas derivadas de outras medidas, ou seja, que sejam obtidas de combinação de outras medidas.

Determinar agregações espaciais candidatas

⊟ Definir estratégias de agregação espacial
Definir estratégias, por meio de operações espaciais, para agregação espac
⊟ Definir medidas a serem agregadas
Considerar, segundo (Kimball, 1996) requisitos do negócio e a distribuição estatística dos dados.
□ Definir tabelas de fatos agregadas
Agregações devem ser alocadas em suas próprias tabelas de fatos, separadas d dados bases.
⊟ Definir tabelas de dimensões agregadas
Tais tabelas são versões reduzidas das dimensões origin

Escolher data marts
⊟ Analisar matrix dimensões X data marts
A primeira tabela de fatos deve vir de um *data mart* de origem úni
⊟ Associar data marts a fontes de dados
Data marts devem estar associados a fontes de dad

A.2.2.3-Atividade definir dimensões espaciais

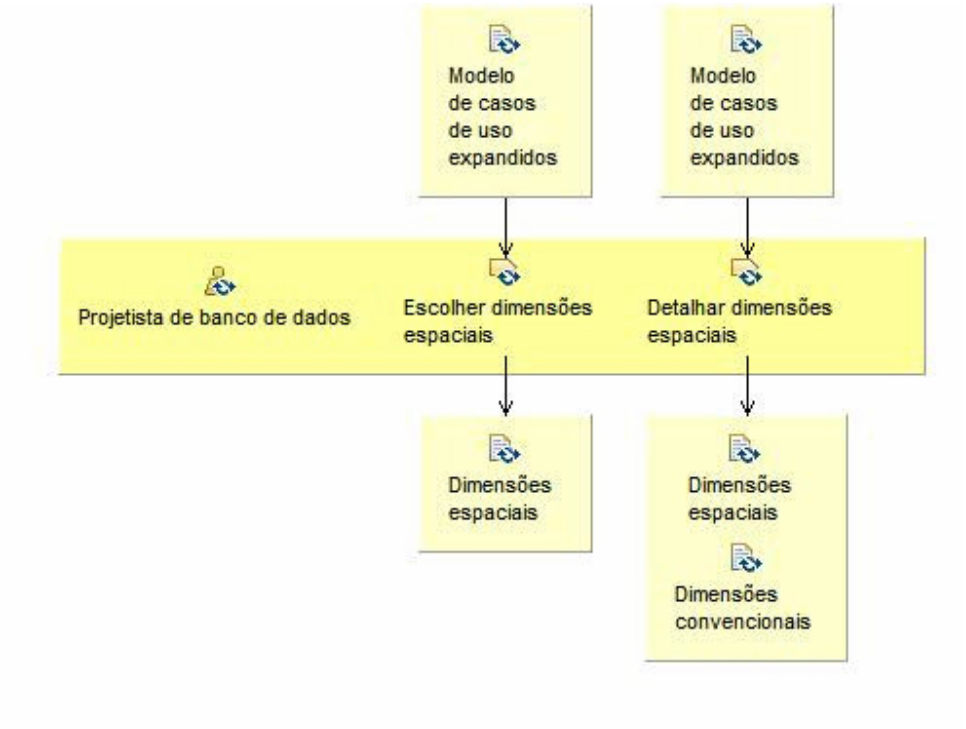


Figura 124- Atividade definir dimensões espaciais

Escolher dimensões espaciais

⊟ Capturar dimensões espaciais

A partir dos requisitos, capturar dimensões espaciais. Tais dimensões estão associadas a feições espaciais.

⊟ Listar dimensões espaciais.

Fazer uma lista de dimensões espaciais de forma a proporcionar um tratamento diferenciado às mesmas em outras tarefas.

Detalhar dimensões espaciais

⊟ Definir categorias das dimensões

Quanto à natureza da granularidade , (Stefanovic, 1997) identifica dois tipos dimensões espaciais:

Dimensão espacial-para-não-espacial: em um nível primitivo de granularida associa-se a um objeto espacial, mas em níveis mais altos torna-se não espacial.

Dimensão espacial-para-espacial: neste caso, a dimensão é associada a um obj

espacial em todos os níveis.

Quanto ao relacionamento entre dimensões espaciais propõe-se a segui classificação(Fidalgo, 2005):

Dimensão espacial primitiva: contém apenas um campo espacial e não possui cha estrangeiras provenientes de outras dimensões primitivas. Não contém cha estrangeiras para tabelas de dimensões convencionais.

Dimensão espacial composta: possui chaves estrangeiras de outras dimens espaciais primitivas ou de outras dimensões espaciais compostas.

Dimensão híbrida: possui chaves estrangeiras de outras dimensões espac primitivas, de outras dimensões espaciais compostas e de dimensões convencion Pode ser classificada em *micro ,* quando faz referência a apenas uma dimensão primiti *macro,* quando referenciar dimensões primitivas ou compostas, e *conjunta,* quando uma união dos dois tipos anteriores . As dimensões hibridas macro podem ser de *úr junção,* quando se relacionarem por meio de uma única chave estrangeira a u dimensão espacial composta, ou de *múltiplas junções,* quando se relacionarem com vá dimensões geográficas primitivas .

⊟ Fazer descrição

Fazer uma descrição detalhada da dimensão, enfatizando sua aplicação no negócio.

⊟ Definir estrutura

Definir a estrutura hierárquica das dimensões, se houver: *snow flake* ou *estr*

A.2.2.4-Atividade definir fontes de dados convencionais

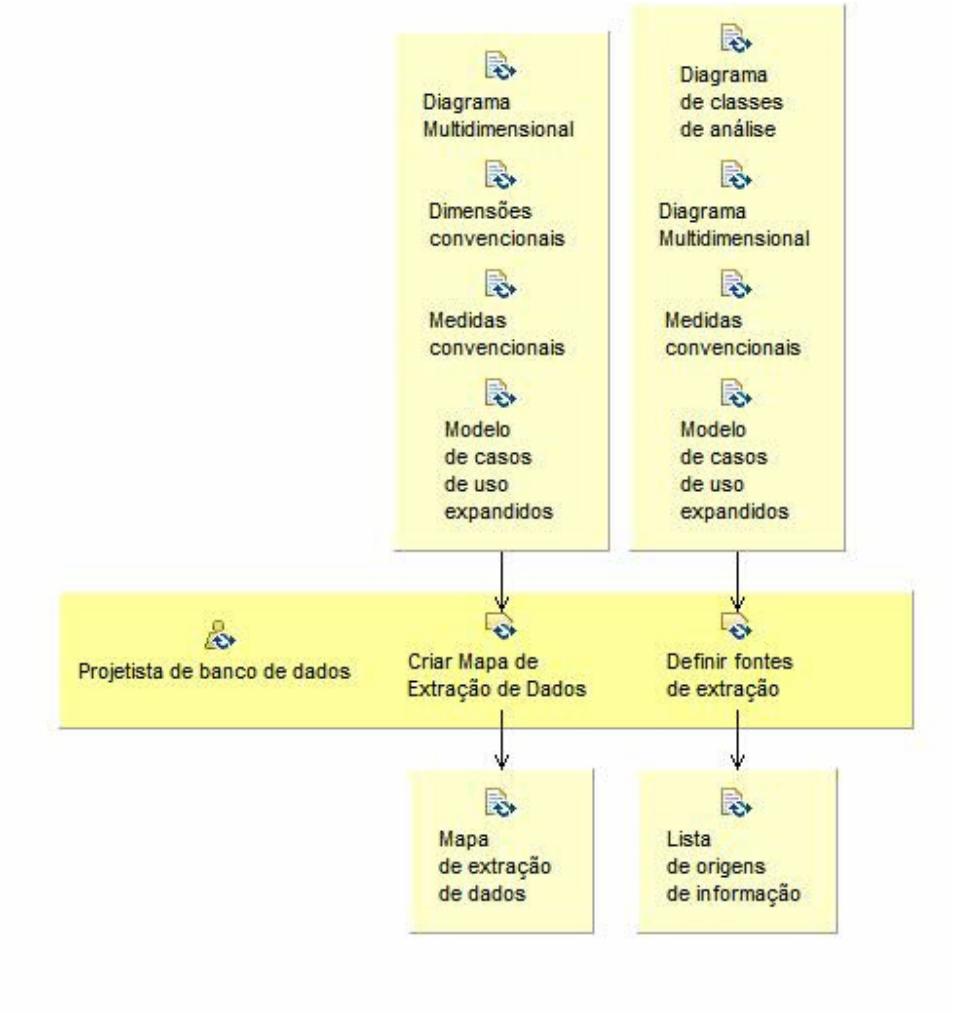


Figura 125- Atividade definir fontes de dados convencionais

Criar mapa de extração de dados

⊟ Fazer mapa de extração de dados
O objetivo principal é capturar onde os dados estão localizados. O mapa de extração é uma tabela com as seguintes colunas:

Nome da tabela lógica no *data warehouse*
Nome da coluna lógica no *data warehouse*
Tipo de dado a coluna lógica no *data warehouse*
Tamanho do campo da coluna lógica
Descrição da coluna lógica
Nome do sistema de origem de onde as colunas lógicas são alimentadas. Pode haver mais de uma fonte.
O nome da tabela ou arquivo que alimenta as colunas lógicas. Pode haver mais de uma fonte.

O nome do campo ou coluna que alimenta as colunas lógicas. Pode haver mais de uma fonte.

Observações sobre quaisquer transformações necessárias à tradução da for de informação ao formato requerido pela coluna de destino

O nome da dimensão ou *data mart* que a coluna representa

Nome de um atributo específico em uma tabela de dimensões ou de fatos.

⊟ Identificar elementos de dados não existentes na fonte de dados

Elementos de dados não existentes na fonte de dados são também identificados. Em alguns casos, pode ser que o dado não seja capturado de lugar algum. Em outros casos, o dado existe, mas o escopo atual do projeto não inclui a fonte

⊟ Revisar o projeto das tabelas de fatos

Revisar o projeto das tabelas de fatos de forma a refletir o que será realmente incluído. Alguns atributos serão possivelmente utilizados no futuro, ou ainda, pode se identificar a necessidade de outros atributos,

Definir fontes de extração

⊟ Identificar fontes de dados candidatas

Identificar em alto nível, o que são cada uma das fontes, sua disponibilidade e qu é responsável por elas. As fontes candidatas, normalmente, estão descritas no documento de requisitos e outras que possam suportá-las.

⊟ Determinar as fontes-bases do projeto

Entender de maneira mais profunda, as fontes que são mais práximas às fontes primárias.

⊟ Definir o ponto de extração

Definir o ponto em que os dados de processamento devem ser extraídos. Se o projeto não for limitado a uma fonte de dados específica, então há mais trabalho a ser feito para definir onde começar. Os critérios para tal incluem:

Escolha de fontes com acesso mais simples
Escolha de tempos de alimentação de dados menores
Acuidade dos dados
Cronograma do projeto

⊟ Incluir informações sobre as fontes de dados

Incluir as seguintes informações:

Nome da fonte
Responsável pela fonte
Plataforma (xls, odbc, etc)
Localização da fonte
Descrição do que a fonte faz

A.2.2.5-Atividade definir medidas convencionais

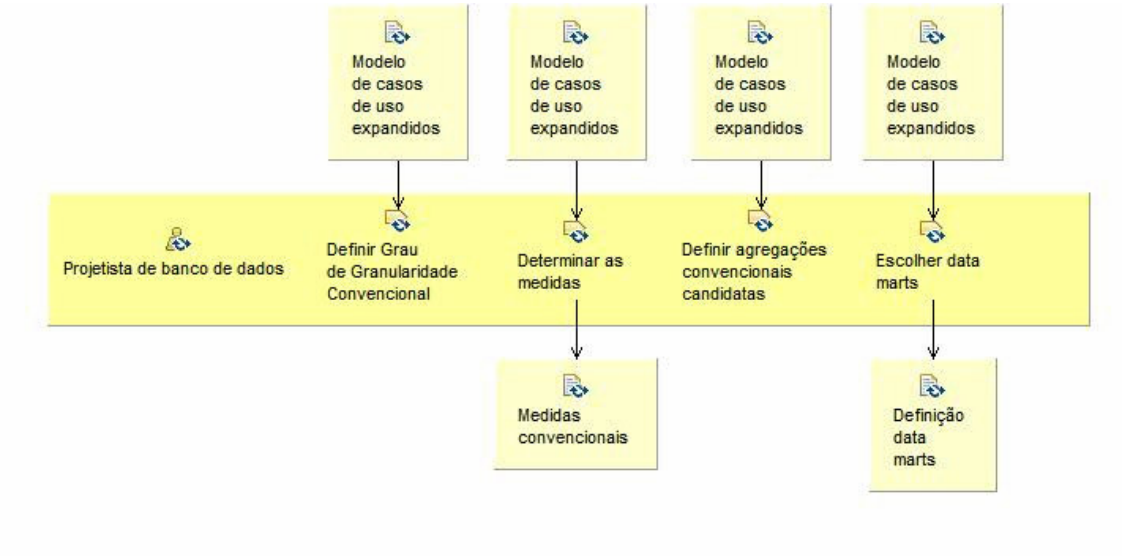


Figura 126- Atividade definir medidas convencionais

Definir grau de granularidade convencional

⊟ Avaliar relatórios definidos nos requisitos.
Este passo verifica os relatórios descritos na etapa de requisitos para definir o menor grau de granularidade requerido para as tabelas de fatos.
⊟ Determinar o grau de granularidade
Uma vez avaliados os requisitos , a granularidade é determinada em função da necessidade mínima de análise, em nível de tomada de decisões. Por exemplo, e uma gerência de produção, pode-se desejar a informação diária acerca da produção. Já para uma gerência geral, basta a informação de produção semanal.

Determinar as medidas
⊟ Escolher medidas
Para cada *data mart, d*efinir tantas medidas quantas sejam necessárias ao grau de granularidade definido na tarefa anterior e considerando-se as necessidades definidas nos requisitos.
⊟ Determinar chaves
Neste nível, determinar apenas chaves temporárias, que serão refinadas posteriormente.
⊟ Definir medidas derivadas
Definir medidas derivadas de outras medidas, ou seja, que sejam obtidas de combinação de outras medidas.

Definir agregações convencionais candidatas

⊟ Definir medidas a serem agregadas
Considerar, segundo (Kimball, 1996) requisitos do negócio e a distribuição estatística dos dados.
⊟ Definir tabelas de fatos agregadas
Agregações devem ser alocadas em suas próprias tabelas de fatos, separadas d dados bases.
⊟ Definir tabelas de dimensões agregadas
Tais tabelas são versões reduzidas das dimensões origin

Escolher data marts

⊟ Analisar matriz dimensões X data marts
A primeira tabela de fatos deve vir de um *data mart* de origem úni
⊟ Associar data marts a fontes de dados
Data marts devem estar associados a fontes de dad

A.2.2.6-Atividade definir fontes de dados espaciais

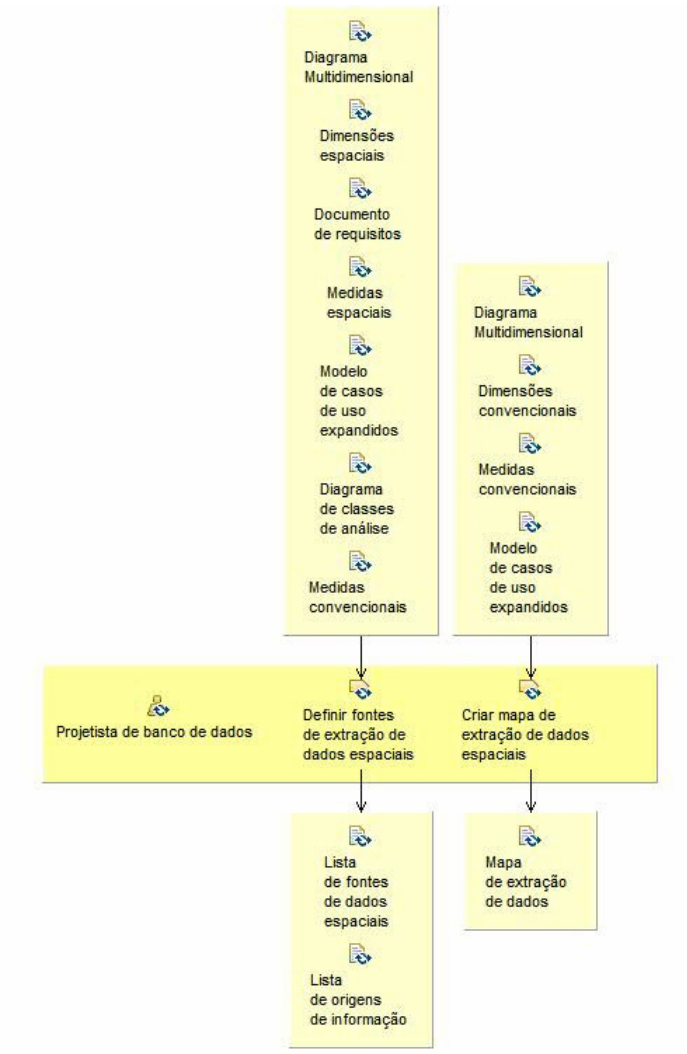


Figura 127- Atividade definir fontes de dados espaciais

Definir fontes de extração de dados espaciais

⊟ Identificar fontes de dados candidatas
Identificar em alto nível, o que são cada uma das fontes, sua disponibilidade e qu
é responsável por elas. As fontes candidatas, normalmente, estão descritas no
documento de requisitos e outras que possam suportá-las.
⊟ Determinar as fontes-bases do projeto
Entender de maneira mais profunda, as fontes que são mais práximas às fontes
primárias.
⊟ Definir o ponto de extração
Definir o ponto em que os dados de processamento devem ser extraídos. Se o
projeto não for limitado a uma fonte de dados específica, então há mais trabalho a

ser feito para definir onde começar. Os critérios para tal incluem:

Escolha de fontes com acesso mais simples
Escolha de tempos de alimentação de dados menores
Acuidade dos dados
Cronograma do projeto

⊟ Incluir informações sobre as fontes de dados
Incluir as seguintes informações:

Nome da fonte
Responsável pela fonte
Plataforma (xls, odbc, etc)
Localização da fonte
Descrição do que a fonte faz

⊟ Definir fontes de dados espaciais
As origens de dados espaciais podem ser SIGs (formatos SHP ou MIF), bancos d
dados com extensões espaciais, planilhas excel, arquivos texto, XML, dentre outr

Criar mapa de extração de dados espaciais
⊟ Fazer mapa de extração de dados
O objetivo principal é capturar onde os dados estão localizados. O mapa de extração é ur
tabela com as seguintes colunas:

Nome da tabela lógica no *data warehouse*
Nome da coluna lógica no *data warehouse*
Tipo de dado a coluna lógica no *data warehouse*
Tamanho do campo da coluna lógica
Descrição da coluna lógica
Nome do sistema de origem de onde as colunas lógicas são alimentadas. Pode have
mais de uma fonte.
O nome da tabela ou arquivo que alimenta as colunas lógicas. Pode haver mais de
uma fonte.
O nome do campo ou coluna que alimenta as colunas lógicas. Pode haver mais de
uma fonte.
Observações sobre quaisquer transformações necessárias à tradução da fonte de
informação ao formato requerido pela coluna de destino
O nome da dimensão ou *data mart* que a coluna representa
Nome de um atributo específico em uma tabela de dimensões ou de fatos.

⊟ Acrescentar particularidades espaciais no mapa de extração
Além das informações convencionais, acrescentar:

Tipo de dado espacial

Operações espaciais a serem aplicadas no processo de extraç

Identificar elementos de dados não existentes na fonte de dados
Elementos de dados não existentes na fonte de dados são também identificados. Em alguns casos, pode ser que o dado não seja capturado de lugar algum. Em outros casos, o dado existe, mas o escopo atual do projeto não inclui a fonte
Revisar o projeto das tabelas de fatos
Revisar o projeto das tabelas de fatos de forma a refletir o que será realmente incluído. Alguns atributos serão possivelmente utilizados no futuro, ou ainda, pode se identificar a necessidade de outros atributos,

A.2.2.7-Atividade definir tabela de fatos

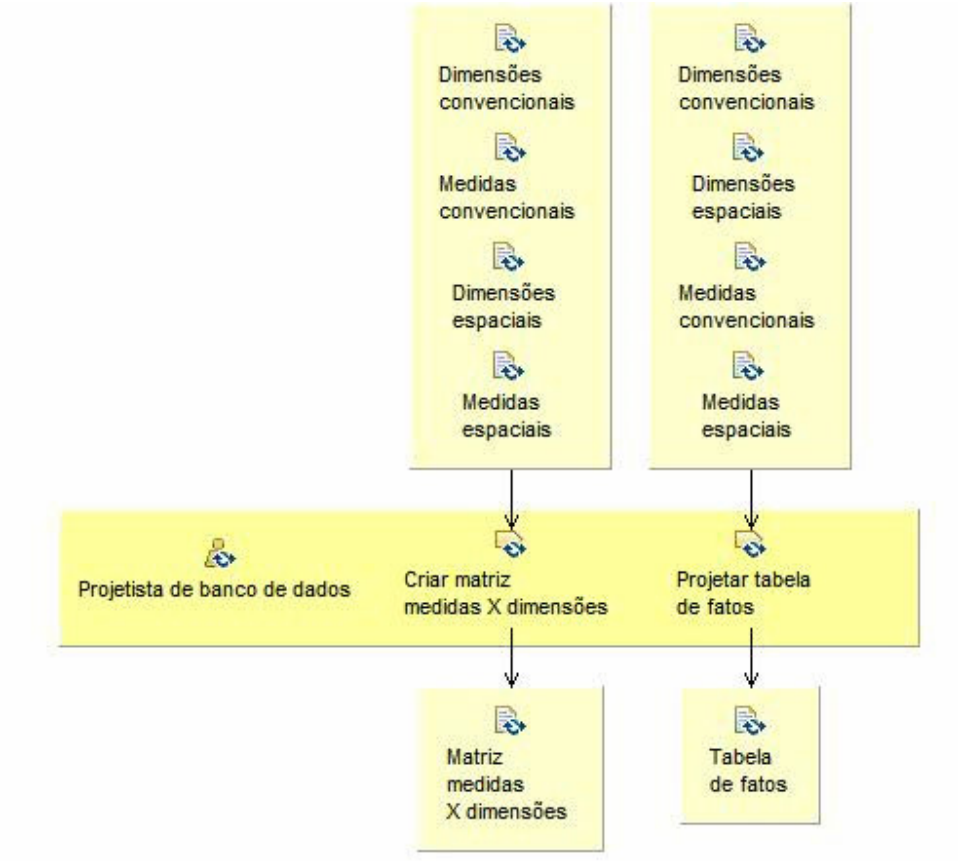


Figura 128 Atividade definir tabela de fatos

Criar matriz medidas X dimensões

Selecionar medidas e dimensões
Selecionar dentre as medidas e dimensões anteriormente definidas, aquelas que

efeitvamente vão fazer parte da matriz.
⊟ Criar matriz
Criar a matriz, alocando as dimensões nas linhas e as medidas nas colunas. As células determinam quais dimensões afetam determinada medida.

Projetar tabela de fatos

⊟ Adicionar medidas básicas
Adicionar as medidas básicas definidas na atividades definir medid
⊟ Adicionar chaves estrangeiras
Adicionar chaves estrangeiras provenientes das dimensões associadas às medidas pertencentes a esta tabela (definidas na matriz dimensões X medidas)
⊟ Adicionar medidas derivadas
Adicionar as medidas derivadas definidas na atividades definir medidas.

A.2.2.8-Atividade definir tabela de dimensões

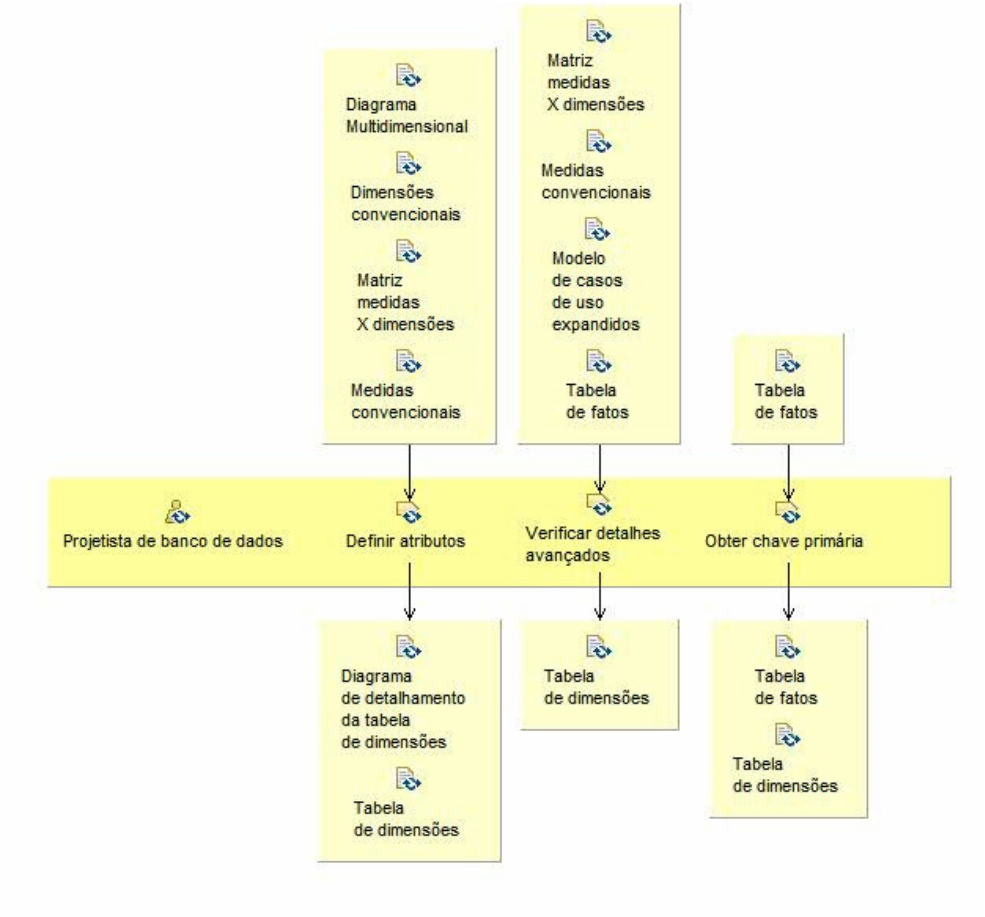
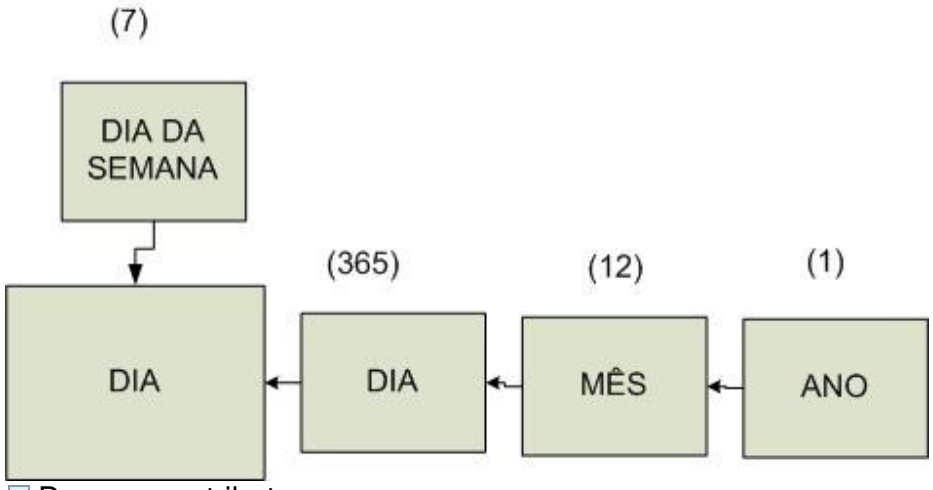


Figura 129- Atividade definir tabela de dimensões

Detalhar atributos

⊟ Fazer lista de atributos
Fazer uma lista de possíveis atributos candidatos. Atributos descritivos são os mais relevantes.
⊟ Fazer diagrama de detalhamento da tabela de dimensões
O diagrama de detalhamento da tabela de dimensões mostra os atributos relacionados a uma única dimensão, a cardinalidade dos mesmos e as hierarquias e relações entre eles.



⊟ Descrever atributos
Itens descritivos: nome, definição, cardinalida

Verificar detalhes avançados
Analisar relações muitos para muitos
Avaliar a real necessidade de tais relaçõ
⊟ Identificar atributos que são pouco modificados
Identificar atributos que são pouco modificados ao longo do tem
⊟ Identificar atributos artificiais
Identificar atributos não existentes no modelo de negócios atual, mas necessários em níveis mais baixos de detalhamento, de forma a suportar *rollups* na dimensão.

Obter chave primária
Listar atributos candidatos
Fazer uma lista de atributos candidatos a identificadores da dimensão
⊟ Determinar chave primária
Obter chaves primárias simples ou compostas.

A.2.2.9-Atividade construir matriz dimensões X data marts

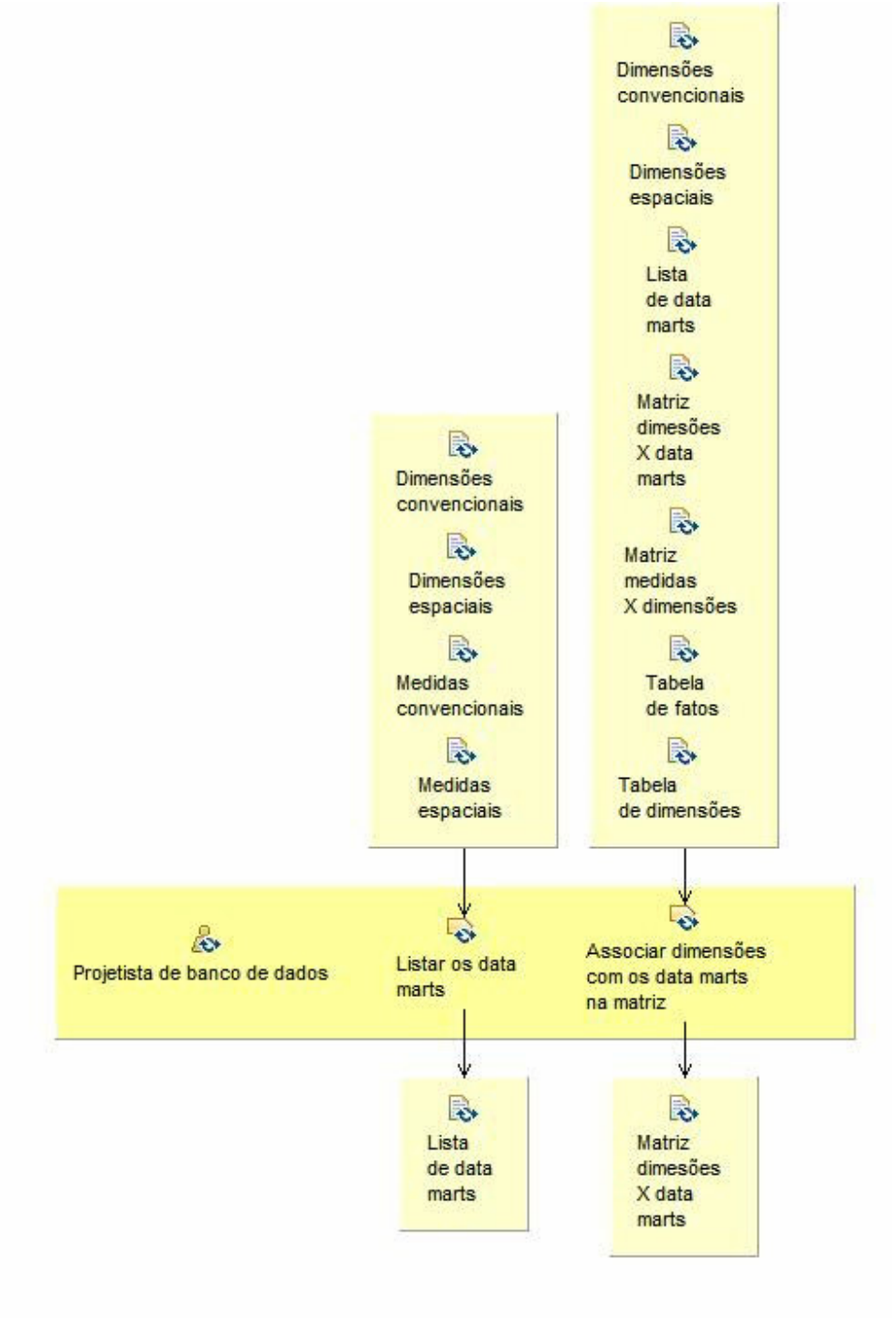


Figura 130- Atividade construir matriz dimensões X data marts

Listar os data marts

⊟ Listar os data marts de origem única

Os *data marts* de origem única são aqueles definidos em torno de um único tipo d fonte de dados.

- Listar os data marts de origem múltipla

Os *data marts* de origem múltipla combinam os de origem única em visões mais abrangentes do negócio.

Associar dimensões com os data marts na matriz

- Marcar as intersecções

Com as linhas (*data marts*) e colunas (dimensões) definidas, marcam-se na matriz, para cada *data mart,* quais dimensões estão a eles vinvuladas.

- Ressaltar as dimensões importantes

Se determinada coluna foi muito marcada, a dimensão a ela correspondente é importante e deve ser cuidadosamente definida em todos os *data marts* aos quais pertence.

A.2.2.10-Atividade Formalizar modelo multidimensional

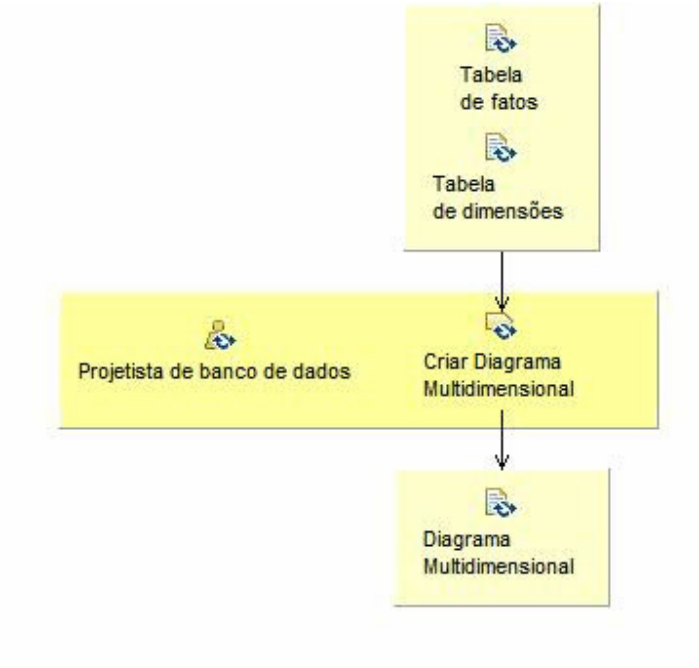


Figura 131- Atividade Formalizar modelo multidimensional

Criar diagrama multidimensional

- Adicionar Entidades

Desenhar diagrama contendo entidades (tabelas de fatos e dimensões)

- Adicionar Atributos

Definir atributos das tabelas de dimensões (códigos e informações) e da tabela de fatos (medidas selecionadas e chaves estrangeiras)

- Estabelecer relações

As relações serão estabelecidas de acordo com o modelo desejado: estrela (mais eficiente em termos de desempenho) ou *snow flake* (mais eficiente em termos de armazenamento)

A.2.3-A disciplina Projeto Análise de Dados

Esta disciplina envolve atividades e tarefas necessários ao projeto de cubos convencionais e espaciais e de definição de técnicas de *data mining* convencional e espacial.

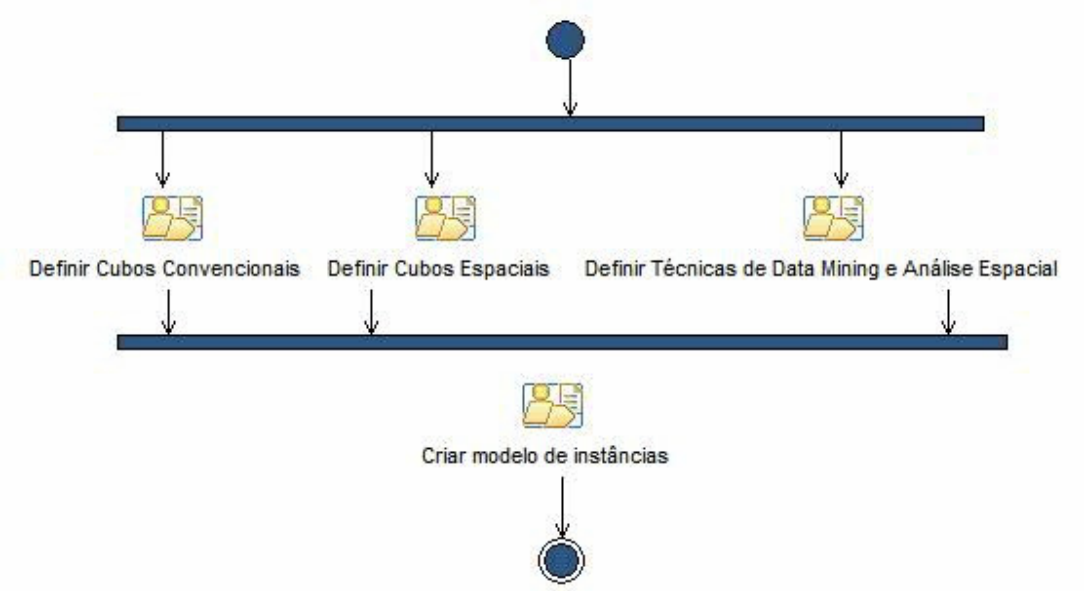


Figura 132-Diagrama de atividades da disciplina Projeto Análise de Dados

A.2.3.1-Atividade definir cubos convencionais

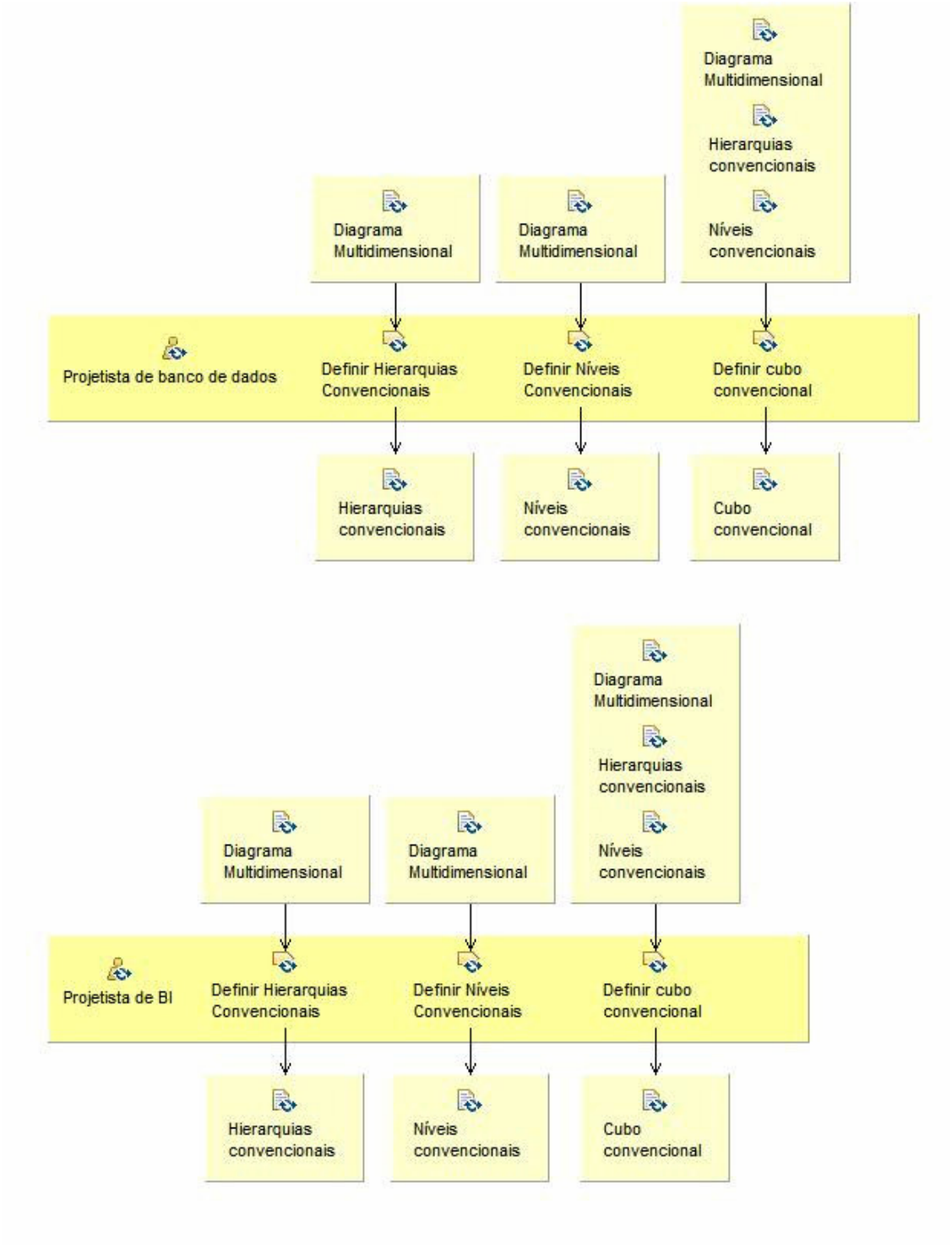


Figura 133- Atividade definir cubos convencionais

Definir hierarquias convencionais

⊟ Definir hierarquias

Hierarquias são determinadas por combinações de nív

⊟ Definir níveis da hierarquia
Definem-se e associam-se níveis de uma determianda dimensão a hierarquias específicas.
⊟ Definir atributos da hierarquia
Definir atributos da hierarquia. Os atributos essenciais são identificador e descrição.

Definir níveis convencionais
Definir nivel
O nível está associado a uma dimensão. Estabelecer os diversos níveis em que uma dimensão pode ser detalhada.
⊟ Definir atributos do nível
Os diversos níveis de detalhamento de uma dimensão possuem atributos específicos que os descreverão. Os níveis vão expor atributos específicos da dimensão associada.

Definir cubo convencional
⊟ Associar a medidas
Associar os cubos às medidas que ele conté
⊟ Associar a dimensões
Associar o cubo a dimensões que o determina

A.2.3.2-Atividade definir cubos espaciais

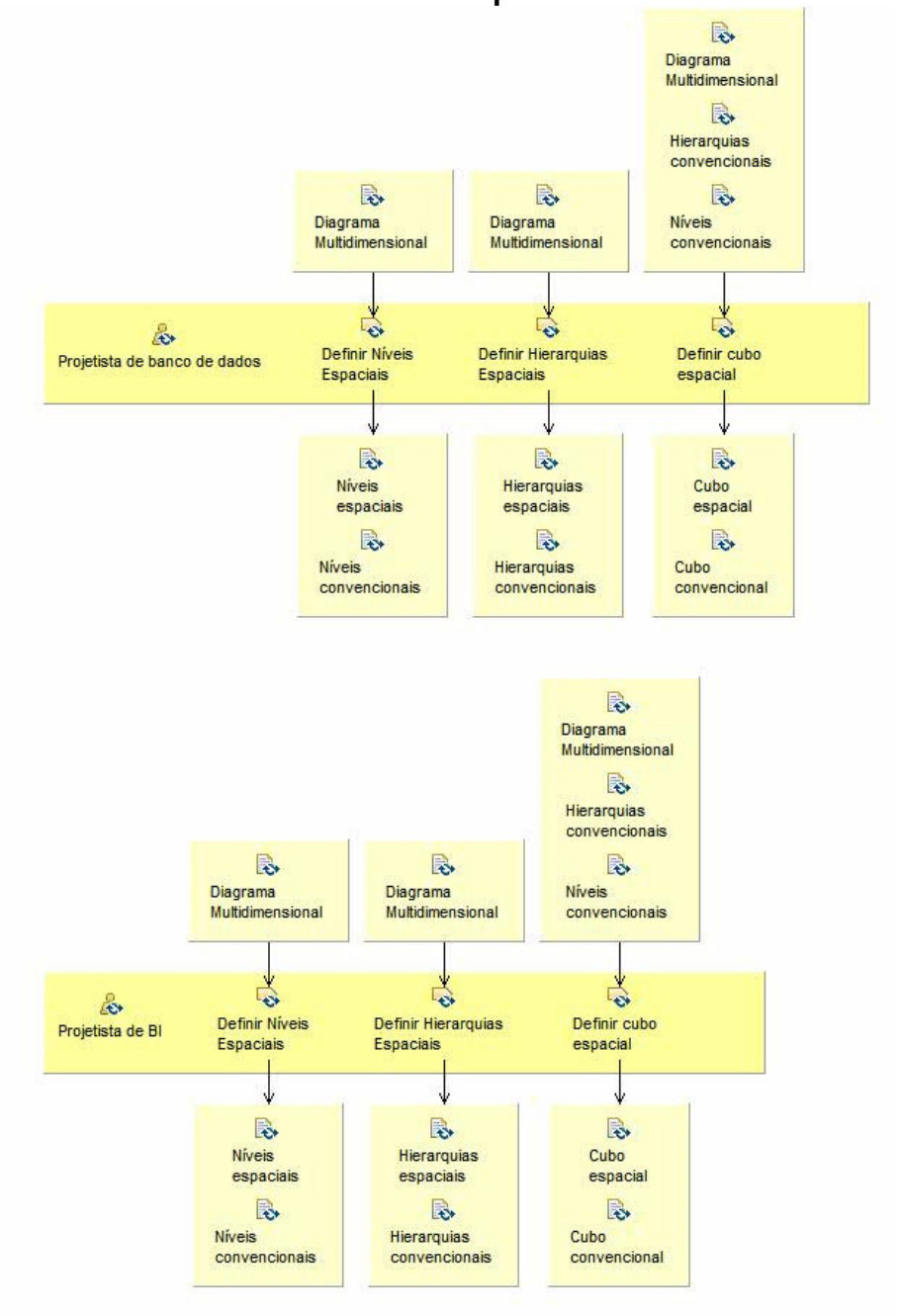


Figura 134- Atividade definir cubos espaciais

Definir níveis espaciais

⊟ Definir nivel
O nível está associado a uma dimensão. Estabelecer os diversos níveis em que uma dimensão pode ser detalhada.
⊟ Definir atributos do nível
Os diversos níveis de detalhamento de uma dimensão possuem atributos específicos que os descreverão. Os níveis vão expor atributos específicos da dimensão associada.

Definir hierarquias espaciais
⊟ Definir nivel
O nível está associado a uma dimensão. Estabelecer os diversos níveis em que uma dimensão pode ser detalhada.
⊟ Definir atributos do nível
Os diversos níveis de detalhamento de uma dimensão possuem atributos específicos que os descreverão. Os níveis vão expor atributos específicos da dimensão associada.

Definir cubo espacial
⊟ Associar a medidas
Associar os cubos às medidas que ele conté
⊟ Associar a dimensões
Associar o cubo a dimensões que o determina

A.2.3.3-Atividade criar modelo de instâncias

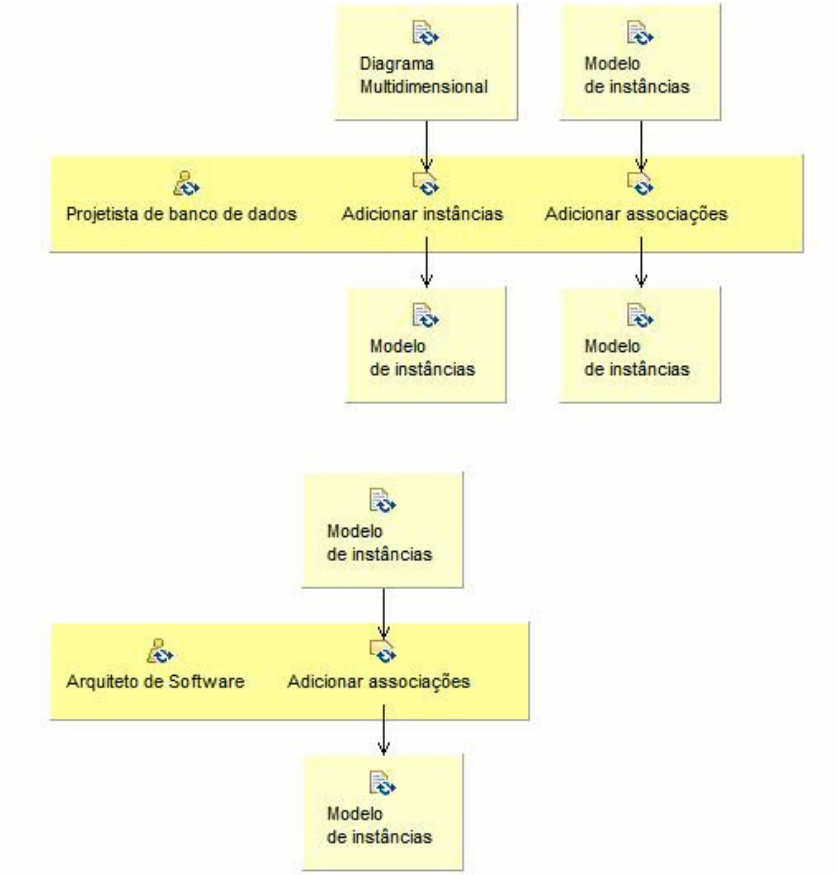


Figura 135- Atividade criar modelo de instâncias

Adicionar instâncias

⊟ Desenhar instâncias de dimensões
Desenhar instâncias de dimensões no modelo de instânci
⊟ Desenhar instâncias de atributos de dimensões
Desenhar instâncias de atributos no modelo de instâncias.
⊟ Desenhar instâncias de níveis
Desenhar instâncias de níveis no modelo de instâncias.
⊟ Desenhar atributos de níveis
Desenhar instâncias de atributos de níveis no modelo de instâncias.
⊟ Desenhar hierarquias
Desenhar instâncias de hierarquias no modelo de instânci
⊟ Desenhar atributos de hierarquias
Desenhar instâncias de atributos de hierarquias no modelo de instâncias.

Adicionar associações

⊟ Associar dimensões a seus atributos
Associar dimensões a seus atributos.
⊟ Associar dimensões a níveis
Associar dimensões a atributos.
⊟ Associar níveis a seus atributos
Associar níveis a seus atributos.
⊟ Associar níveis a hierarquias
Associar níveis a hierarquias.
⊟ Associar hierarquia a seus atributos
Associar hierarquia a seus atributos.

A.2.3.4-Atividade definir técnicas de data mining e análise espacial

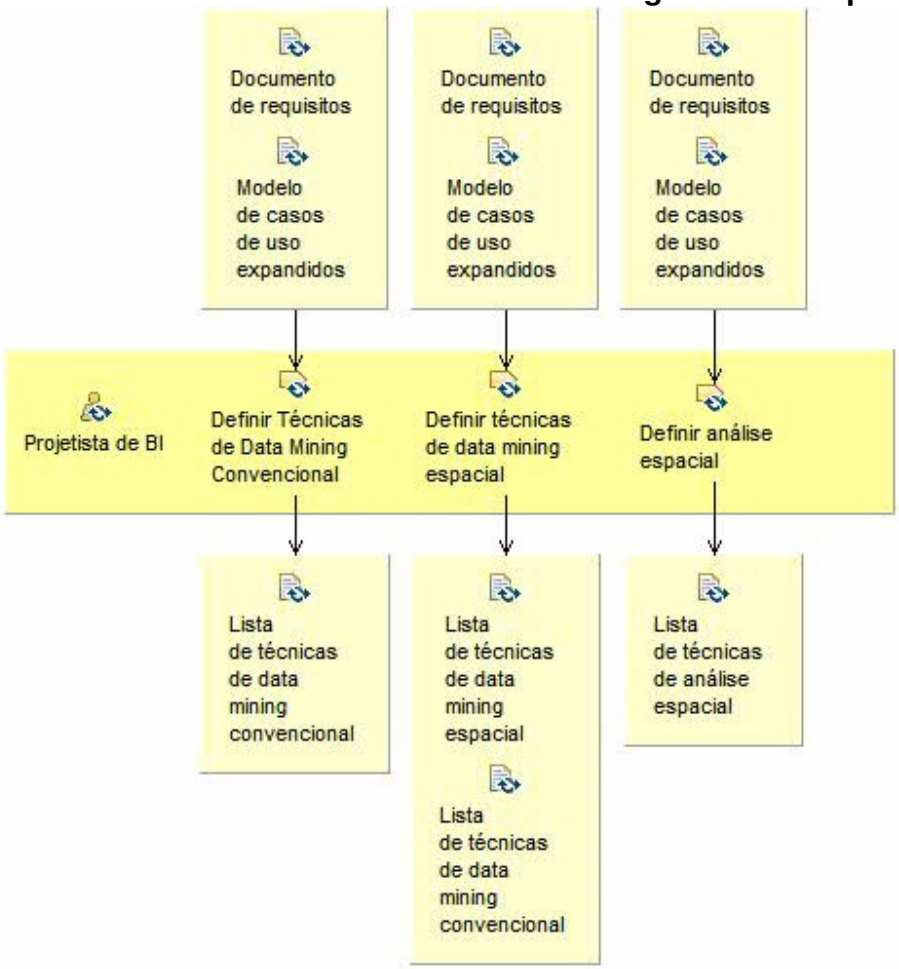


Figura 136- Atividade definir técnicas de data mining e análise espacial

Definir técnicas de data mining convencional

⊟ Definir técnicas de data mining
Com base nas necessidades de análise do negócio definidas anteriormente, determinar as técnicas de data mining necessárias ao atendimento das mesmas:

Obtenção de classificação de dados
Obtenção de regras de associação entre dados
Obtenção de agrupamentos de dados

⊟ Definir dados necessáios
Definir dados necessários à utilização das técnicas de *data mining* definidas no passo anterior.

Definir técnicas de data mining espacial

Verificar particularidades espaciais

Acrescentar critérios de distância aos algoritmos de *data mining* anteriormente definidos,

Definir técnicas de data mining

Com base nas necessidades de análise do negócio definidas anteriormente, determinar as técnicas de data mining necessárias ao atendimento das mesmas:

Obtenção de classificação de dados
Obtenção de regras de associação entre dados
Obtenção de agrupamentos de dados
Regressão de dados

Definir dados necessáios

Definir dados necessários à utilização das técnicas de *data mining* definidas no passo anterior.

Definir análise espacial

Definir camadas de mapeamento necessárias

Nesta etapa são definidas camadas de mapeamento nas quais os dados serão visualizados.

Definir métodos estatísticos quantitativos

Nesta etapa definem-se métodos de estatística descritiva e de análise multivaria sobre os dados (média, variância, desvio-padrão, análise de componen principais, dentre outros).

Definir métodos de análise espacial

Neste passo, métodos de análise espacial são definidos, de forma a realizarem-se inferências sobre os dados. Tais inferências podem ser das seguintes naturezas:

De variação contínua, onde determinam-se superfícies contínuas nas quais a probabilidade de ocorrência de um evento é maior.

De variação discreta, em que determinam-se vários agrupamentos discretos dados por área.

Pontual, onde determinam-se locais específicos onde a probabilidade de ocorrência é maior.

Definir dados necessários

Com base nos métodos de análise definidos anteriormente, selecionar dados necessários (atributos, fatos ou medidas).

A.2.4-A disciplina Projeto Arquitetural da Aplicação

Esta disciplina envolve o projeto das três camadas (apresentação, controle e persistência) que definem a infraestrutura proposta neste trabalho.

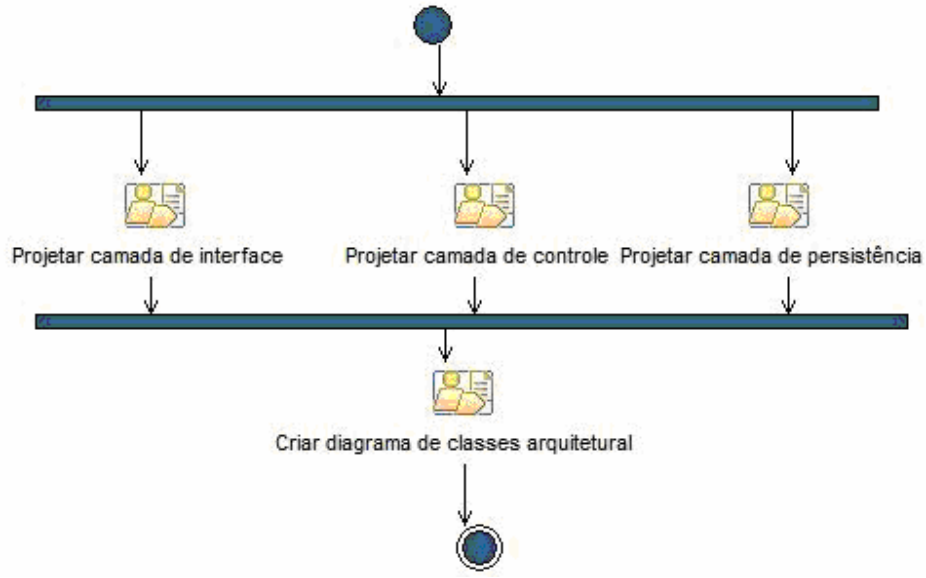


Figura 137-Diagrama de atividades da disciplina Projeto Arquitetural da Aplicação

A.2.4.1-Atividade projetar camada de interface

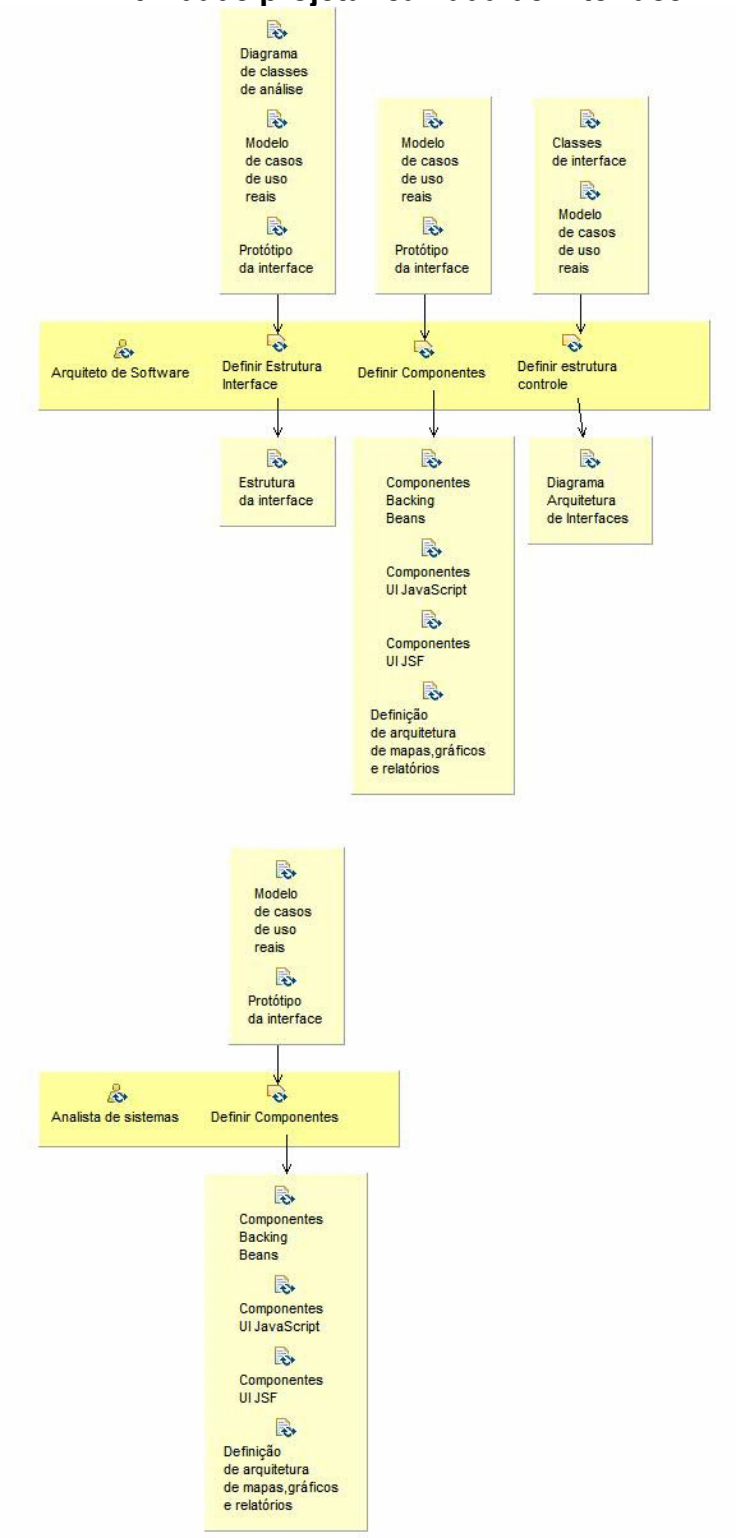


Figura 138- Atividade projetar camada de interface

Definir estrutura interface

- Definir tipo de interface

Definir tipo e natureza da interface: interface web local (Applet), inteface web com JSP/JSF, interface web com JSP e JavaScripit.

- Definir estrutura propriamente dita

Definir a estrutura por meio de *frames*, utilizando-se o *framework* para *design* de interfaces com o usuário. A estrutura será definida de acordo com o protótipo da interface criado na etapa de análise e especificação. Podem-se utilizar os *frames* originalmente propostos pelo *framework,* onde cada *frame* contém arquivos JSP ou HTML (com ou sem Java Script). A figura abaixo exibe o *layout* padrão sugerido.

<table>
<tr><td>Logo</td><td>Header</td><td>Header2</td></tr>
<tr><td colspan="3">Band</td></tr>
<tr><td>LeftSide1</td><td rowspan="4">Center</td><td>RightSide1</td></tr>
<tr><td>LeftSide2</td><td>RightSide2</td></tr>
<tr><td>LeftSide3</td><td>RightSide3</td></tr>
<tr><td>LeftSide4</td><td>RightSide4</td></tr>
<tr><td colspan="3">Footer</td></tr>
</table>

Definir componentes

- Definir tipo de interface

Definir tipo e natureza da interface: interface web local (Applet), inteface web com JSP/JSF, interface web com JSP e JavaScripit.

- Definir estrutura propriamente dita

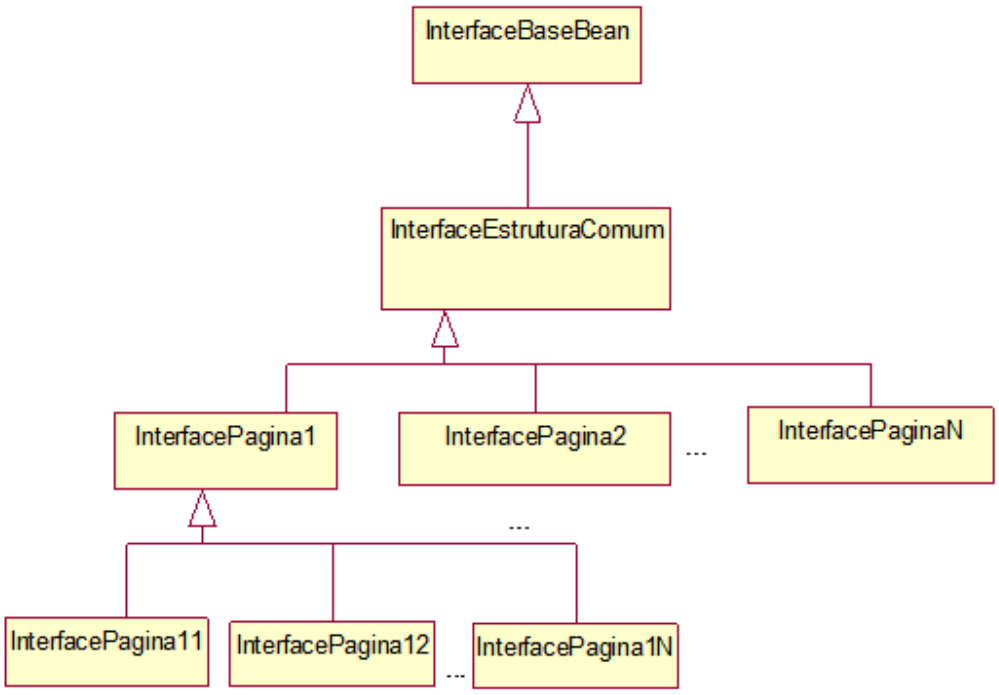
Definir a estrutura por meio de *frames*, utilizando-se o *framework* para *design* de interfaces com o usuário. A estrutura será definida de acordo com o protótipo da interface criado na etapa de análise e especificação. Podem-se utilizar os *frames* originalmente proposto pelo *framework,* onde cada *frame* contém arquivos JSP ou HTML (com ou sem Java Script). A figura abaixo exibe o *layout* padrão sugerido.

Logo	Header	Header2
Band		
LeftSide1	Center	RightSide1
LeftSide2		RightSide2
LeftSide3		RightSide3
LeftSide4		RightSide4
Footer		

Definir estrutura controle

Definir hierarquia de classes de interface

Classes de interface seguem uma hierarquia de dependência entre as mesmas, d forma a serem reutilizados recursos, ou seja, recursos de classes em níveis hierárquicos superiores são reutilizadas nas classes de hierarquia inferior.



Alocar interfaces a classes de interface

Neste passo as interfaces projetadas com arquivos HTML ou JSP são relacionad às respectivas classes de interface que as invocam.

A.2.4.2-Atividade Projetar Camada de Controle

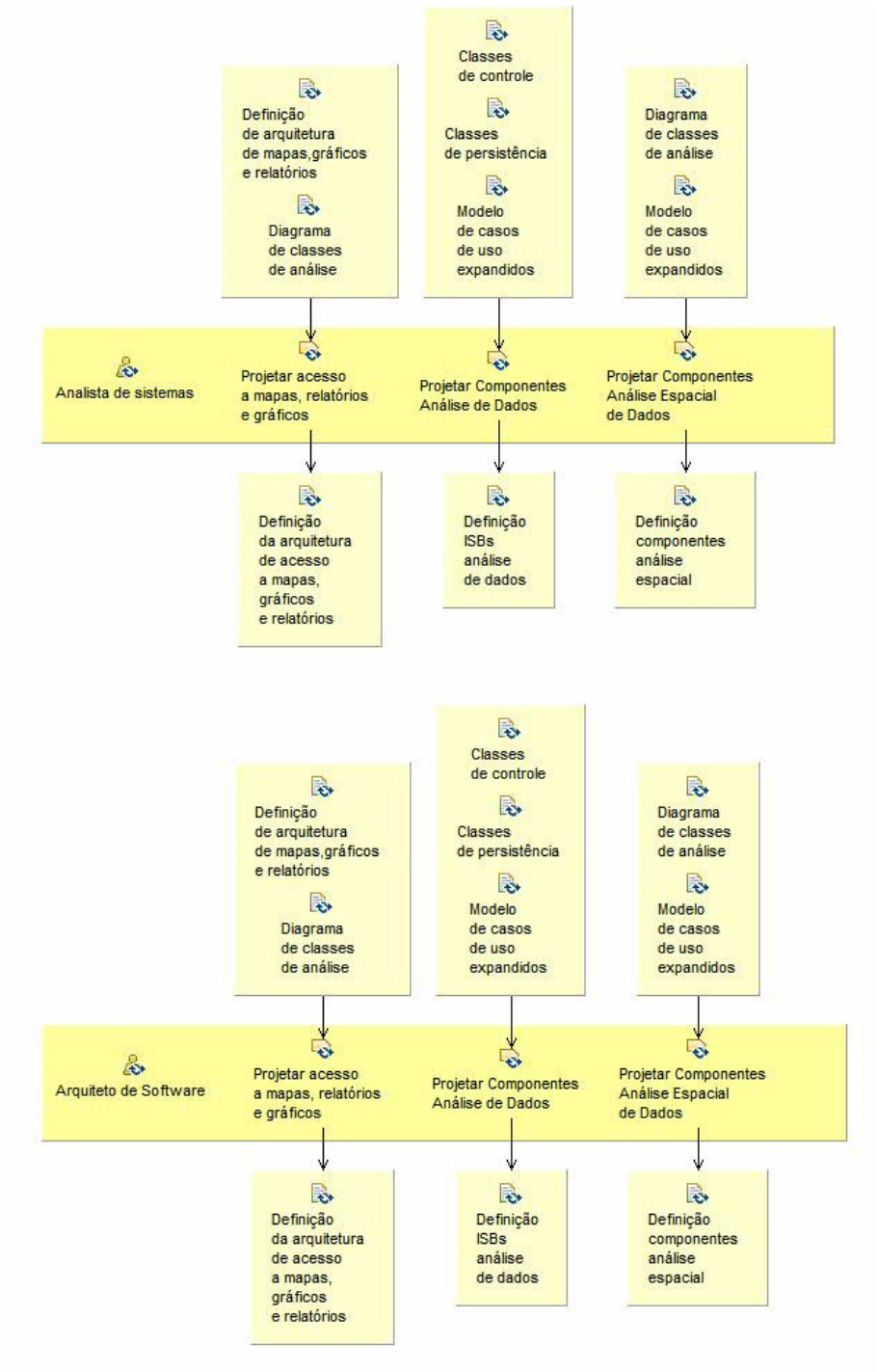


Figura 139- Atividade Projetar Camada de Controle

Projetar acesso a mapas, relatórios e gráficos

⊟ Definir arquitetura para acesso a mapas
Nesta etapa define-se se acesso aos serviços WMS (implementados como *Sessior Beans)* será feito via w*eb services* ou via *servlets*.

⊟ Definir arquitetura para acesso a gráficos
Definir acesso a gráficos via *sevlets* ou via *web servic*
⊟ Definir arquitetura para acesso a relatórios
Definir acesso a relatórios via *sevlets* ou via *web servic*

Projetar componentes análise de dados

⊟ Definir ISBs de mineração de dados convencional
Com base nos requisitos de análise de dados definidos em etapa anteriores, defir os componentes ISB (*session beans)* de mineração de dados que serão utilizados na aplicação.
⊟ Definir ISBs de análise OLAP convencional
Com base nos requisitos de análise de dados definidos em etapa anteriores, escolher ISBs de OLAP convencional que serão utilizados na aplicação.

Projetar componentes análise espacial de dados

⊟ Definir ISBs de mineração e análise espacial de dados
Com base nos requisitos de análise de dados definidos em etapa anteriores, defir os componentes ISB (*session beans)* de mineração e análise espacial de dados c serão utilizados.
⊟ Definir ISBs de análise OLAP espacial
Com base nos requisitos de análise de dados definidos em etapa anteriores, escolher ISBs de OLAP espacial que serão utilizados na aplicação.

A.2.4.3-Atividade Projetar camada de persistência

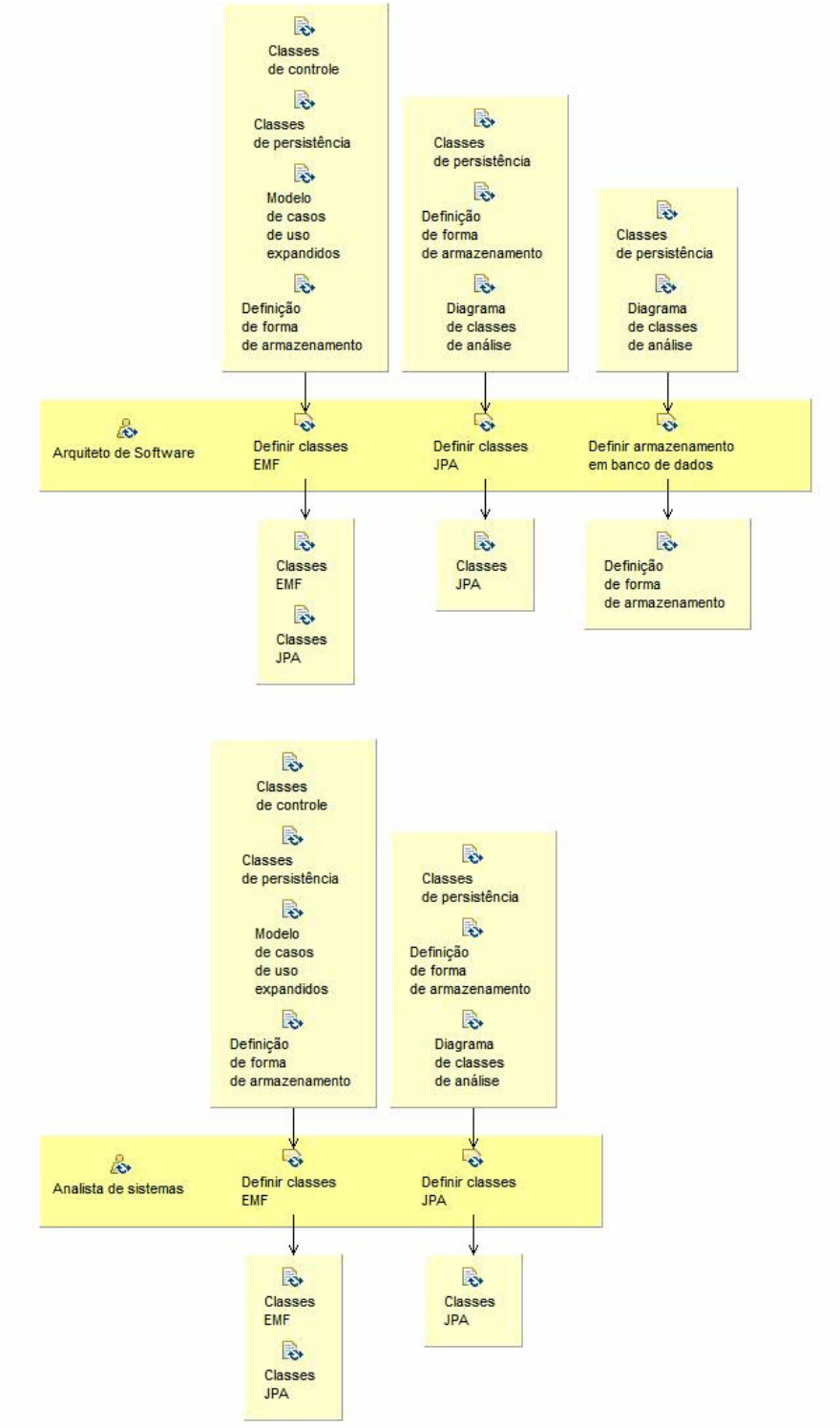


Figura 140- Atividade Projetar camada de persistência

Definir classes EMF

⊟ Projetar acesso a classes feature OGC
Projetar acesso classes EMF do modelo features OC
⊟ Projetar modelo Ecore classes de domínio de problema
Projetar modelo Ecore de classes provenientes dos diagramas de classes de domínio do problema. Este modelo é gerado automaticamente no ambiente Eclipse/EMF, a partir do diagrama de classes definido na etapa de análise.
⊟ Projetar acesso a classes do CWM-G
Projetar acesso a classes CWM-G utilizadas no mode

Definir classes JPA

⊟ Definir estrutura de classes JPA
Definir a estrutura de classes JPA segundo o padrão EMF/JI

Definir armazenamento em banco de dados
⊟ Selecionar objetos para persistência em EMF/XMI
Objetos que não necessitem de um controle via banco de dados convencionais podem ser pesistidos por meio de arquivos XMI, no *framework* Ecore do EMF.
⊟ Selecionar objetos para persistência em banco de dados via JPA
Objetos que necessitem de CRUD em bancos de dados, bem como de *queries* para consultas estruturadas, são armazenados por meio de tecnologia JPA integrada ao EMF/Ecore (tecnologia TENEO do EMF).

A.2.4.4. Atividade criar diagrama de classes arquitetural

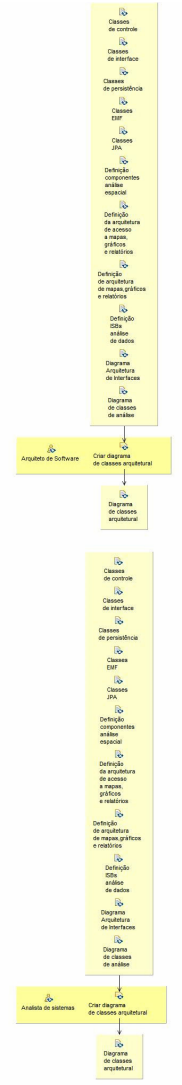


Figura 141- Atividade criar diagrama de classes arquitetural

Criar diagrama de classes arquitetural

⊟ Acrescentar classes de interface
Ao diagrama de classes de análise, acrescentar classes de interface com particularidades da implementação definida.
⊟ Acrescentar classes de controle
Ao diagrama de classes de análise, acrescentar classes de controle com particularidades da implementação definida.
⊟ Acrescentar classes de persistência

Ao diagrama de classes de análise, acrescentar classes de persistência com particularidades da implementação definida.
⊟ Criar associações
Definir novas associações resultantes da introdução de particularidades de projeto.

A.3-A Etapa Construção

Esta etapa envolve disciplinas relacionadas à implementação da aplicação, ao teste da mesma e à importação de dados.

A.3.1-A Disciplina Migração de Dados

Esta disciplina envolve atividades de extração, transformação e carga de dados convencionais e espaciais.

Figura 142-Diagrama de atividades da disciplina Migração de Dados

A.3.1.1-Atividade extração

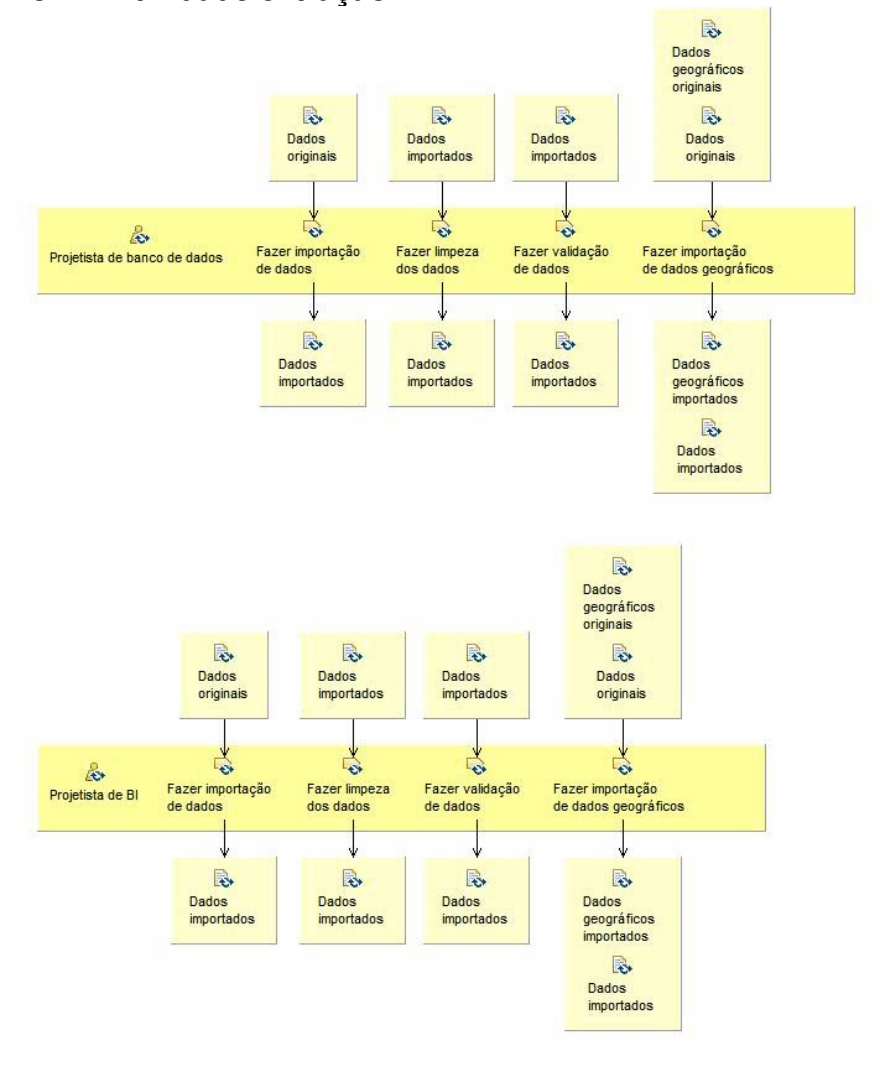


Figura 143- Atividade extração

Fazer importação de dados

⊟ Preparar fontes
Deixar as fontes prontas para que o processo de importação seja feito. Por exemplo, caso seja um arquivo utilizado por outra aplicação, deixá-lo

desbloqueado para que sua leitura seja feita livremente.
⊟ Fazer importação propriamente dita
As principais questões a serem levadas em consideração são as seguintes:

Importação de formatos distintos de arquivos, que podem estar estruturados em bancos de dados, em planilhas, arquivos texto, ou até mesmo em formatos complexos como vídeo, áudio, imagens, etc.

Importação de arquivos provenientes de tecnologias diferentes, como por exemplo, sistemas operacionais diferentes;

Associação de arquivos estruturalmente diferentes, com granularidades temporalidades distintas;

Dados podem ser estruturalmente complexos, sem qualquer grau de normalização.

Deve-se manter informação de temporalidade dos dados armazenados, de forma a se evitarem duplicidades.

Deve-se manter o mapeamento entre nomes de atributos nos arquivos de origem e de destino.

Fazer limpeza dos dados

⊟ Uniformizar abreviações e sinônimos

niformizar abreviações (Av. e Avenida, por exemplo) , presença ou não preposição(Av.Contorno e Av. do Contorno, por exemplo), sinônimos (gênero sexo por exemplo), dentre outras.

⊟ Uniformizar formatos

Uniformização de formatos de datas, horas, dias, dentre out

⊟ Compatibilizar dados

ompatibilização de dados, como por exemplo, converter unidades de medi

⊟ Verificar dependência entre dados

Compatibilização de dados, como por exemplo, converter unidades de medi

⊟ Verificar caracteres

Retirar caracteres espúrios ou desnecessários, como por exemplo, espaç

Fazer validação dos dados

⊟ Validar dados

Trata-se da verificação das duas últimas tarefas, no que diz respeito à verificação da presença de todos os dados necessários, busca por mudanças bruscas, verificação de uniformidade de versão , verificações de consistência e de uniformidade das agregações.

Fazer importação de dados geográficos

⊟ Fazer geocodificação

Neste passo são realizadas operações de geocodificação de dados por meio de no mínimo 3 pontos de mapeamento.

⊟ Fazer associações de dados convencionais a dados geográficos

Consiste na junção de dados convencionais a objetos geográficos presentes

mapas armazenados em GIS. Nesse caso, os dados convencionais e os objetos geográficos devem possuir uma chave comum que permita associá-los a um banco de dados comum (por exemplo, associar uma base de dados sóc econômicos de países a um mapa mundi, pelo código do país).

- Preparar fontes

Deixar as fontes prontas para que o processo de importação seja feito. Por exemplo, caso seja um arquivo utilizado por outra aplicação, deixá-lo desbloquea para que sua leitura seja feita livremente.

- Fazer importação propriamente dita

As principais questões a serem levadas em consideração são as seguintes:

Importação de formatos distintos de arquivos, que podem estar estruturados em bancos de dados, em planilhas, arquivos texto, ou até mesmo em formatos complexos como vídeo, áudio, imagens, etc.

Importação de arquivos provenientes de tecnologias diferentes, como por exemplo, sistemas operacionais diferentes;

Associação de arquivos estruturalmente diferentes, com granularidades ou temporalidades distintas;

Dados podem ser estruturalmente complexos, sem qualquer grau de normalização.

Deve-se manter informação de temporalidade dos dados armazenados, de forma a se evitarem duplicidades.

Deve-se manter o mapeamento entre nomes de atributos nos arquivos de origem e de destino.

A.3.1.2-Atividade transformação

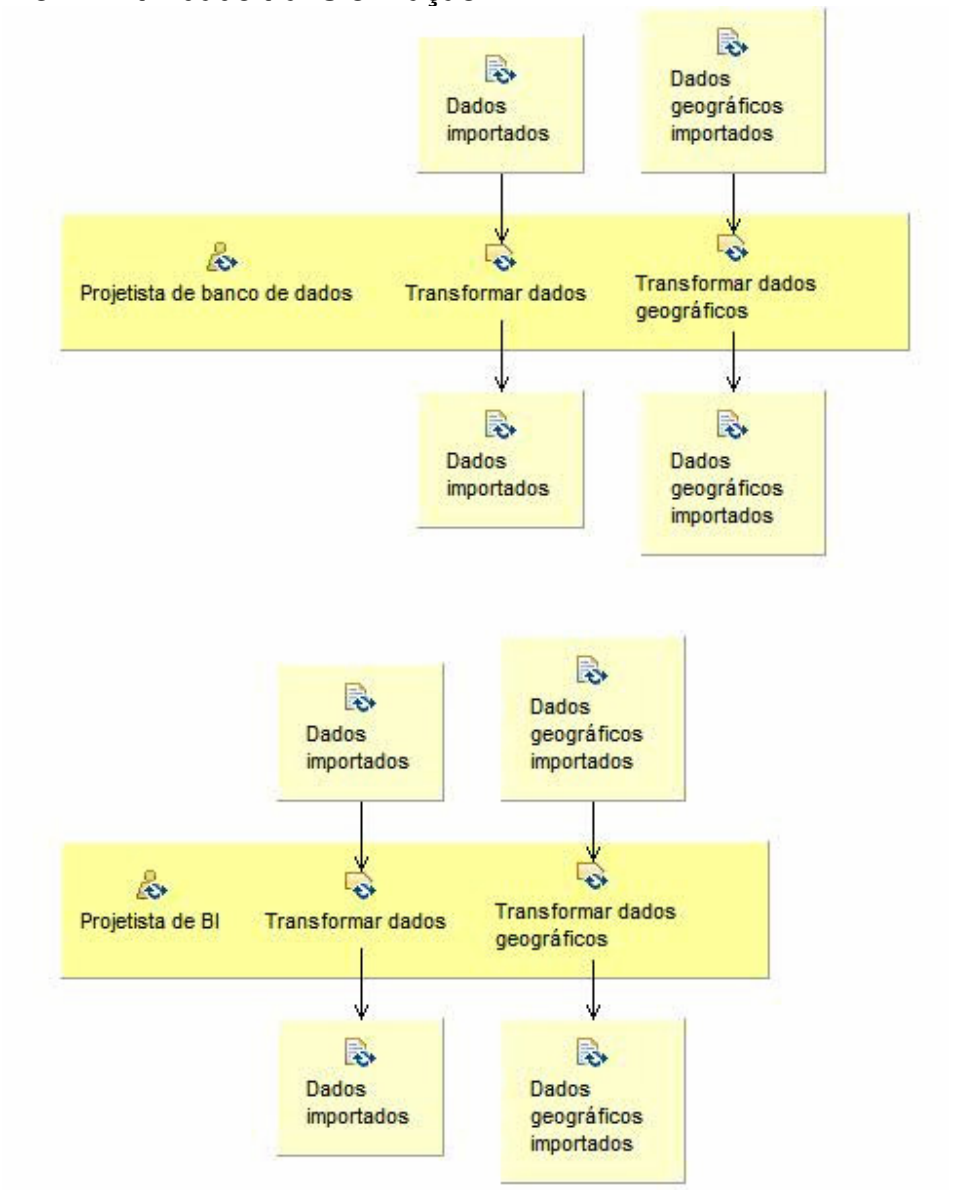


Figura 144- Atividade transformação

Transformar dados

☐ Particionar dados
Particionamento dos dados de acordo com um critério qualquer de seleç
☐ Tratar atributos
Extração ou divisão dos atributos dos dados ou das tabelas dos modelos diversas tabelas que serão utilizadas em diversos contextos.

Concatenação de atributos pertencentes a várias tabelas sob determinado contexto.

Enriquecimento, que consiste em aumentar a complexidade de um atribu Como exemplos, podemos citar a junção de atributos, a mudança de forma dentre outros.

Conversão de um tipo de dado de um atributo para outro.

Remoção de atributos que não serão úteis

⊟ Desnormalizar dados

Desnormalização", que consiste em juntar informações de diversas tabelas em uma s criando redundâncias no modelo relacional de forma a "desnormalizá-lo". Este proces ainda que aumente o uso de espaço em disco, permite maior velocidade de recupera da informação. O processo inverso, a normalização, pode ocorrer em data warehou mas é muito menos comum.

⊟ Construir consultas

Construção de consultas que resultarão na agregação de todo, ou parte, do banco de d warehouse em uma dimensão (tempo, ou também outra qualquer). Os dados devem po ser desagrupados para serem vistos em resumo ou em detalhe. Estes processos s denominados RollUp e DrillDown, respectivamente. Denomina-se nível de granulariza ao menor nível de detalhe que pode ser alcançado em uma consulta. Em agregação dados, perde-se um nível de informação que só pode ser recuperado através dos da originais, mas isto custa memória e desempenho. Quanto maior a granularização ta menor o desempenho.

⊟ Combinar fontes de dados

Combinação de fontes de dados, pelo casamento exato de valores chaves ou pel busca em atributos não chaves, como a procura por equivalências textuais.

⊟ Criar chaves substitutas

Criação de chaves substitutas em dimensões para diminuir a dependência chaves legadas, de forma a reforçar a integridade entre tabelas de dimensõe tabelas de fatos.

⊟ Filtrar e interpolar

Filtragem de dados irrelevantes ou, interpolações e extrapolações para aproximação de dados não presentes

Transformar dados geográficos

Fazer junção espacial

Combina duas geometrias espaciais de acordo com predicados topológicos (vizinhança, por exemplo), métricos (distância, por exemplo) ou de ordem (ao lad de, por exemplo) (Fornary,2006). Por exemplo, realizar a junção espacial entre a feição geográfica que representa um município e as feições que representam rodovias segundo o predicado intercepta(rodovia,município). (Orenstein, 1996) sugere que a junção espacial seja feita em duas etapas: a primeira, denominada filtragem, testa os predicados para aproximações de objetos, como o envelope M (Minimum Bounding Rectangle),obtendo-se pares de objetos candidatos; a segunda etapa, de refinamento, verifica o predicado espacial para a geometria completa de cada objeto dos pares selecionados na etapa anterior. Essa abordagem pode gerar falsos candidatos, por isso alguns autores sugerem refinamentos, como (Brinkhoff, 1993) e (Kriegel,2003), que usam polígonos de 5 lados para identificar objetos que não se interceptam e (Azevedo, 2004), que usa aproximações raster.

Fazer agregação espacial

Consiste na agregação de geometrias espaciais em uma só geometria espacial resultante. Sistemas de bancos de dados com extensões espaciais realizam esta operação corriqueiramente.

Fazer mudança do sistema de projeção

É usual a necessidade de conversão entre sistemas de projeção, principalmente entre os mais comuns, latitude-longitude e UTM (e vice-versa). SIGs (MapInfo e ARCGIS, por exemplo) realizam essa operação correntemente, portanto, muitas vezes essa operação é realizada durante o processo de extração.

A.3.1.3-Atividade carga

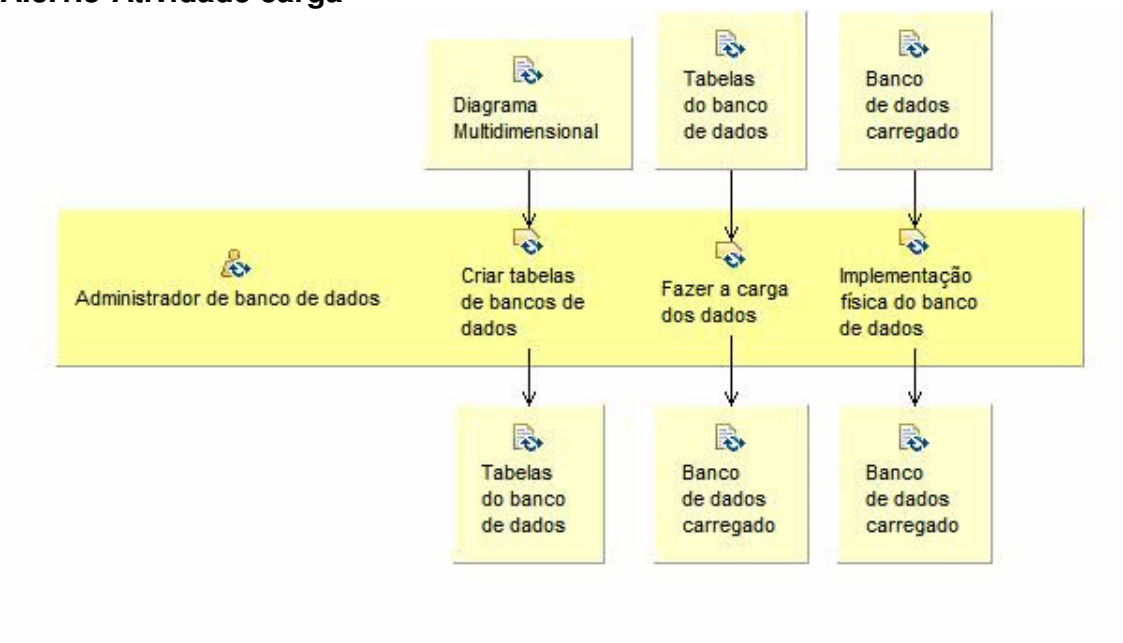


Figura 145- Atividade carga

Criar tabelas de bancos de dados

⊟ Criar tabelas de banco de dados
Fazer a criação das tabelas físicas no banco de dac

Fazer a carga dos dados
⊟ Fazer a carga dos dados
Consiste no processo de efetivamente carregar os dados (tabelas de dimensões e fatos) no data warehouse. Tais dados devem ser devidamente indexados de forma aumentar o desempenho de consultas.

Implementação física do banco de dados

⊟ Definir estrutura de armazenamento físico
Definir estrutura de armazenamento físico necessárias ao volume e estrutura dos dados a serem armazenados.
⊟ Ajustar parâmetros do banco de dados
Ajustar parâmetros de configuração do banco de dac

A.3.2-A disciplina Implementação

Esta disciplina envolve atividades relacionadas à implementação das camadas de apresentação, controle e persistência da aplicação.

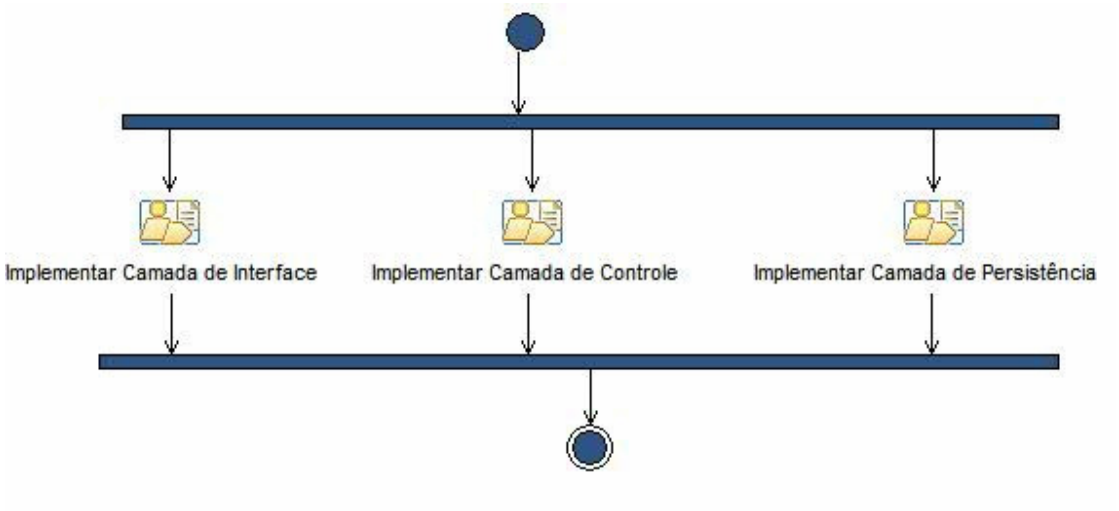


Figura 146-Diagrama de atividades da disciplina Implementação

A.3.2.1-Atividade implementar camada de interface

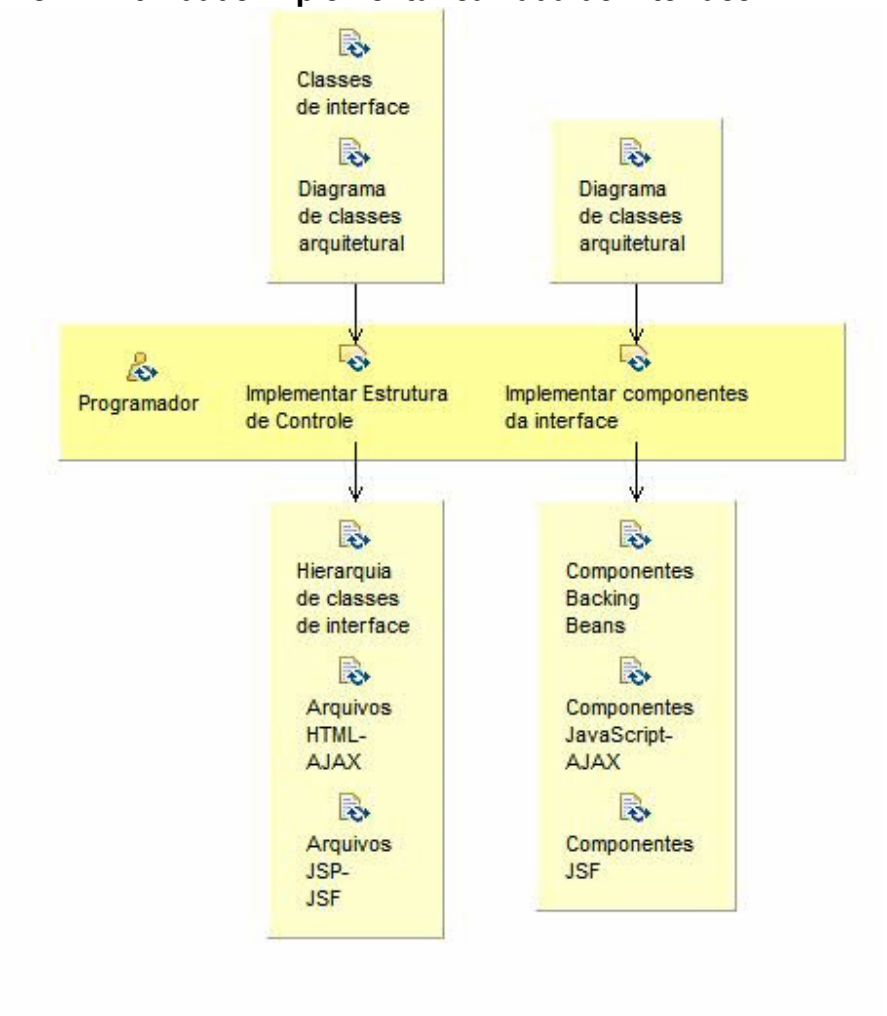
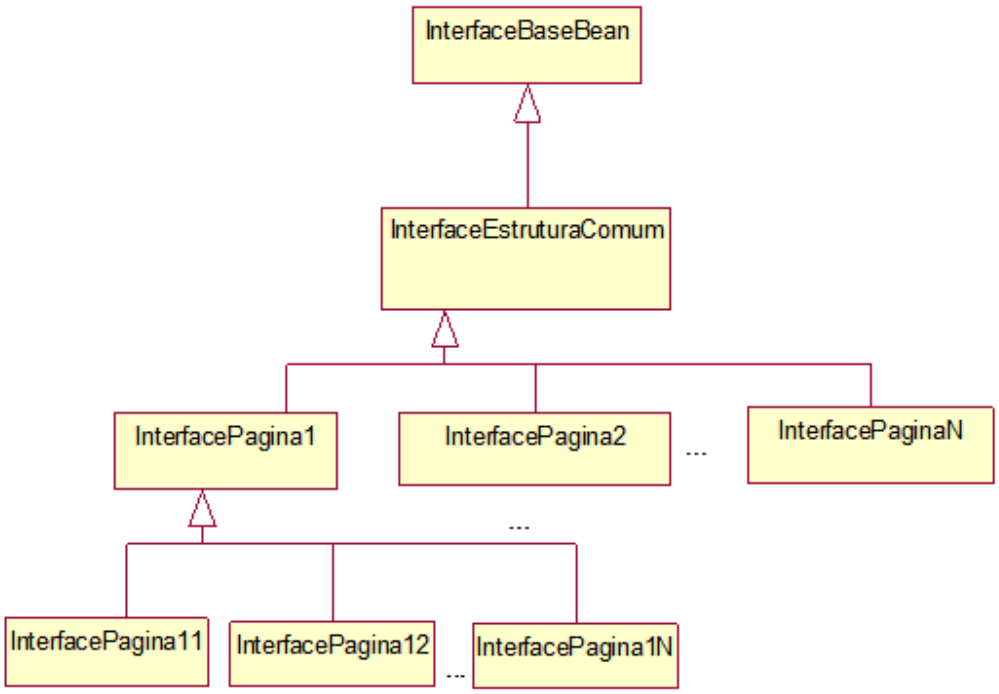


Figura 147- Atividade implementar camada de interface

Implementar estrutura de controle

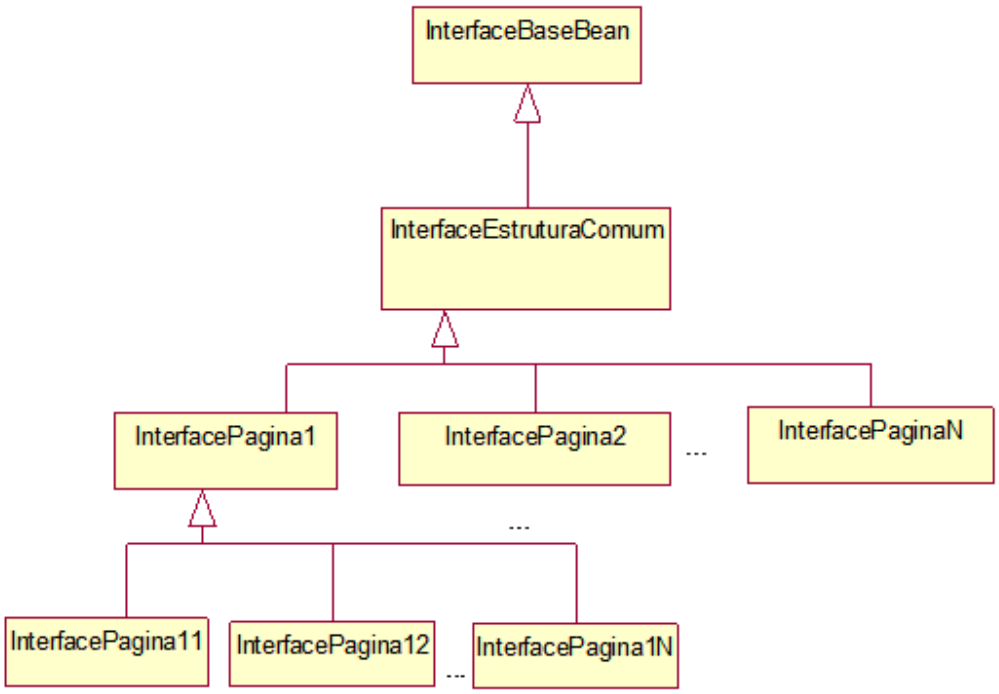
⊟ Implementar Hierarquia de Classes InterfaceBaseBean
Implementar todas as extensões da classe IntefaceBaseBean de forma a abstrair hierarquia definida no projeto.



Implementar Interfaces projetadas.
Nesta etapa implementam-se todas as interfaces projetadas (JSP,JSF, Java Scrip
HTML, applets ou aplicações Java convencioanais). Cada interface projetada
associa-se a uma classe de interface IntefaceBaseBean.

Implementar componentes da interface

Implementar Hierarquia de Classes InterfaceBaseBean
Implementar todas as extensões da classe IntefaceBaseBean de forma a abstrair
hierarquia definida no projeto.



Implementar Interfaces projetadas.
Nesta etapa implementam-se todas as interfaces projetadas (JSP,JSF, Java Scrip
HTML, applets ou aplicações Java convencioanais). Cada interface projetada
associa-se a uma classe de interface IntefaceBaseBean.

A.3.2.2-Atividade implementar camada de controle

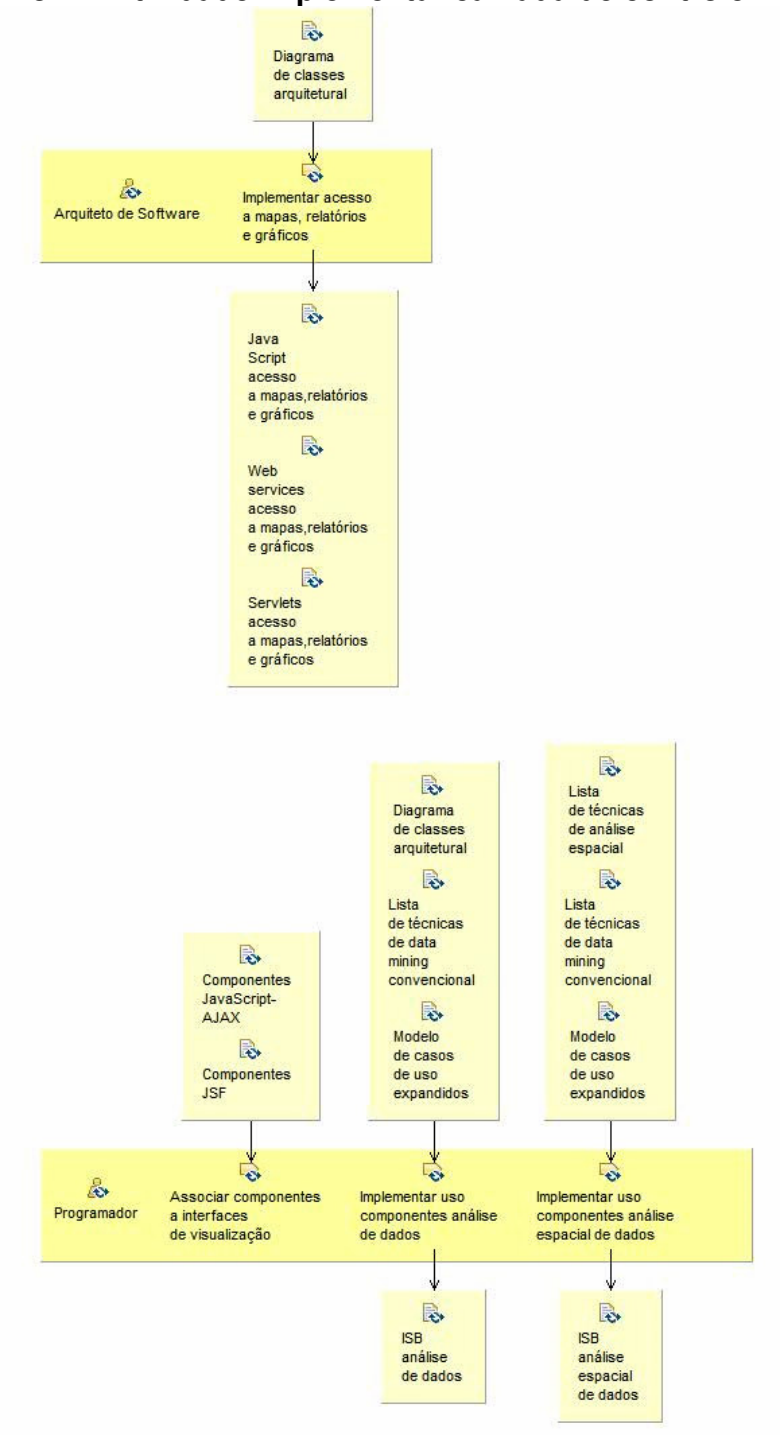


Figura 148- Atividade implementar camada de controle

Implementar acesso a mapas relatórios e gráficos

⊟ Implementar acesso a mapas
Visualização de informação em mapas é realizada por meio da ISBWMS *(*Sessior Bean que implementa o serviço WMS do OGC)*,* que pode ser acessada por meio de:

Java Script: em especial, acessa a API do Google Maps e disponibiliza interface para interação com o mapa.
Servlets: geram páginas dinámicas com acesso à API do Google Maps e disponibilização da interface para interação com o mapa.

⊟ Implementar acesso a relatórios
Relatórios são implementados por meio de:

Servlets: exibem o relatório gerado dinamicamente após acesso a um web service que recebe as requisições e retorna os dados necessários.
Código Java Script/AJAX: envia requisições a servlets ou *web services* que retornam os dados necessários.

⊟ Implementar acesso a gráfios
Gráficos são implementados por meio de:

Servlets: exibem o gráfico gerado como image após acesso a um componen gráfico que recebe as requisições e retorna os dados necessários.
Código Java Script/AJAX: envia requisições a servlets ou *web services* que retornam o gráfico como uma imagem.

Associar componentes a interfaces de visualização
⊟ Associar componentes a classes de interface
Componentes JSP./JSF, Java Script são associados às classes de interfa

Implementar uso componentes análise de dados
⊟ Associar ISBs de mineração de dados a servlets ou web services
ISBs de mineração de dados a serem utilizados na aplicação são associados a servlets ou web services que vão disponibilizar sua funcionalidade para ser visualiada em mapas, relatórios ou gráficos.
⊟ Associar ISBs de análise OLAP a servlets ou web services
Os ISBs definidos a serem utilizados em análise OLAP de dados são associados a servlets ou web services, que vão disponibilizar as informações fornecidas por eles principalmente em mapas, mas também em relatórios e gráficos.

Implementar uso componentes análise espacial de dados

⊟ Associar ISBs de mineração análise espacial de dados a servlets ou web services

Os ISBs definidos a serem utilizados na mineração ou análise espacial de dados sã associados a servlets ou web services, que vão disponibilizar as informações fornecidas por eles principalmente em mapas, mas também em relatórios e gráficos

⊟ Associar ISBs de análise OLAP espacial a servlets ou web services

Os ISBs definidos a serem utilizados em análise OLAP espacial de dados são associados a servlets ou web services, que vão disponibilizar as informações fornecidas por eles principalmente em mapas, mas também em relatórios e gráficos

A.3.2.3-Atividade implementar camada de persistência

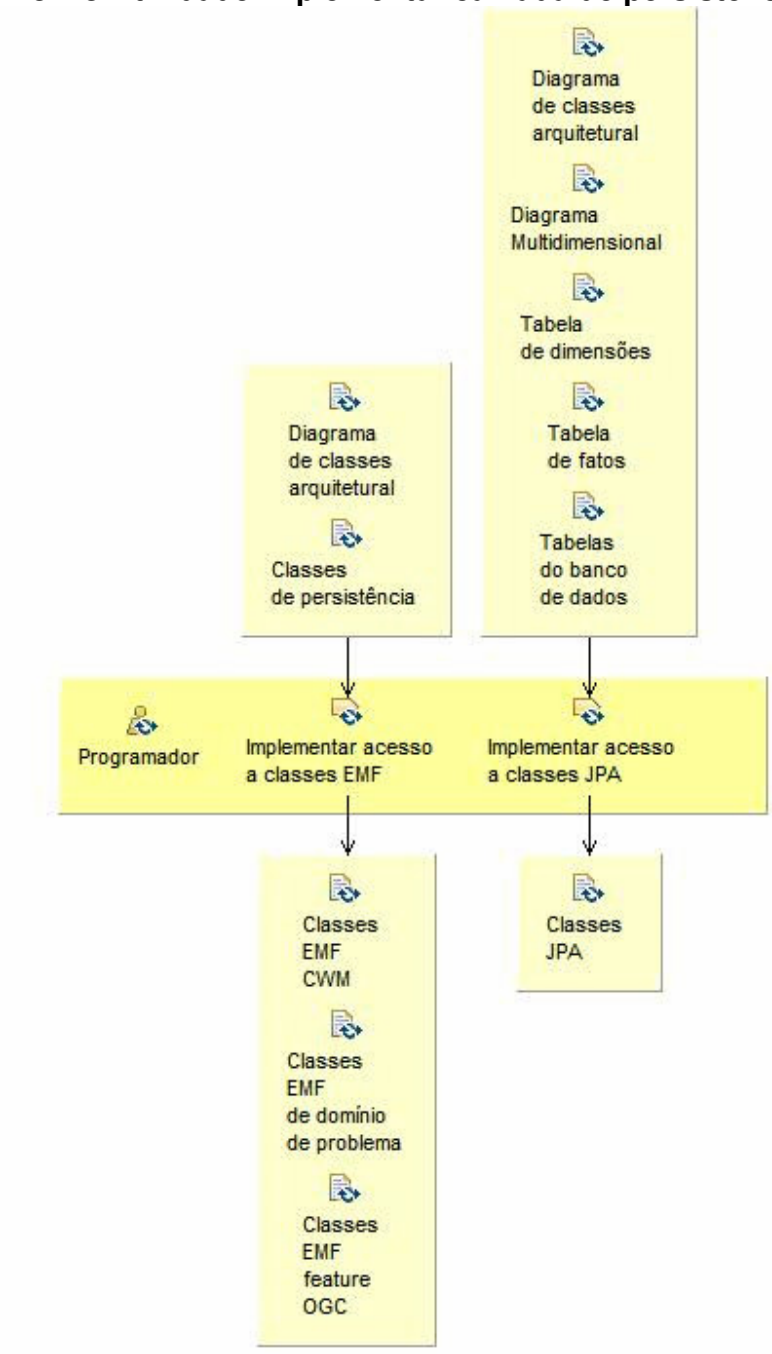


Figura 149- Atividade implementar camada de persistência

Implementar acesso a classes EMF

⊟ Implementar acesso a classes feature OGC
Este passo consiste em criar instâncias das classes *feature OGC* que serão utilizadas na aplicação principalmente em dimensões e medidas espaciais. As classes *feature OGC* são implementadas segundo a estrutura do *framework* Ecor do Eclipse EMF.
⊟ Implementar modelo Ecore das classes de domínio de problema
Este passo consiste em criar instâncias das classes de domínio do problema que serão utilizadas na aplicação principalmente em dimensões e medidas espaciais. Tais são implementadas segundo a estrutura do *framework* Ecore do Eclipse EMF.
⊟ Implementar acesso a classes do modelo CWM-G
Este passo consiste em criar instâncias das classes *CWM-G* que serão utilizadas aplicação principalmente em dimensões e medidas espaciais. As classes *feature OGC* são implementadas segundo a estrutura do *framework* Ecore do Eclipse EM
⊟ Implementar acesso a classes de domínio de problemas utilizadas nos métodos de análise
Implementar é implementado acesso a classes de domínio de problemas utilizada nos métodos de análise.

Implementar acesso a classes JPA
⊟ Criar classes Java JPA/EMF
Criar classes JPA/EMF a serem associadas a tabelas de bancos de dac
⊟ Associar classes ao BD
Associar classes que serão persistidas, a um banco de dad
⊟ Associar atributos a colunas de tabelas
Associar atributos das classes a colunas de tabelas do banco de dad
⊟ Criar associações
Criar associações do tipo um para um, um para muitos ou muitos para muitos, en classes JPA

A.3.4-A disciplina Testes

Esta disciplina envolve atividades relacionadas à definição, realização e verificação de resultados de testes aos níveis de unidade, integração, sistema, verificação, validação e aceitação.

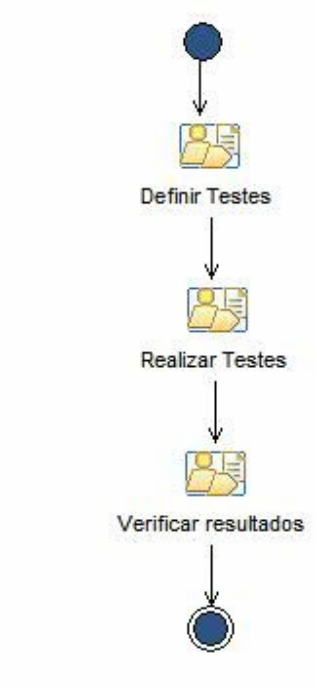


Figura 150-Diagrama de atividades da disciplina Testes

A.3.4.1-Atividade definir testes

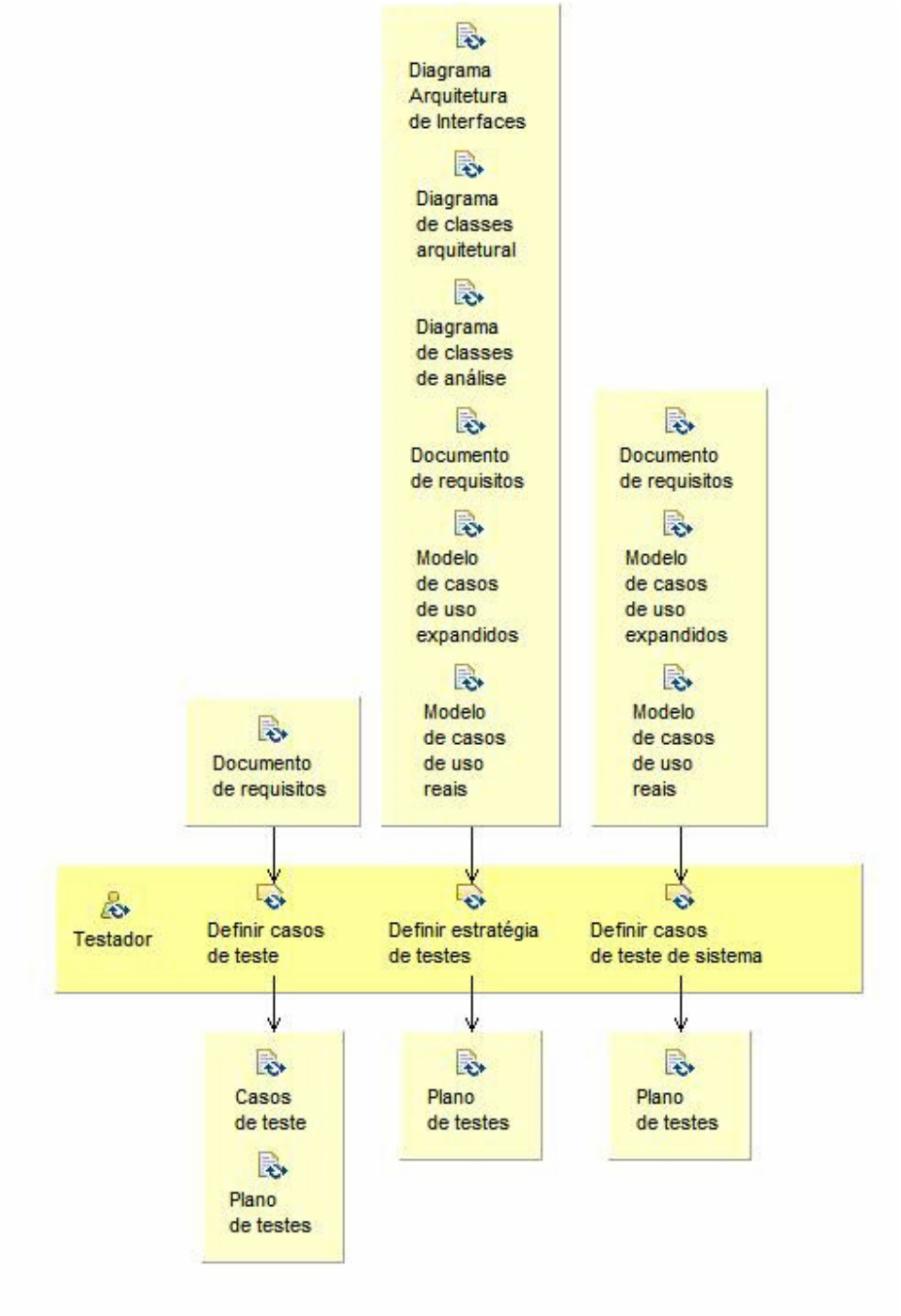


Figura 151- Atividade definir testes

Definir casos de teste

⊟ Definir testes serem feitos
Definir os testes a serem feitos a partir dos requisitos definid

⊟ Definir métodos a serem utilizados.
Definir os métodos mais adequados aos testes definidos no passo anterior. Normalmente, utilizam-se métodos de caixa branca para testes de unidades e testes de caixa preta para testes de integração e validação.
⊟ Gerar casos de teste
Com base nos métodos definidos gerar os casos de teste em nível de unidade, verificação e validação.
⊟ Identificar dados necessários ao teste
Rever cada caso de teste e definir cada entrada necessá

Definir estratégia de testes
⊟ Escolher a estratégia de testes
Definir a forma como os diversos módulos e classes serão testados por relevânci ou por percorrimento da hierarquia entre os diversos módulos (*botton up ou top down).*
⊟ Definir sequência de percorrimento de módulos
De acordo com a estratégia definida no passo anterior, definir a sequência de percorrimento de módulos e consequentemente os casos de teste de integração.

Definir casos de teste de sistema
⊟ Definir casos de teste de desempenho
Definir casos de teste de desempen
⊟ Definir casos de teste de segurança
Definir casos de teste de segura
⊟ Definir casos de teste de stress
Definir casos de teste que coloquem o sistema à prova de queda: exceder capacidade de armazenamento, excesso de dados, de necessidades de processamento, etc.
⊟ Definir testes de recuperação
Definir casos de teste que testem a capacidade do sistema para se recuperar de falhas.

A.3.4.2-Atividade realizar testes

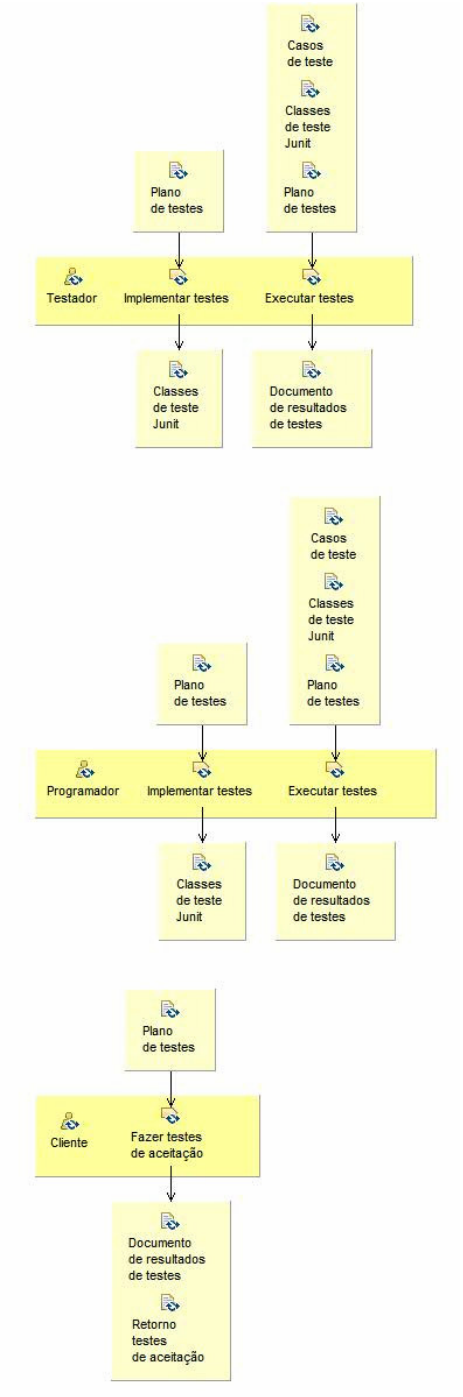


Figura 152- Atividade realizar testes

Implementar testes

Criar classes Junit

Criar classes Junit para implementar casos de teste do sistema.

Criar suites de teste Junit

Alocar testes a suite de testes Junit.

Executar testes

Executar suite de testes de unidades

Executar casos de teste de unidades.

Executar suite de testes de integração

Executar casos de teste de integração.

Executar suíte de testes de sistema

Executar casos de teste de sistema

Executar suíte de testes de verificação

Executar casos de teste de verificação

Executar suíte de testes de validação

Executar casos de teste de validação.

Fazer testes de aceitação

Fazer alfa-testes

Realizar testes de aceitação no ambiente de desenvolcimento, Tais testes são supervisionados e verificam usabilidade, atendimento a requisitos funcionais, completude dos dados e ocorrência de erros.

Fazer beta-testes

Realizar testes de aceitação no ambiente dos usuários finais. Estes utilizam livremente o sistema e periodicamente retornam informações sobre erros, melhor de usabilidade, funcionalidades adicionais, dentre outras.

Avaliar resultados do teste de aceitação

Avaliar resultados dos alfa e beta testes de forma a efetuar correções e melhoramentos.

A.3.4.3-Atividade verificar resultados

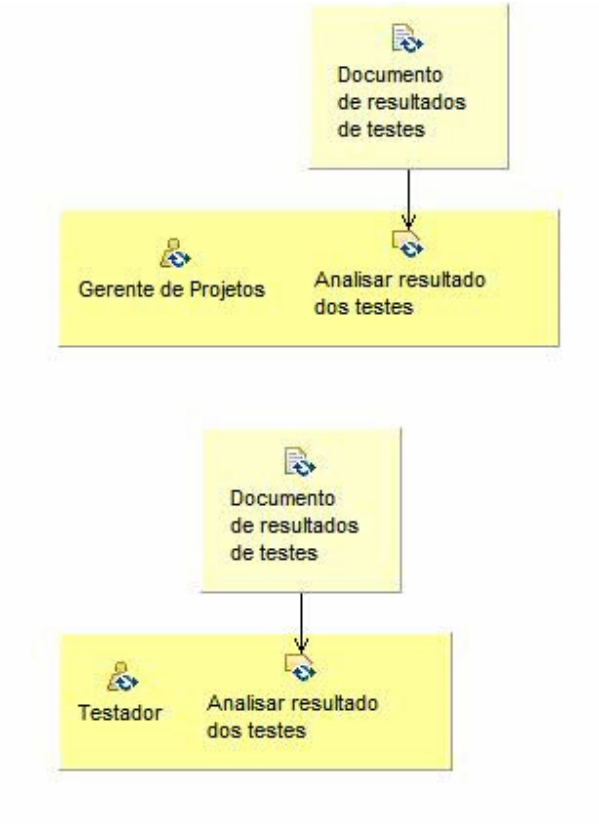


Figura 153- Atividade verificar resultados

Analisar resultado dos testes
⊟ Analisar resultados
Analisar resultados de testes de forma a identificar possíveis falhas.
⊡ Comunicar resultados
Comunicar resultados em um relatório de resultados de testes.
⊟ Submeter resultados para possíveis correções
Submeter resultados a um processo de ações corretivas.

A.4-A Etapa Implantação

A etapa de implantação envolve a implantação da aplicação em seu ambiente operacional.

A.4.1-A disciplina Instalação

A disciplina instação envolve o planejamento e a realização da instalação da aplicação.

Figura 154-Diagrama de atividades da disciplina Instalação

A.4.1.1-Atividade Planejar Instalação

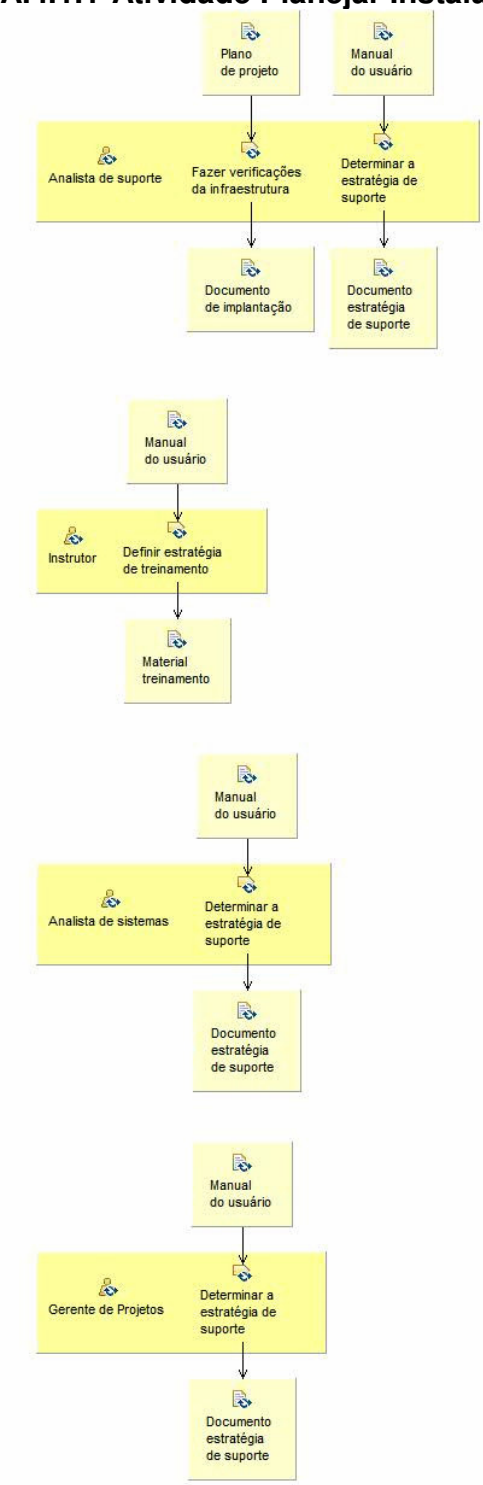


Figura 155- Atividade Planejar Instalação

Fazer verificações da infraestrutura

⊟ Verificar requisitos de configuração
Determinar requisitos de configuração do cliente de forma a suportar acesso a softwre, hardware base de dados e internet.
⊟ Comparar configuração atual com a exigida
Comparar a configuração atual com a exigida de forma a estabelecer necessidad de aquisições e alterações.
⊟ Completar a configuração
Efetuar aquisições de hardware, software e atualizações de forma a completar a configuração exigida.
⊟ Definir segurança de acesso
Definir senhas de acesso e protocolos de segurança para acesso à rede e ao bar de dados.
⊟ Verificar instalação em várias máquinas
Verificar procedimento de instalação em várias máquinas diferentes de forma a capturar possíveis incompatibilidades.
⊟ Fazer cronograma de instalação
Sincronizar o cronograma de instalação com o treinamento de usuári
⊟ Instalar software e hardware necessário
Instalar hardware e software necessários e realizar testes da instalaç

Determinar a estratégia de suporte
⊟ Determinar a estrutura de suporte organizacional
Determinar pessoal necessário, infraestrurutura de hardware e equipamento e material de apoio necessário ao suporte.
⊟ Estabelecer forma de comunicação do suporte
Estabelecer um processo de comunicação rastreável de forma a capturar dúvidas sugestões e críticas.
⊟ Prover documentação de suporte
Criar um web-site para consulta, manuais e guias de referência ráp
⊟ Determinar procedimentos de suporte
Formalizar como o suporte será realizado em termos de procedimentos dos responsáveis pelo suporte.

Definir estratégia de treinamento
⊟ Integrar e adequar conteúdo
O conteúdo deve ser adequado de forma a atender às necessidades de usários finais e gestores dos negócios, separadamente.
⊟ Desenvolver o treinamento do usuário final
Desenvolver o treinamento dos usuários finais levando em conta as adequações definidas no passo anterior.
⊟ Desenvolver o treinamento de gestores do negócio
Desenvolver o treinamento dos gestores de negócio levando em conta as adequações definidas no passo anterior.

A.4.1.2-Atividade instalação propriamente dita

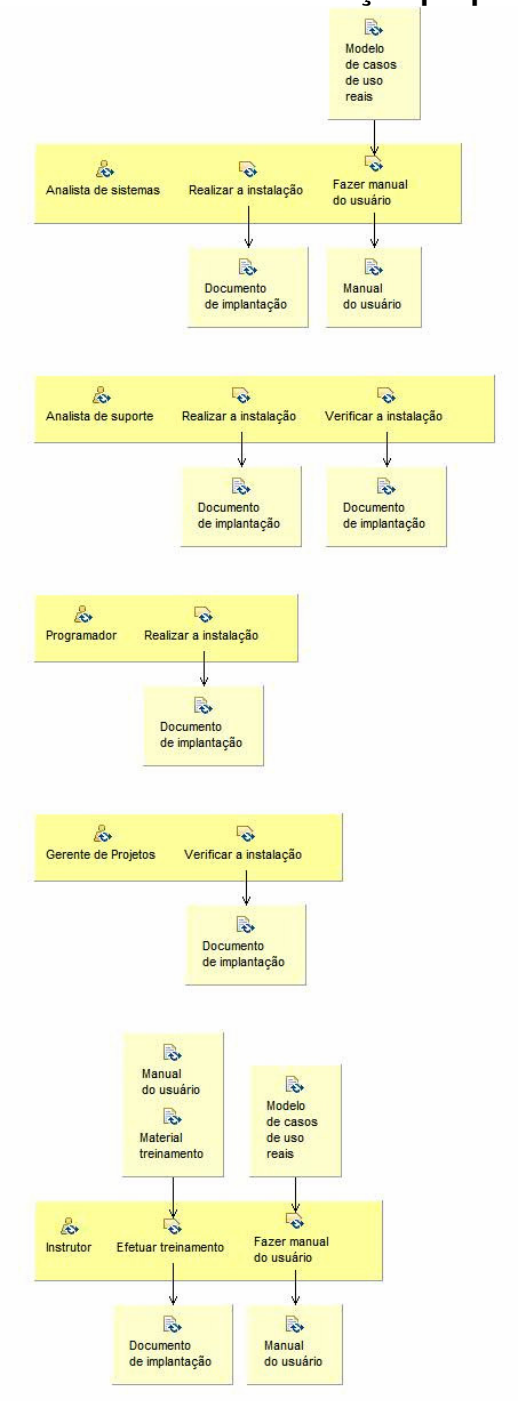


Figura 156- Atividade instalação propriamente dita

Realizar a instalação

⊟ Instalar hardware
Instalar e configurar todo o hardware necessá
⊟ Instalar software de apoio
Instalar todo o software de apoio necessário à operação do *data wareho*
⊟ Instalar a aplicação
Fazer a instalação da aplicação do *data warehouse:* configurar banco de dados, privilégios, senhas, restrições de acesso.

Fazer manual do usuário

⊟ Fazer manual do usuário
Escrever o manual do usuário a partir da descrição de casos de uso r

Verificar a instalação

⊟ Verificar desempenho
Verificar se a instalação atende ao desempenho estabelec
⊟ Verificar ajustes de segurança
Verificar se a instalação implementa todos os ajustes de segurança estabelecid
⊟ Verificar funcionamento
Verificar o funcionamento adequado da aplicaç

Efetuar treinamento

⊟ Determinar período e local do treinamento
Determinar quando e onde o treinamento vai ser efetuado, em função da disponibilidade dos usuários envolvidos.
⊟ Efetuar o treinamento
Efetuar o treinamento de usuários no período e local estabelecid

A.5-A Disciplina Gerência do Projeto

A disciplina gerência de projetos é realizada em todas as etapas do desenvolvimento e envolve atividades de verificação e gerenciamento do andamento do projeto e gerência de mudanças.

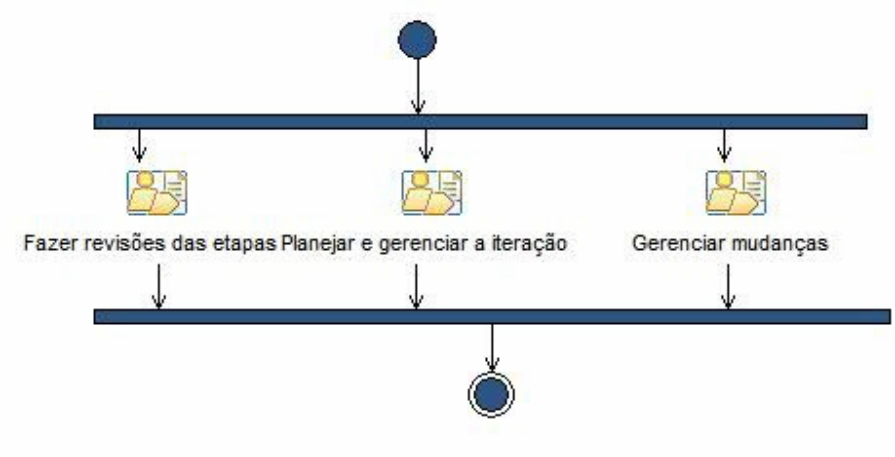


Figura 157-Diagrama de atividades da disciplina Gerência do Projeto

A.5.1-Atividade fazer revisão das etapas

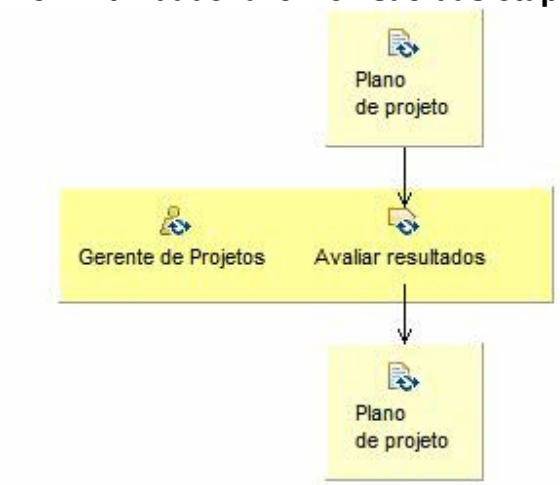


Figura 158- Atividade fazer revisão das etapas

Avaliar resultados

⊟ Preparar avaliação da iteração
Ao final de cada iteração, a equipe de desenvolvimento avalia se os objetivos e critérios de avaliação estabelecidos foram atingidos e se a equipe aderiu ao planc completou todos os itens de trabalho determinados para a iteração.
⊟ Demonstrar e obter retorno

Os produtos são demonstrados aos envolvidos, que dão o retorno correspondent
suas avaliações dos mesmos.

- Fazer uma retrospectiva

Rever com a equipe a efetivadade e adequabilidade de tudo que foi feito durante
desenvolvimento da iteração. Discute-se o que falhou e o que pode ser melhorad

- Fechar o projeto

Este passo deve ser realizado somente quando o final da iteração corresponder a
final da projeto, envolvendo avaliações finais e análises de aceitação.

A.5.2-Atividade Planejar e gerenciar a iteração

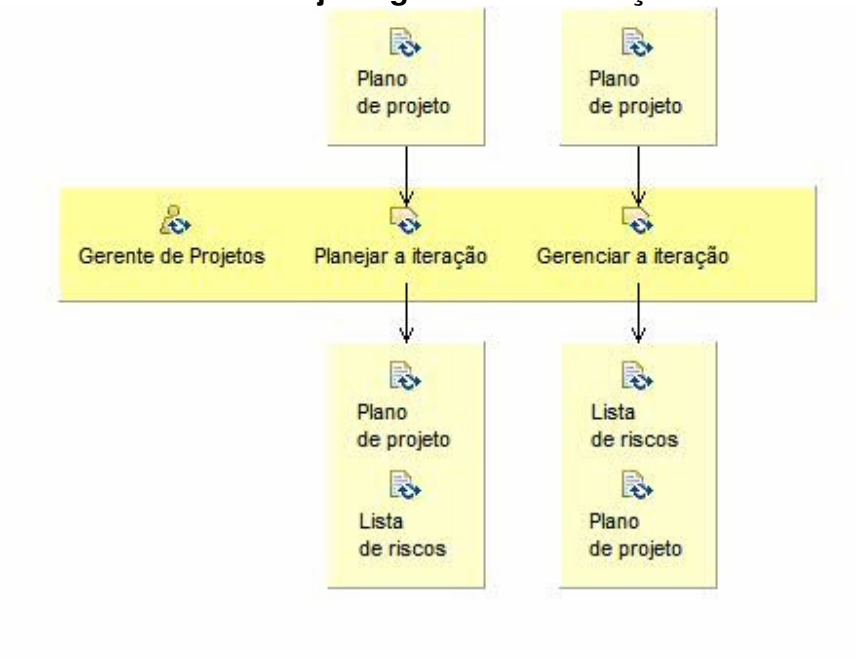


Figura 159- Atividade Planejar e gerenciar a iteração

Planejar a iteração

- Priorizar itens de trabalho

Estabelecer prioridades sobre os itens de trabalho antes da próxima iteração.
Considere o que foi modificado antes da última iteração.

- Definir objetivos da iteração

Trabalhar com a equipe de desenvolvimento de forma a refinar
objetivos estabelecidos na definição e escopo do projeto, em função de novas
prioridades da atual iteração.

- Acompanhar trabalho na iteração

Realizar acompanhamento do trabalho na iteração, tendo como referência as
prioridades definidas anteriormente e dando especial atenção aos trabalhos de
maior prioridade.

- Identificar e rever riscos

Durante o desenvolvimento, novas suposições e premissas vão surgir, as quais
consequentemente vão gerar novos riscos a serem identificados e avaliados.

⊟ Definir critérios de avaliação
Definir critérios de avaliação do andamento do projeto e da qualidade dos artefat
produzidos.
⊟ Refinar escopo e definições do projeto
De acordo com avaliações feitas e iterações anteriores, atualizar o escopo e
definições estabelecidos para o projeto (principalmente produtos de trabalho).

Gerenciar a iteração

⊟ Avaliar o progresso da iteração
Monitorar continuamente a iteração para garantir que esteja progredindo
adequadamente.
⊟ Capturar e comunicar a situação do projeto
Comunicar a todos os envolvidos, periodicamente, o andamento do projeto: meta
atingidas, possíveis atrasos, causas dos mesmos, próximas metas a serem
atingidas.
⊟ Gerenciar problemas
O gerente deve ter a capacidade de identificar problemas que estejam impedind
ou dificultando o andamento do projeto e atuar de forma a priorizá-los resolvê-los
⊟ Identificar e gerenciar riscos
Identificar riscos tão logo o projeto se inicie e durante seu andamento. Gerenciá-l
durante todo o desenrolar do projeto.
⊟ Gerenciar objetivos
Revisar o planejamento do projeto de forma a manter controle sobre o alcançar d
objetivos estabelecidos.

A.5.3-Atividade Gerenciar Mudanças

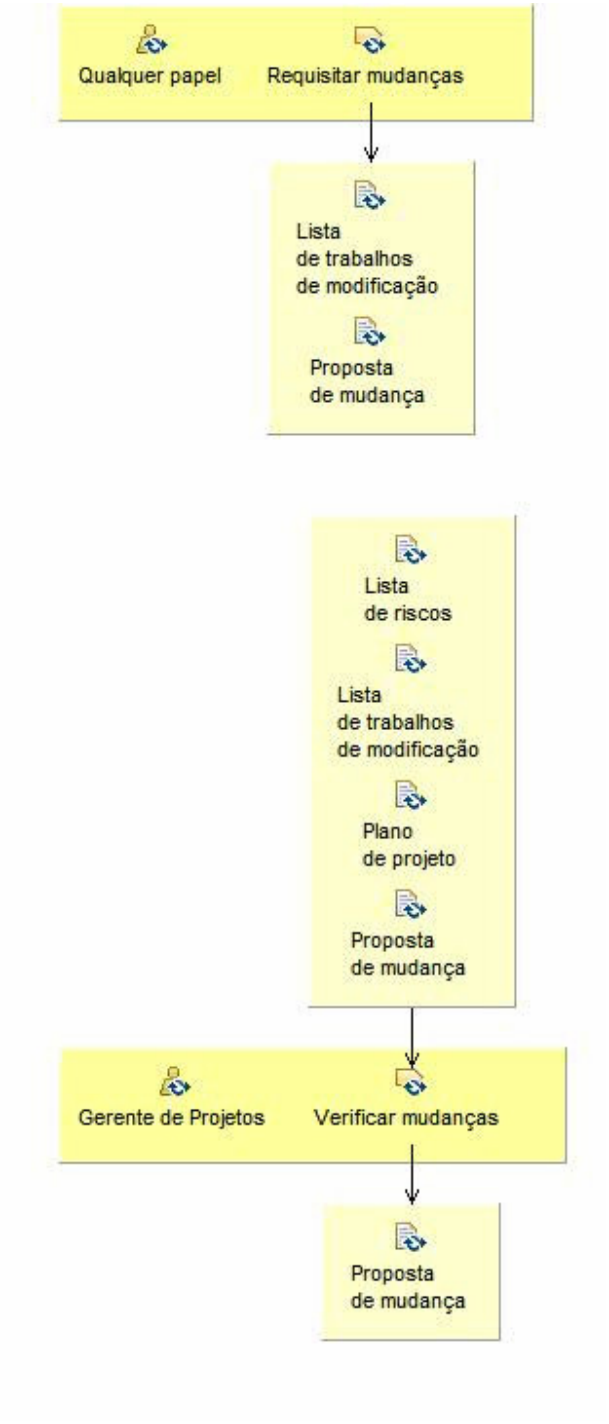


Figura 160-Atividade gerenciar mudanças

Requisitar Mudanças

⊟ Obter informação sobre requisicão de mudanças
Obter a informação necessária à descrição da requisição de mudanças. Isto deve incluir uma descrição da mudança, a razão da mudança (defeito ou melhoramento) artefatos afetados e a prioridade da mudança. Se possível, estimar o esforço para efetuá-la.
⊟ Atualizar lista de itens de trabalho
Atualizar a lista de itens de trabalho de forma a documentar as informações que foram obtidas no passo anterior.

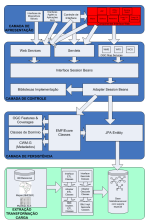
Verificar mudanças

⊟ Avaliar mudanças efetuadas
As mudanças requisitadas são verificadas de forma avaliar sua efetividade. Mudanças não aprovadas são retornadas aos responsáveis por sua implementaç para que sejam refeitas.
⊟ Fazer registro de mudanças
Todas as mudanças efetuadas devem ser registradas, com data, responsável, tempo para efetuá-la e status final.

APÊNDICE B- Diagramas de Classes e Detalhes de Implementação da Arquitetura de Software

B.1-A Camada de Apresentação

B.1.1-Módulo Componentes de Interface



Este módulo comporta classes básicas de interface para exibição de mapas relatórios e gráficos e objetos básicos (listas, *combo-boxes,* botões, caixas de checagem, etc) por meio de tecnologias JSF ou JavaScript/AJAX acessíveis respectivamente por arquivos JSP ou HTML. A Figura 161 exibe os pacotes envolvidos neste módulo:

- *ComponentesUIJavaScript/JSFInterfFixa:* a Figura 162 exibe o conjunto de classes definidas neste pacote, que implementam o padrão de interface mostrado na Figura 163, que é a proposta para visualização de cubos em aplicações do *data warehouse.* Ela é constituída por objetos de interfaces que podem ser construídos a partir de componentes JSF (*tags JSF)* ou a partir de componentes JavaScript com tecnologia AJAX. Tais objetos são painéis, quadros (*frames), combo-boxes, list-boxes,* mapas, gráficos e relatórios. A classe *Mapa*, especificamente, exibe mapas via serviço *WMS-Web Map Service* do OGC (classe *OGCWebMapService* descrita adiante). O *layout* desta interface contém as seguintes áreas:
  - A área *definição de filtros e dimensões* contém *list-boxes* para obtenção de filtros aplicáveis às dimensões.
  - A área *seleção de dimensões* contém um *combo-box* para seleção da dimensão a ser visualizada.
  - A área *seleção de fatos* contém um *combo-box* para seleção do fato a ser visualizado.

- A área *exibição de gráficos, relatórios e mapas* permite a visualização das informações na forma de gráficos torta ou barras , de relatórios sintéticos ou analíticos e em mapas (quando aplicável).

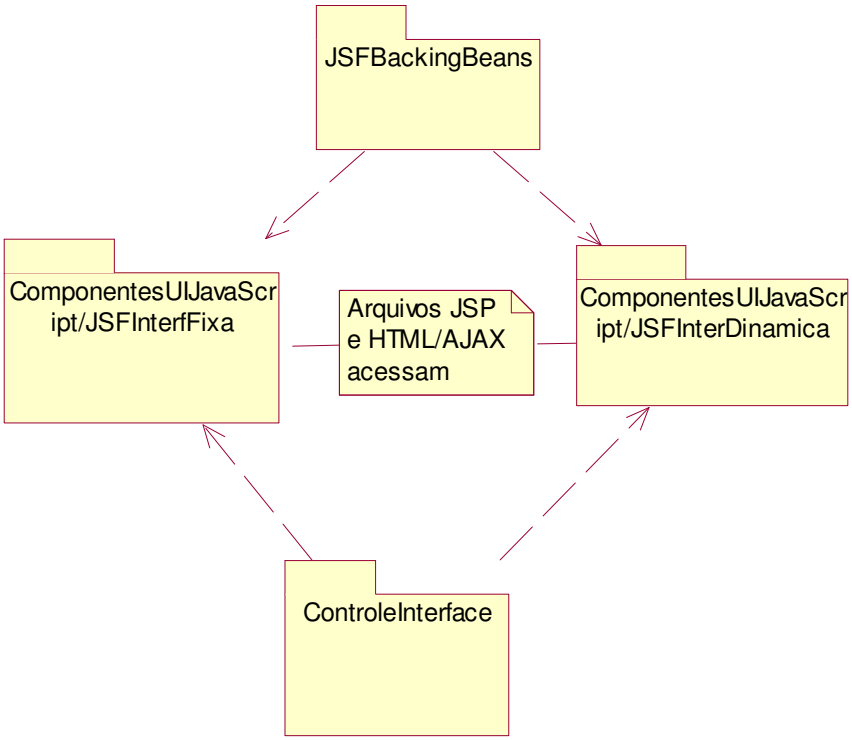


Figura 161-Pacotes dos módulos Backing Beans Interface e Controle Interface

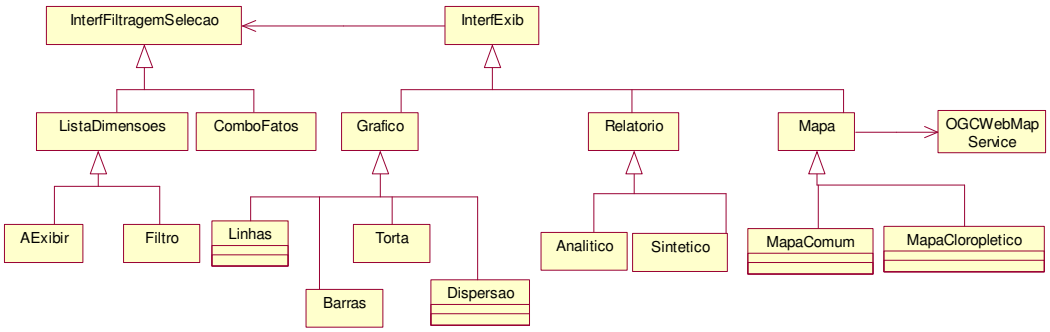


Figura 162-Componentes do pacote InterfaceFixa

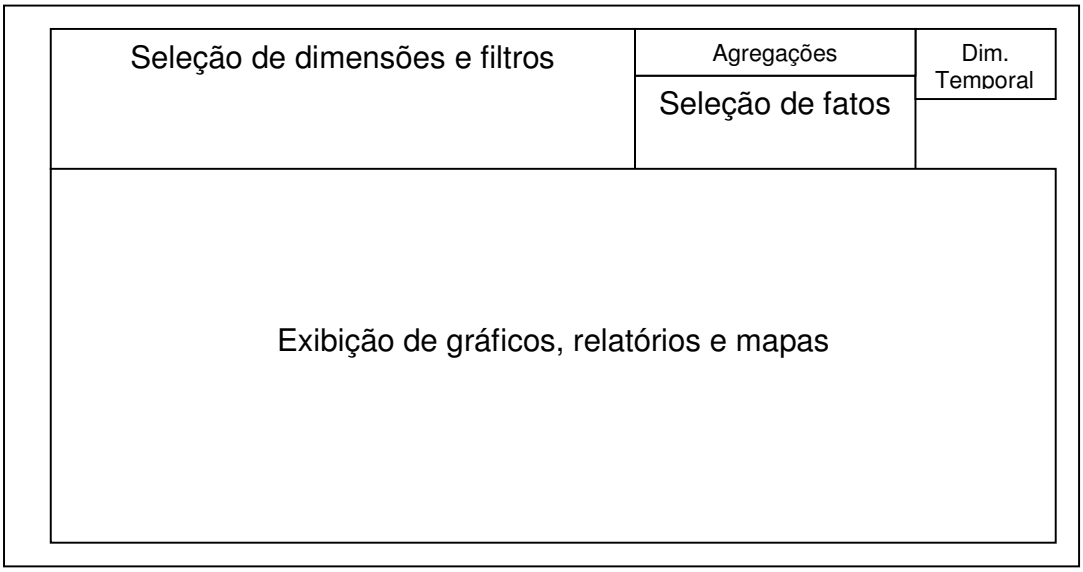


Figura 163-Padrão de interface

- *ComponentesUIJavaScript/JSFInterfDinamica:* esta interface, exibida na Figura 164, tem dois propósitos: permitir a seleção de dimensões e fatos de forma a gerar q*ueries* dinamicamente, exibidas em relatórios, gráficos e mapas, bem como permitir a geração rápida de uma interface fixa a partir dos fatos, dimensões e filtros selecionados, por meio de um código XML que pode ser traduzido em uma interface

conforme descrita no *layout* exibido na Figura 163. A Figura 165 exibe a estrutura de classes desse pacote.

- *JSFBackingBeans:* envolve um conjunto de *backing beans JSF* que definem propriedades e métodos que implementam funcionalidades de interfaces JSF.

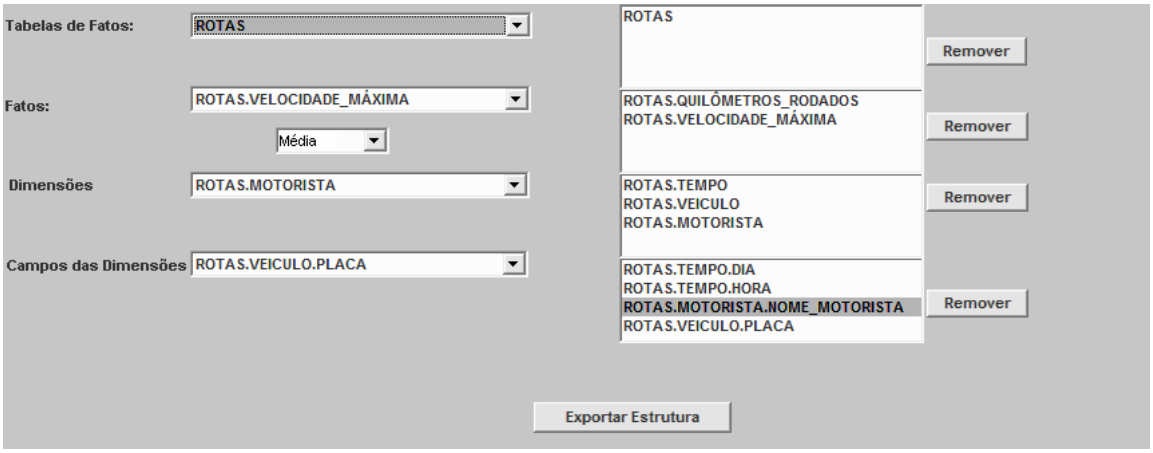


Figura 164- Layout da Interface Dinâmica

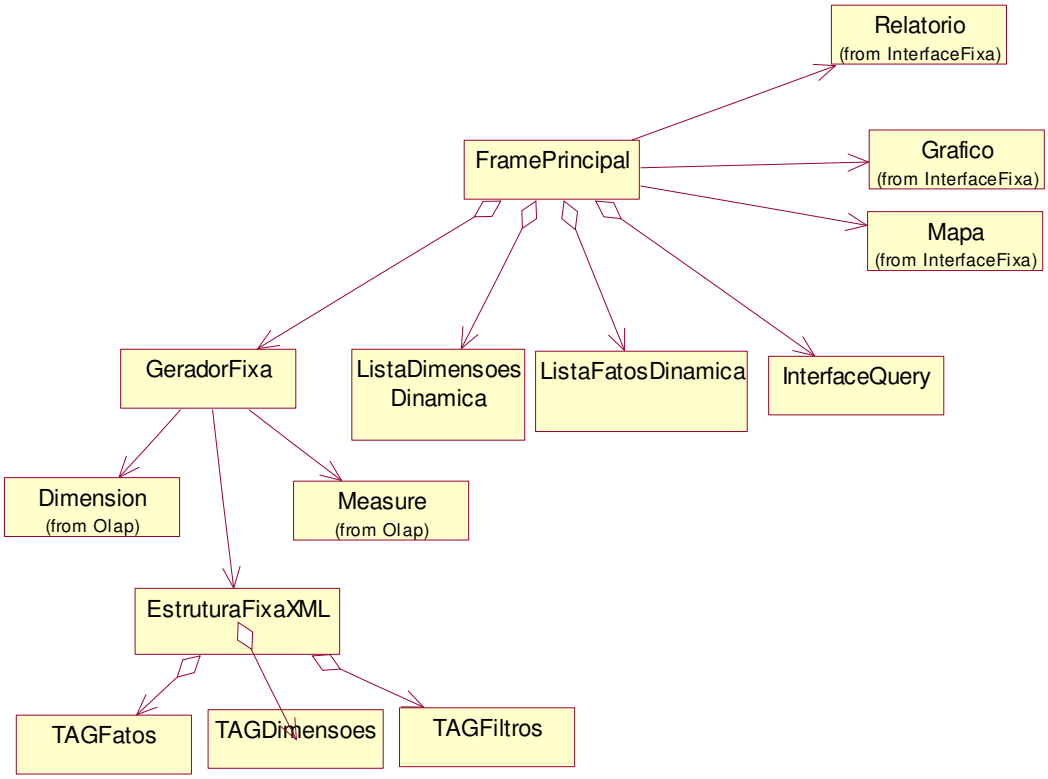
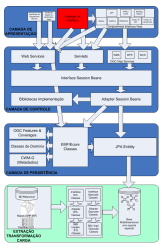


Figura 165-Diagrama de classes do pacote BackingBeansInterfaceDinamica

B.1.2-Módulo Controle de Interface



Este módulo possui classes que gerenciam a exibição das interfaces do módulo componentes de interface. O controle da exibição de interfaces é todo feito em nível de servidor, por meio de *servlets*, *Java Beans* e JSPs. A Figura 166 abaixo exibe a estrutura de classes utilizada.

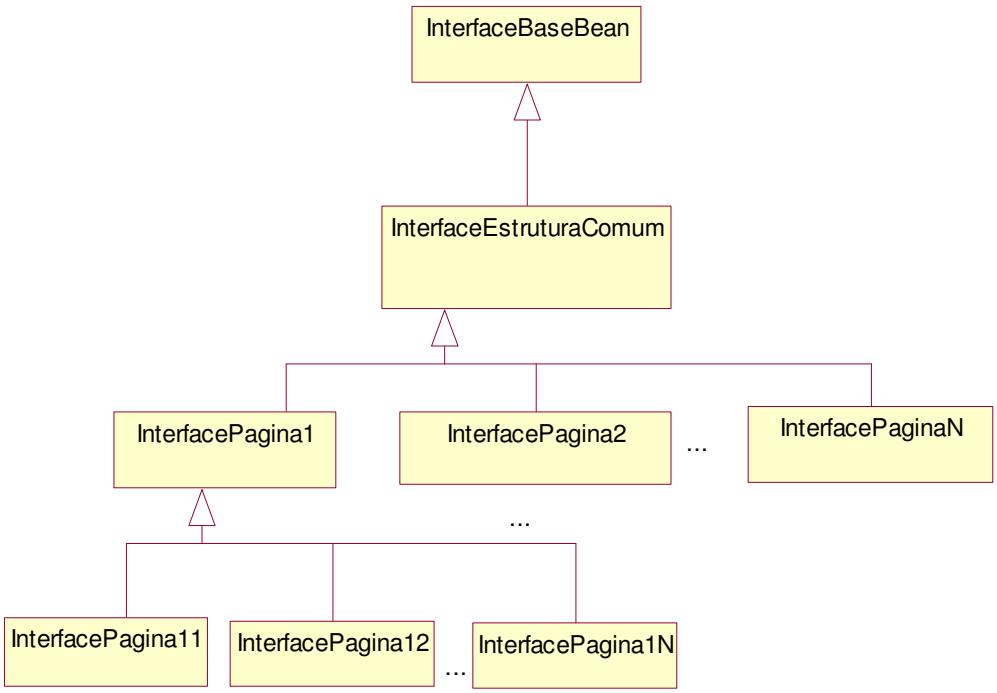


Figura 166-Estrutura de classes de Interface

A filosofia adotada consiste em permitir a criação de classes de interface que possam ser reutilizadas. Por exemplo, na Figura 166 a classe *InterfaceEstruturaComum* possui partes que são comuns a todas as interfaces da aplicação *web*, por exemplo, o cabeçalho, as laterais e o rodapé. As subclasses herdam estas características e acrescentam particularidades, como uma lateral à direita diferente, ou um setor central com determinado conteúdo. A interface padrão para consultas em cubos, exibida na Figura 163 foi desenvolvida utilizando-se este arcabouço.

A classe *InterfaceBaseBean* é a base para todas as classes de interface da aplicação. Em Java, é implementada como um *Java Bean* (item xx), cujos atributos denotam partes de um *template* padronizado, adequado a aplicações web, conforme mostrado na Figura 167:

<table>
<tr><td>Logo</td><td colspan="2">Header</td><td>Header2</td></tr>
<tr><td colspan="4">Band</td></tr>
<tr><td>LeftSide1</td><td colspan="2" rowspan="4">Center</td><td>RightSide1</td></tr>
<tr><td>LeftSide2</td><td>RightSide2</td></tr>
<tr><td>LeftSide3</td><td>RightSide3</td></tr>
<tr><td>LeftSide4</td><td>RightSide4</td></tr>
<tr><td colspan="4">Footer</td></tr>
</table>

Figura 167-Template da estrutura sugerida para uma interface web

O formato padrão de página da Figura 167 pode ser alterado, inclusive com modificação de formato, tamanho e posicionamento dos setores. Este formato não é obrigatório, mas os nomes dos setores do *template* (*logo, header, footer, etc)* devem ser preservados e são também nomes de atributos da classe *InterfaceBase,* como mostrado abaixo:

```
public class InterfaceBaseBean {

  private String logo;
  private String header;
  private String header2;
  private String leftside1;
  private String rightside1;
  private String leftside2;
  private String rightside2;
  private String leftside3;
  private String rightside3;
  private String leftside4;
  private String rightside4;
  private String center;
```

```
private String footer;
private String band;
```

Cada um destes atributos possui métodos de acesso *setters,* alguns deles mostrados abaixo:

```
public void setlogo(String logo) {
    this.logo = logo;
}

  public void setheader(String header) {
  this.header = header;
}

public void setheader2(String header2) {
  this.header2 = header2;
}
```

Estes métodos, ao serem invocados, exibem no espaço correspondente visto na Figura, um conteúdo diversificado, sobre o qual pode-se observar:

- Tal conteúdo pode envolver arquivos HTML ou JSP, passando-se como parâmetro o nome do arquivo correspondente, como em *setheader2("logo2.html")*, que exibe o conteúdo de logo2.html no setor header2. Esta forma é *default* e não demanda o ajuste dos atributos que serão descritos a seguir (*dlogo, dcenter, etc*). Os arquivos JSP e HTML possuem recursos avançados, tais como:
    - Arquivos JSP, que podem ser criados com o uso de recursos JSF: APIs de componentes de Interface e *Backing Beans* para implementação de funções relacionadas a tais componentes.
    - Arquivos HTML utilizam *scripts Java Script,* para implementação de recursos de interface avançados, como por exemplo, a tecnologia Ajax e acesso a APIs sofisticadas, como a do *Google Maps* ou de servidores de mapas como o *Geoserver.*

- O conteúdo pode ser também um texto livre, formatado ou não com *tags* HTML. Neste caso, o texto livre é passado como parâmetro ao método *setter.* Entretanto, neste caso, devemos modificar o valor de atributos da classe *InterfaceBaseBean*, que designam se vai ser inserido um texto livre ou o conteúdo de um arquivo. Estes atributos são os seguintes:

```
private String dlogo;
private String dheader;
private String dheader2;
private String dleftside1;
private String drightside1;
private String dleftside2;
private String drightside2;
private String dleftside3;
private String drightside3;
private String dleftside4;
private String drightside4;
private String dcenter;
private String dfooter;
private String dband;
```

Repare que o nome das partes mostradas na Figura 41 aparece precedido da letra *d* de *direct (direto).* Estes atributos são do tipo string e assumem o valor “true” or “false”. Por default, são “false”. Quando “true” designam que o parâmetro passado para os métodos *setLogo, setHeader, setRoot,* etc, deve ser um *texto livre,* formatado ou não com tags HTML. Quando “false” (o default), indicam que tal parâmetro é o nome de um arquivo HTML ou JSP. Como estes atributos são privados (private), devem ser ajustados pelos métodos *setters,* alguns deles mostrados abaixo:

```
public void setdlogo(String dlogo) {
    this.dlogo = dlogo;
```

```
 }
public void setdheader(String dheader) {
   this.dheader = dheader;
 }
public void setdheader2(String dheader2) {
   this.dheader2 = dheader2;
 }
```

Assim, para colocar o texto “Hello world” no centro de uma página, faríamos assim:

```
setCenter(“Hello world”);
setdCenter(“true”);
```

Entretanto, o *template* define uma estrutura “genérica” , cujo formato pode ser modificado em relação ao mostrado na Figura 41 (mantendo-se, entretanto os mesmos setores básicos). O *layout* mostrado na Figura 41, por exemplo, é um versão modificada deste *template,* em que as partes do mesmo encontram-se em uma disposição específica.
A classe *InterfaceBase* possui ainda o método

```
chamapagina(HttpServletRequest req,HttpServletResponse res, ServletContext context)
```

que é invocado para exibição da página. Ele recebe como parâmetros os objetos do tipo HttpServletRequest e HttpServletResponse já mencionados e ainda ServletContext que se refere ao contexto do servlet, obtido na inicialização do mesmo.

O *template* da Figura 41 é gerenciado por um arquivo JSP que contém *tags* personalizadas formalizadas em um arquivo XML de bibliotecas (com a extensão .TLD).

A classe *InterfaceBaseBean* é abstrata, ou seja, não são criadas instâncias da mesma. Ela contém os métodos de acesso descritos no item anterior, que permitem a definição de conteúdo alocado aos setores mostrados na Figura 167.

Portanto, toda a estrutura da interface é montada sobre classes derivadas de InterfaceBase (ver Figura 166). Para criar uma interface de exibição de uma página, seguimos os seguintes passos:

1-Cria-se uma classe derivada de InterfaceBaseBean:

```
public class InterfaceBaseTudo extends InterfaceBaseBean
```

A classe InterfaceBaseTudo do exemplo contém a estrutura comum a todas as classes de interface da aplicação.

2-No construtor da classe derivada, invocam-se os métodos *set<setor>* e/ou *setd<setor>* que estão definidos como públicos na classe *InterfaceBaseBean*, para alocar conteúdo específico de um arquivo HTML/JSP ou texto livre. Abaixo, no construtor de *InterfaceBaseTudo* invocamos os métodos *setBand, setLogo, serheader, setdheader e setfooter* . Todos inserem arquivos HTML nos setores específicos, exceto o *setheader* que insere um texto formatado com tags HTML, seguido pela chamada de *setdheader* tendo "true" (indicando inserção direta de texto) como parâmetro. Repare que deve-se colocar no construtor da classe derivada, a chamada ao construtor da classe base (super.InterfaceBaseBean()) para que sejam realizadas todas as inicializações.

```
 public InterfaceBaseTudo () {
        super.InterfaceBaseBean();

        setlogo("/logopuc.htm");
        setheader2("/logosg.htm");
        setband("/faixa.htm");
        setheader("<strong><font size='6'>REVISTA DIGITAL DA UNIDADE PUC SÃO
GABRIEL</font></strong>");
        setdheader("true");
        setfooter("/creditos.htm");
 }
```

3-No local onde a exibição da página deve ser efetuada, cria-se uma instância da classe derivada e invocamos o método *chamapagina* pertencente à classe InterfaceBaseBean. Isto será sempre feito dentro dos métodos *get, post, put ou service* do servlet, daí a necessidade de se passarem os parâmetros *req, res* e *context* ao método *chamapagina:*

```
//Chamada efetuada dentro dos métodos get, post, put ou service do //servlet

InterfaceBaseTudo ibt= new InterfaceBaseTudo();
ibt.chamapagina(req,res,context);
```

4-Podem ser criadas classes derivadas que herdam o que foi inserido nos setores pelas classes mães e inserem particularidades. No exemplo, a classe InterfacePrimPag herda tudo o que foi inserido nos setores *logo, header, header2, footer e band* e insere um texto no centro (*center)* e arquivos nos setores *leftside1, leftside2* e *rightside1.*

```
public class InterfacePrimPag extends InterfaceBaseTudo {

  public  void InterfacePrimPag() {
  }

  public InterfacePrimPag (String texto,String dcenter) {
        super.InterfaceBaseTudo();
        setcenter(texto);
        setdcenter(dcenter);
        setleftside1("/areas.htm");
        setleftside2("/link.htm");
        setrightside1("/cartas.htm");
  }
}
```

É fácil observar que o construtor recebe dois parâmetros da classe *String*. Estes parâmetros correspondem ao conteúdo a ser colocado na parte central (center) da página, podendo ser um arquivo (dcenter = “false”) ou texto livre (dcenter=”true”). Desta forma, podemos iniciar a classe *IntefacePrimPag* de diversas formas, variando os parâmetros de seu construtor. Abaixo, inicia-se esta classe com o texto “SEJA BEM-VINDO!!!”.

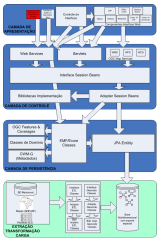
```
InterfacePrimPag ipp= new InterfacePrimPag("SEJA BEM-VINDO!!!","true");
ipp.chamapagina(req,res,context);
```

Os parâmetros de *chamapagina (req, res, context)* são obtidos respectivamente a partir dos métodos *get, post, put ou service* do servlet e do método *init* do servlet:

```
import javax.servlet.*;
import javax.servlet.http.*;

public class primpag extends HttpServlet {

//Declarar o contexto
private ServletContext context;

public void init(ServletConfig config) throws ServletException {
   super.init(config);

  //Obter o contexto do servlet

   context = config.getServletContext();
}

public void doGet (HttpServletRequest req, HttpServletResponse res)
throws ServletException, IOException{

//Criar a instância da interface e invocá-la

InterfacePrimPag ipp= new InterfacePrimPag("SEJA BEM-VINDO!!!","true");
```

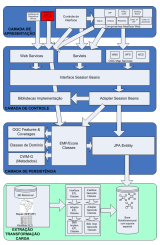
```
ipp.chamapagina(req,res,context);

}
}
```

B.1.3-Módulos Interface de Dispositivos Móveis



Este módulo tem como objetivo a implementação de classes que permitam a exibição de informações em dispositivos móveis como telefones celulares, PDAs, *Pocket PCs, Palm Tops,* dentre outros. Não serão detalhadas classes deste módulo pois existe um número muito grande de modelos e de linguagens de implementação para os mesmos, os quais têm em comum o fato de acessarem os serviços disponibilizados no arcabouço via *web services.*

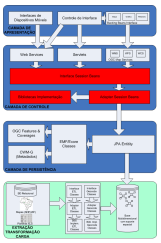
B.1.4-Módulos Interfaces Applet e Aplicações Java



Este módulo contém classes de interface comuns que implementam *applets* e aplicações Java convencionais, por meio, por exemplo, de componentess AWT ou Swing. O acesso a dados e ou a funcionalidades proporcionadas pelas classes ISB é feito, principalmente, por meio de *web services.*

B.2-A Camada de Controle

B.2.1-Módulos relacionados a *Session Beans*



Os módulos *Interface Session Beans, Adapter Session Beans* e *Bibliotecas de Implementação* constituem o núcleo funcional do arcabouço. Eles descrevem e implementam todas as interfaces de acesso a regras de negócio, algoritmos e bibliotecas, além de prover acesso a classes da camada de persistência. O núcleo arquitetural descrito no item 3.6.1 é, principalmente, aqui instanciado. As classes do tipo *Interface Session Bean* -ISB são implementadas por classes *Adapter Session Bean*-ASB, via acesso a *bibliotecas Implementação,* normalmente fornecidas por terceiros, que contêm pacotes e classes prontas, com a implementação da funcionalidade. Ressalta-se novamente que essa estrutura provê flexibilidade na alteração de implementações, de forma que as classes e pacotes contidas nas Bibliotecas de Implementação possam ser substituídas, sem impacto na aplicação que as acessa, pois estas o fazem via ISBs, que não se alteram. Desta forma, quando houver necessidade de alterações, implementam-se novas classes ASB específicas sem, entretanto, qualquer alteração na assinatura de seus métodos. O arcabouço aqui apresentado é facilmente extensível, com a possibilidade de inclusão de outras técnicas de análise de dados além das exibidas , a título de exemplo.

Pacotes para Análise de Dados

Na versão atual, estes pacotes contêm definições de interfaces relacionadas a algumas técnicas de análise de dados. As técnicas exibidas foram escolhidas em função de sua popularidade ou utilização em projeto específico.. Encontram-se distribuídas nos pacotes exibidos na Figura 168.

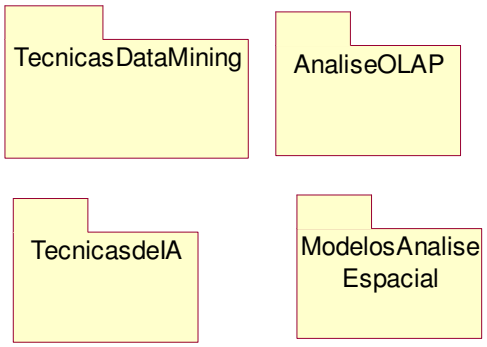


Figura 168-Pacotes de Análise de Dados

Pacote TecnicasDataMining

Especifica interfaces para implementações de algoritmos de *data mining* convencionais e geográficos. A Figura 169 exibe alguns pacotes de técnicas de *data mining* que contêm algumas técnicas especificadas a título de exemplo.

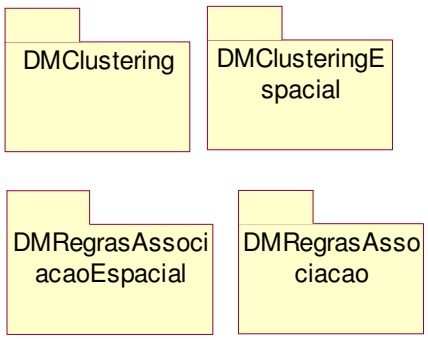


Figura 169- Pacotes de técnicas de data mining

O diagrama de classes e interfaces pertencentes ao pacote *DMClustering* está exibido na Figura 170. A interface *ISBClustering* provê métodos comuns a todas as outras, que a especializam com implementações específicas de cada algoritmo.

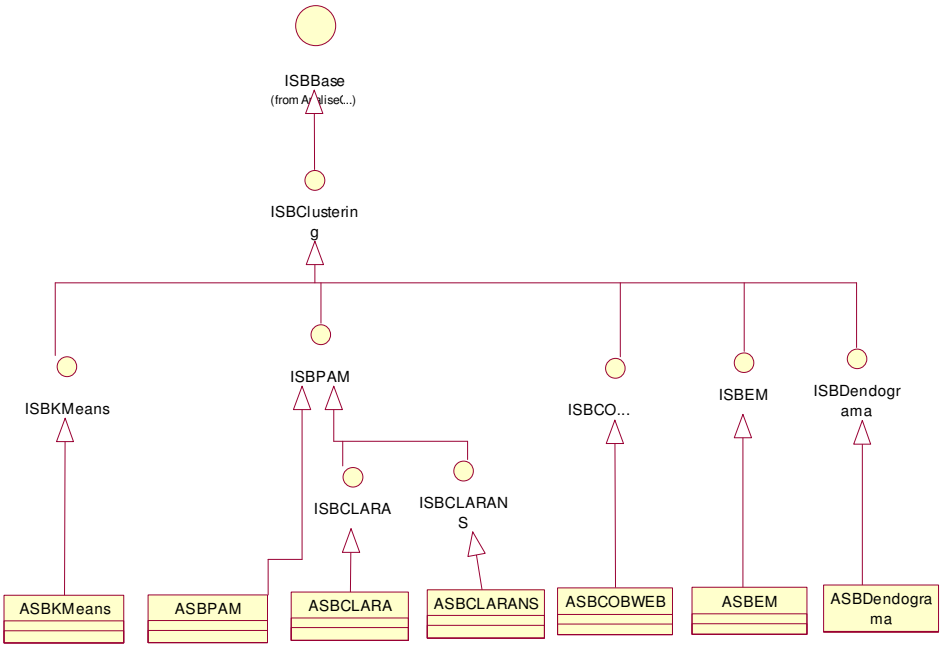


Figura 170 – Interfaces que especificam alguns algoritmos de clustering convencional

A interface *ISBBase* provê acesso comuns a *classes* que implementam aspectos do metamodelo CWM, que constituem a base do arcabouço proposto. A Figura 171 exibe as relações entre ISBBase e classes do CWM necessárias à realização das funcionalidades de *data mining* e *clustering* em especial.

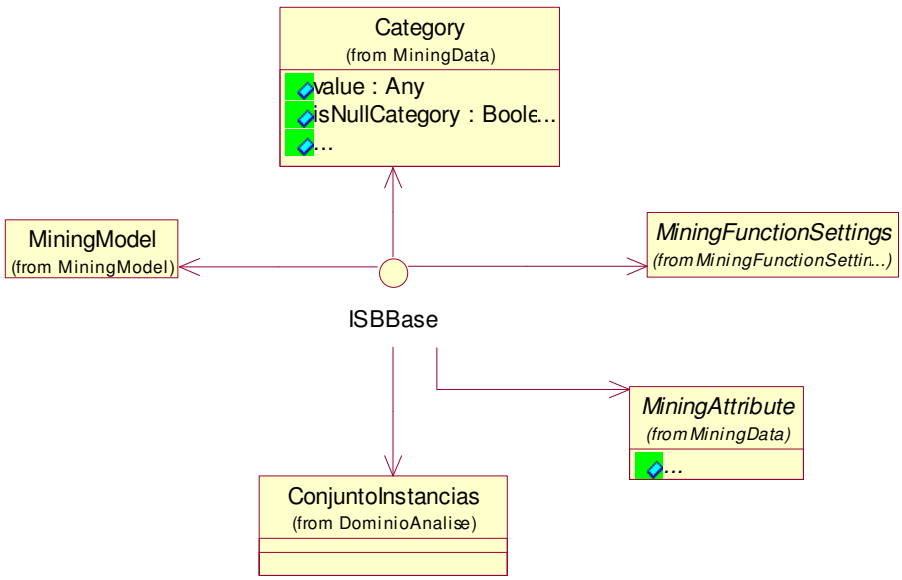


Figura 171-Relação com classes do CWM

A Figura 172 exibe as interfaces do pacote DMClusteringEspacial

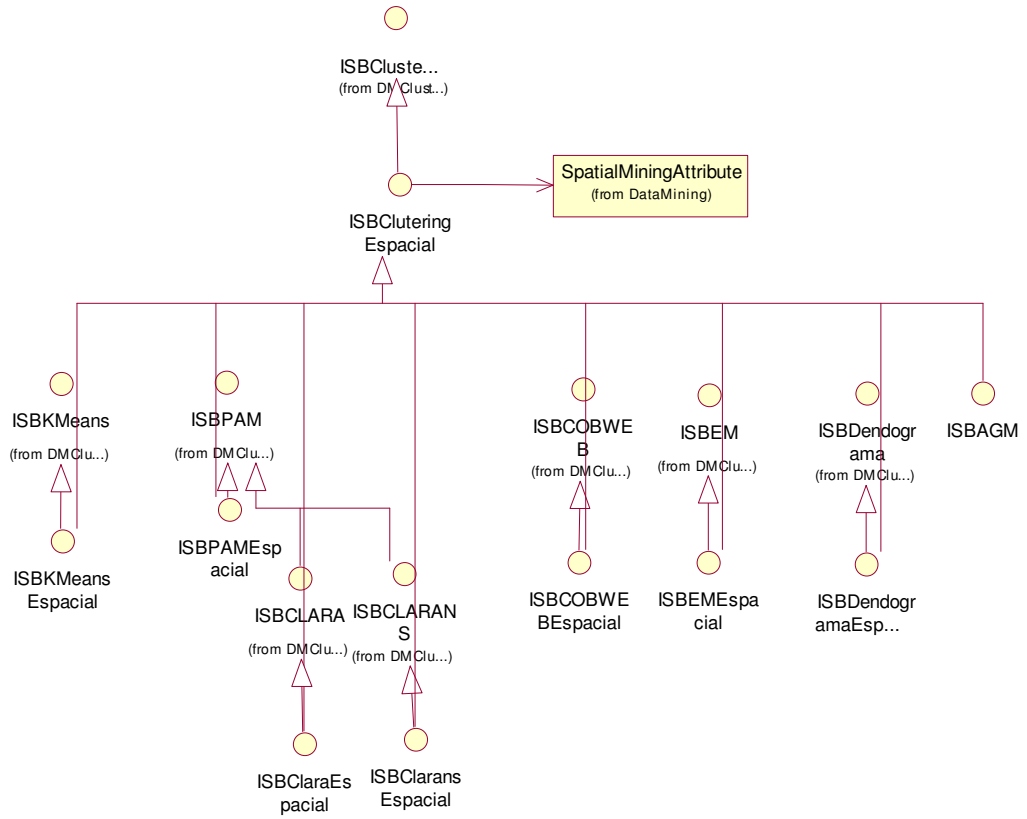


Figura 172- Interfaces que especificam alguns algoritmos de clustering espacial

As Figuras 173 e 174 a seguir exibem os diagramas de interfaces e classes dos pacotes que especificam respectivamente interfaces para implementação de regras de associação convencional e espacial.

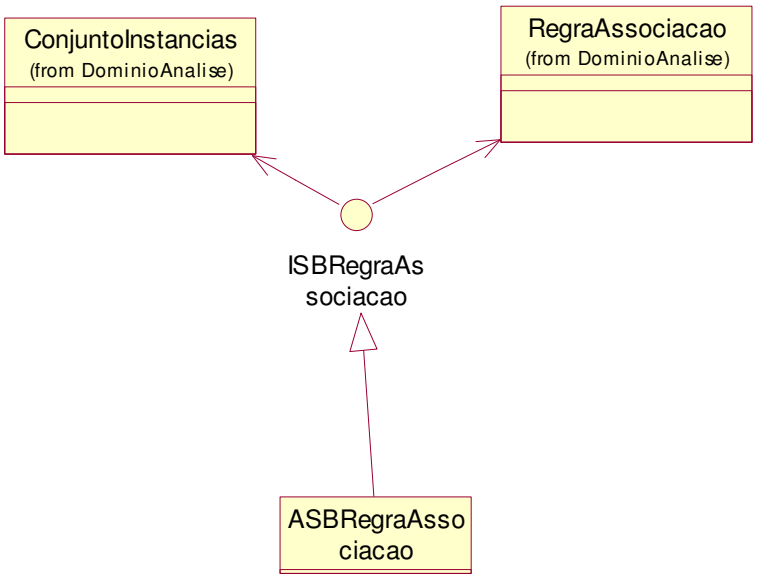


Figura 173 – Diagrama de classes e interfaces para ISB regras associação

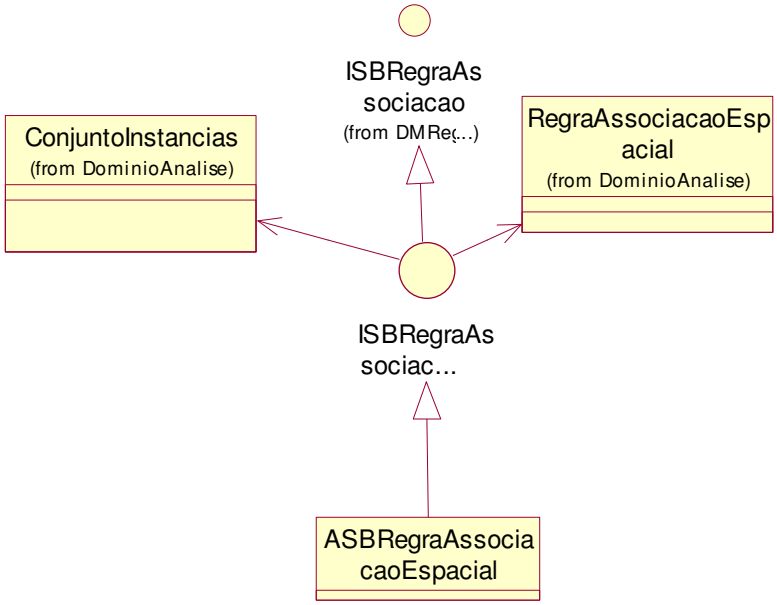


Figura 174 - *Diagrama de classes e interfaces para ISB regras associação espacial*

Pacote TecnicasdeIA

Este pacote especifica interfaces para implementações algoritmos de Inteligência Artificial. Neste trabalho, foram especificadas interfaces para as técnicas de redes neurais e algoritmos genéticos, localizadas nos pacotes exibidos na Figura 175.

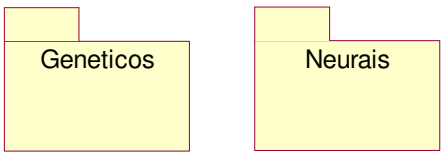


Figura 175-Pacotes de técnicas de IA

A Figura 176 exibe as classes ISBs do pacote Neurais que implementam treinamento e recuperação de informação armazenadas em redes neurais MLP (*Multi Layer Perceptron)* e SOM (*Sef Organizing Maps).* A Figura 177 exibe o ISB especificado para algoritmos genéticos.

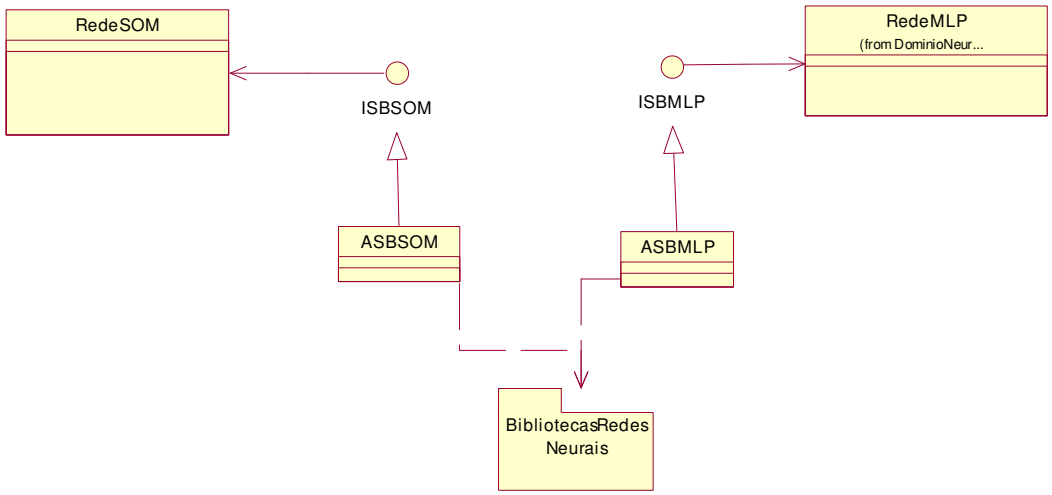


Figura 176-Interfaces ISBs que especificam redes neurais MLP e SOM

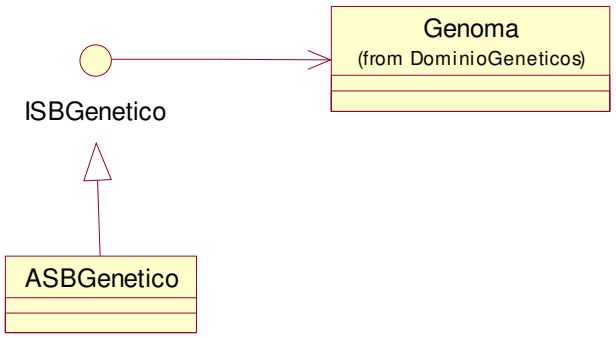


Figura 177 -Interfaces ISBs que especificam redes neurais MLP e SOM

Pacote AnaliseOLAP

Este pacote implementa acesso a cubos convencionais e espaciais. O diagrama de classes correspondente está exibido na Figura 178. O aplicativo para análise multidimensional, descrito no item 4.5, acessa uma implementação desta interface.

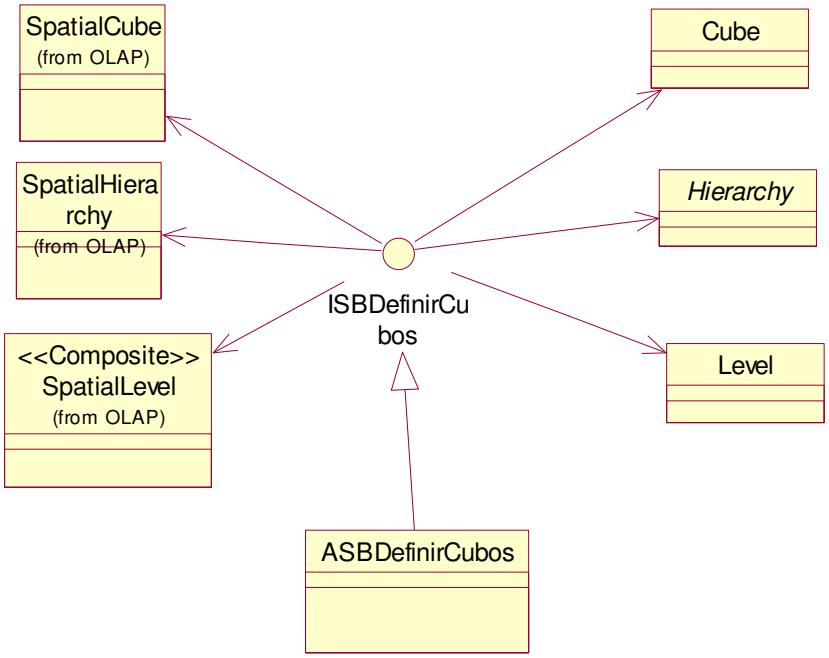


Figura 178 – Diagrama de classes e interfaces do pacote AnaliseOLAP

O Pacote Modelos de Análise Espacial

Este pacote contém interfaces que especificam acesso a alguns algoritmos de análise espacial. A Figura 179 exibe os ISBs que implementam os métodos autocorrelação (LISA), Hägerstrand e Geometria Táxi. O Capítulo 4 exibe exemplos de aplicações desenvolvidas que utilizam implementações dessas interfaces.

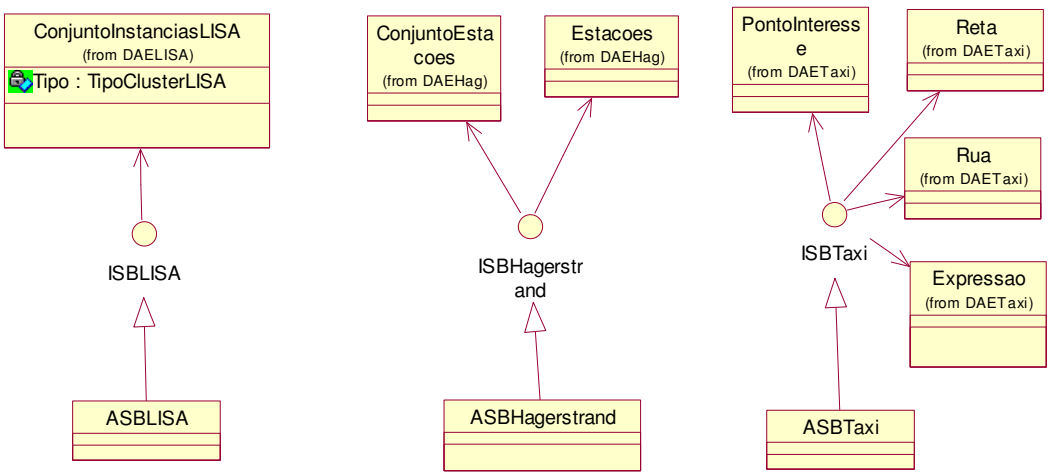
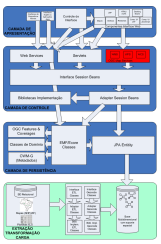


Figura 179-ISBs que implementam alguns métodos de análise espacial

B.2.2-Módulos OGC *Map Services*



Pacote de Interfaces para Serviços OGC

Este pacote possui interfaces ISB para implementação de alguns serviços OGC, conforme a descrição apresentada no item 3.1.2.3. As interfaces especificadas são as seguintes:

- *ISBWMS:* especifica operações do serviço OGC *Web Map Services, (WMS);*
- *ISBWFS:* especifica operações do serviço OGC *Web Feature Services, (WFS);*

- *ISBWCS:* especifica operações do serviço OGC *Web Coverage Services, (WMS);*

Estas interfaces são acessadas via *web services, servlets* ou arquivos JSP, que provêem acesso às mesmas.

As interfaces ISBWMS e ICBWCS foram implementadas por meio das classes ASBWMS e ASBWCS que utilizam a respectivamente as APIs do *Google Maps* para acesso a mapas e do pacote *Geotools* , que possui implementação de todos os serviços OGC, inclusive do WCS. A aplicação-piloto (item 4.6) e a aplicação para consultas em bases de dados multidimensionai (item 4.5) utilizam *servlets* e JSPs para acesso às classes ASBWCS e ASBWMS implementadas.

As interfaces deste módulo foram apresentadas separadamente para que seja destacada sua importância dentro da arquitetura proposta. A Figura 180 abaixo exibe o diagrama de classes com as interfaces que especificam os serviços OGC.

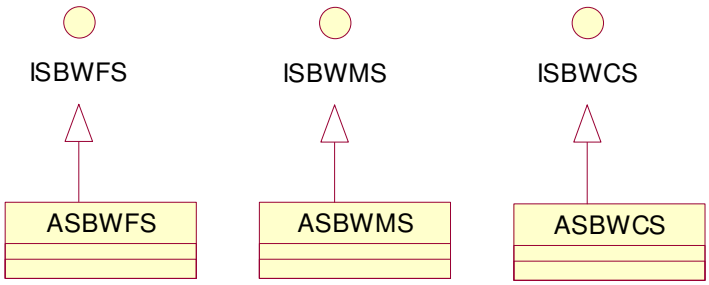
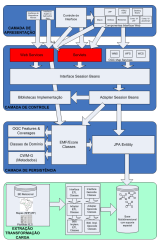


Figura 180 – Diagrama de classes para Serviços OGC

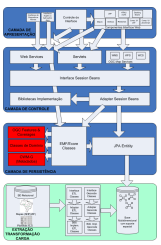
B.2.3-*Servlets e Web Services*



Este módulo provê todas as interfaces para conexão entre as camadas de controle e apresentação da arquitetura, principalmente para as classes de análise. *Servlets e Web Services* provêem o acesso da camada de apresentação às ISBs definidas na camada de controle.

B.3-A camada de Persistência

B.3.1-Classes de Domínio da Aplicação



Este módulo especifica todas as classes de domínio de problemas que, em última instância, constituem o metamodelo da aplicação. Três categorias de classes foram definidas na arquitetura proposta:

Features e Coverages OGC

Especificam os modelos *Feature e Coverage* do OGC, apresentados no item 3.1.2.2. As Figuras 181 e 182 exibem os diagramas de classes que contém as principais classes destes modelos.

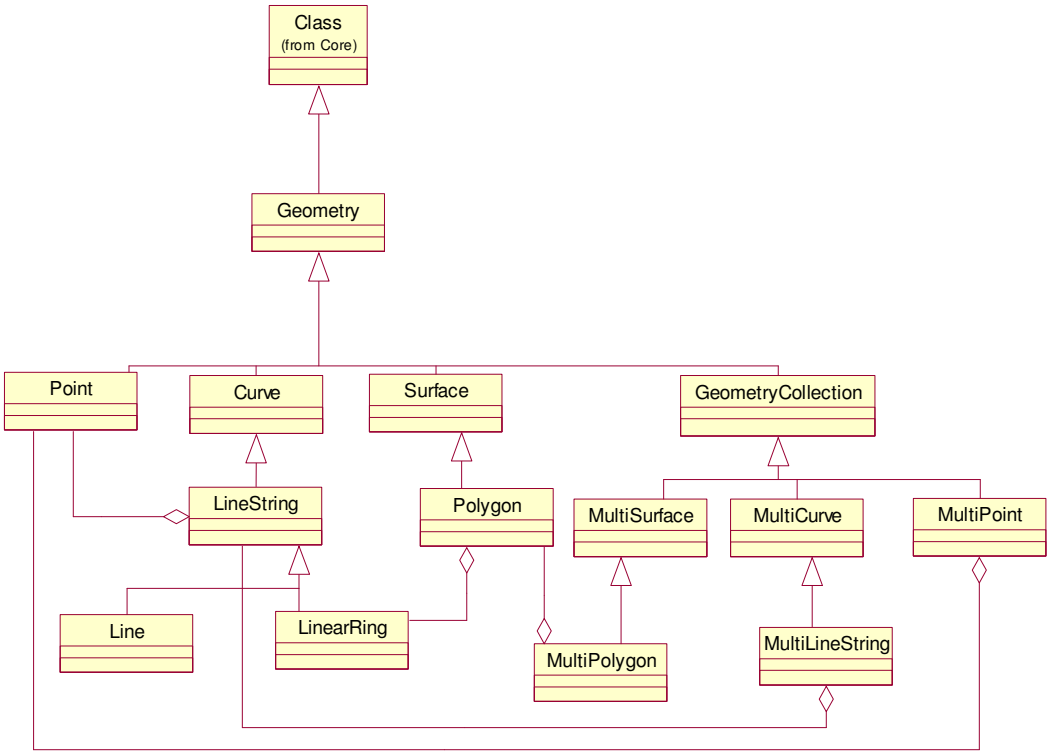


Figura 181 – Classes OGC Feature

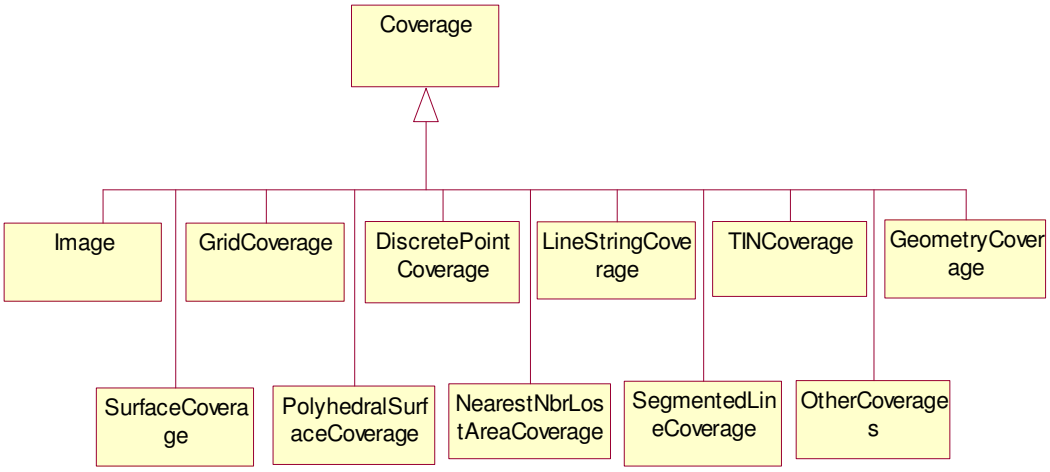


Figura 182-Classes OGC Coverage

Classes de domínio do problema propriamente ditas

Especificam classes de domínio de problema para análise de dados, transformação de dados e extração de dados.

A Figura 183 exibe as classes utilizadas em funções de análise de dados (ISBs relacionados a análise de dados: *data mining* convencional e espacial e análise espacial).

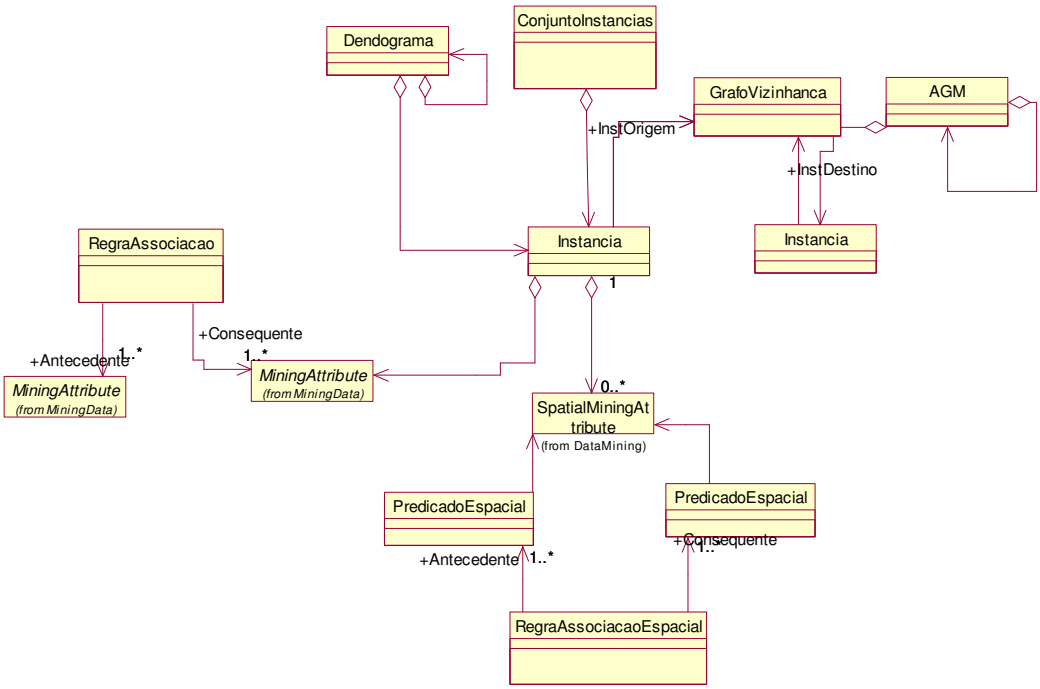


Figura 183 – Classes de domínio de problema de análise de dados

A Figura 184 exibe as classes de domínio utilizados no ISBELISA, que especifica interfaces para métodos de autocorrelação

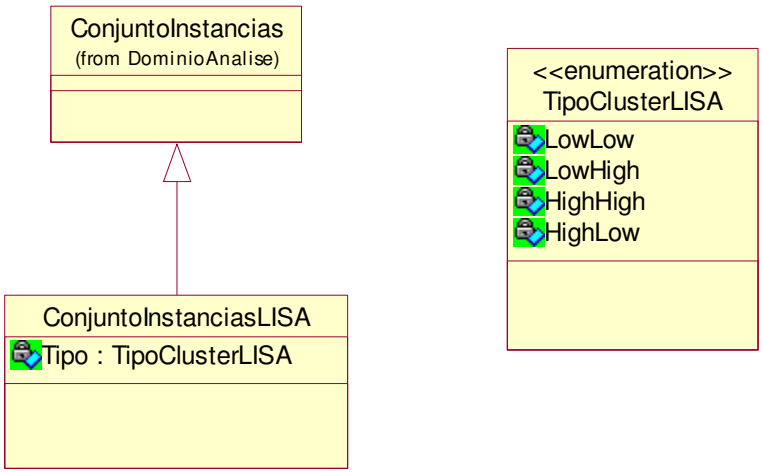


Figura 184 – Classes de domínio para métodos de autocorrelação

A Figura 185 exibe as classes de domínio de problema da interface ISBHagerstrand que especifica interfaces para o modelo têmporo-espacial de Hägerstrand.

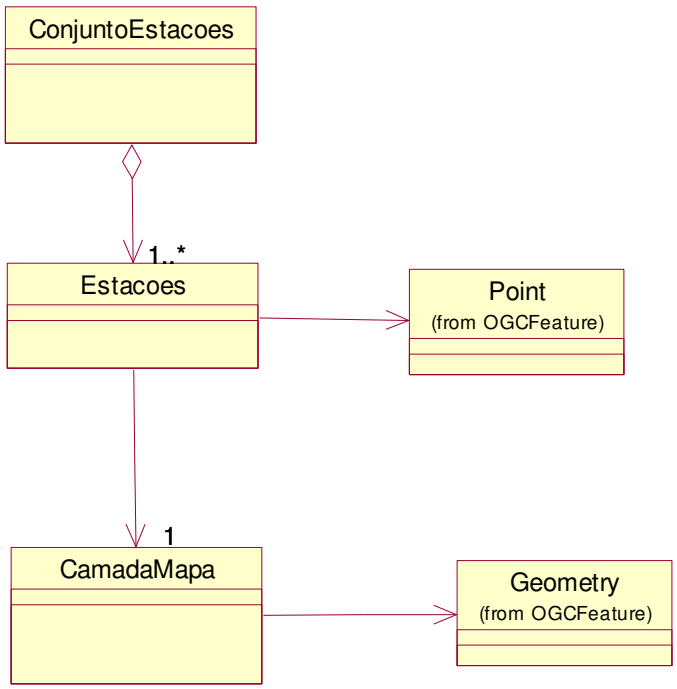


Figura 185- Classes de domínio do modelo Hägerstrand

A Figura 186 exibe as classes de domínio de problema da interface ISBTaxi que especifica interfaces para o modelo de análise espacial que utiliza a Geometria Táxi.

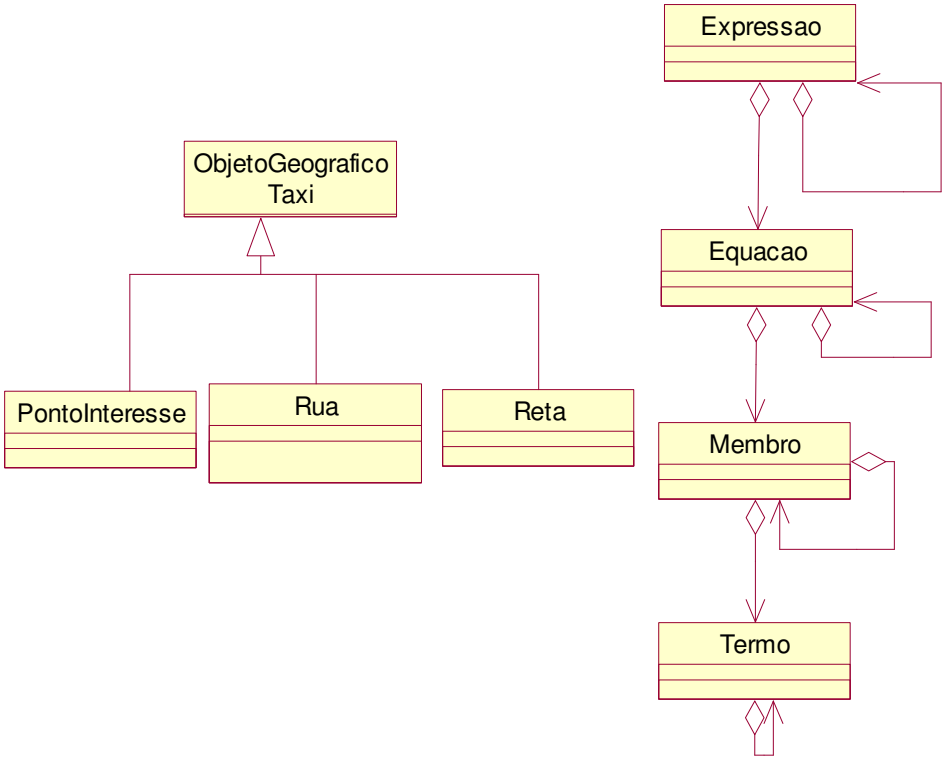


Figura 186 – Classes do domínio do método de análise espacial que utiliza a Geometria Táxi

A Figura 187 exibe as classes de domínio de problema das interfaces ISBMLP e ISBSOM que especificam interfaces para implementação de redes neurais.

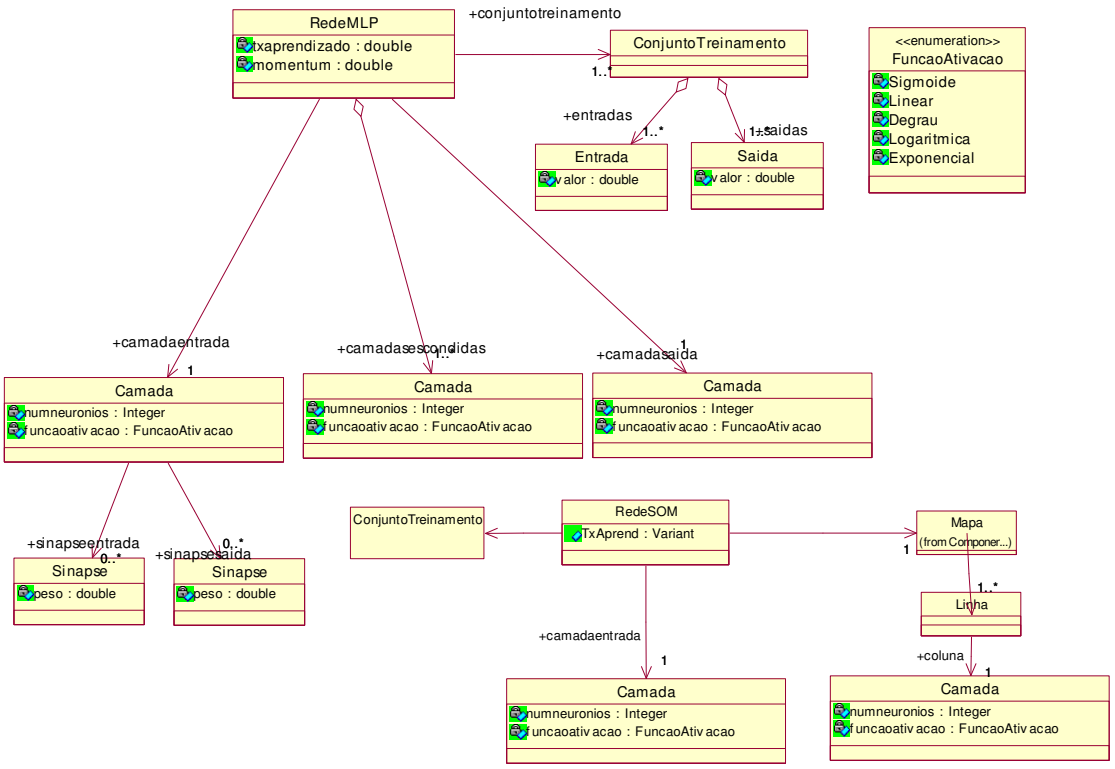


Figura 187 – Classes de domínio para redes neurais MLP e SOM

A Figura 188 exibe as classes de domínio de problema das interfaces ISBGenetico eespecifica interfaces para implementação de algoritmos genéticos.

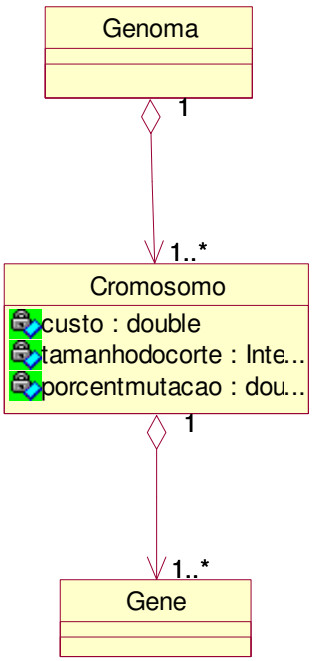


Figura 188- Classes de domínio para algoritmos genéticos

Classes CWM-G

Este pacote estende os pacotes OLAP e *data mining* do modelo CWM, de forma a torná-lo adequado ao tratamento de informações espaciais. Isto é feito pelo acréscimo de hierarquias, níveis, dimensões, medidas e atributos espaciais.

A Figura 189 exibe o diagrama que mostra a extensão realizada sobre o modelo CWM de forma a comportar atributos espaciais e métodos de análise espacial.

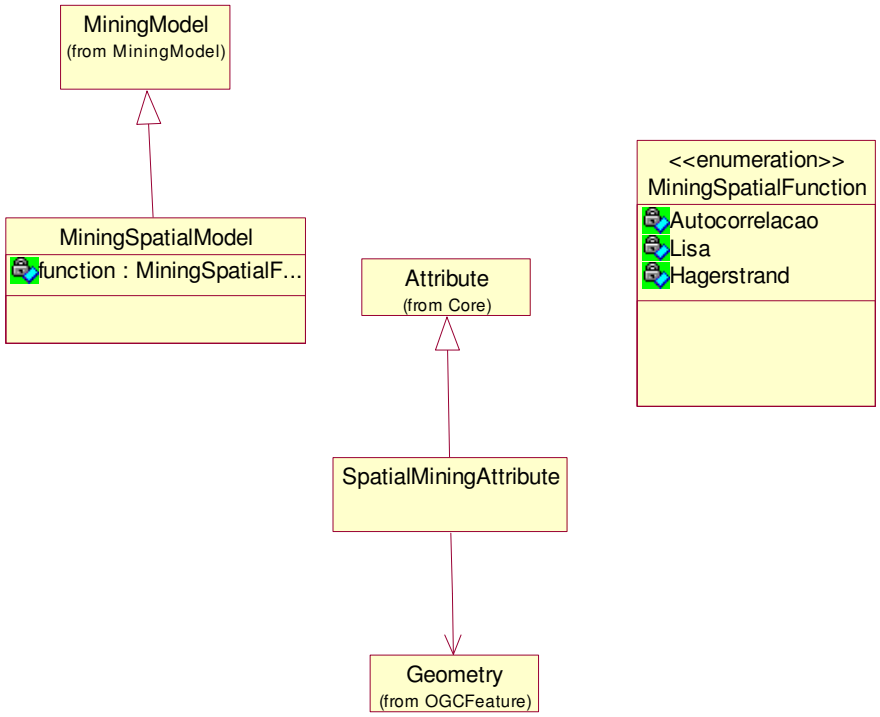


Figura 189-Extensões espaciais ao pacote data mining do modelo CWM

A Figura 190 exibe o diagrama que mostra a extensão realizada sobre o modelo OLAP do CWM de forma a comportar atributos atributos, hierarquias e níveis espaciais. Dimensões e medidas encontram-se estruturados conforme descrito no item 3.1.1.2.5.

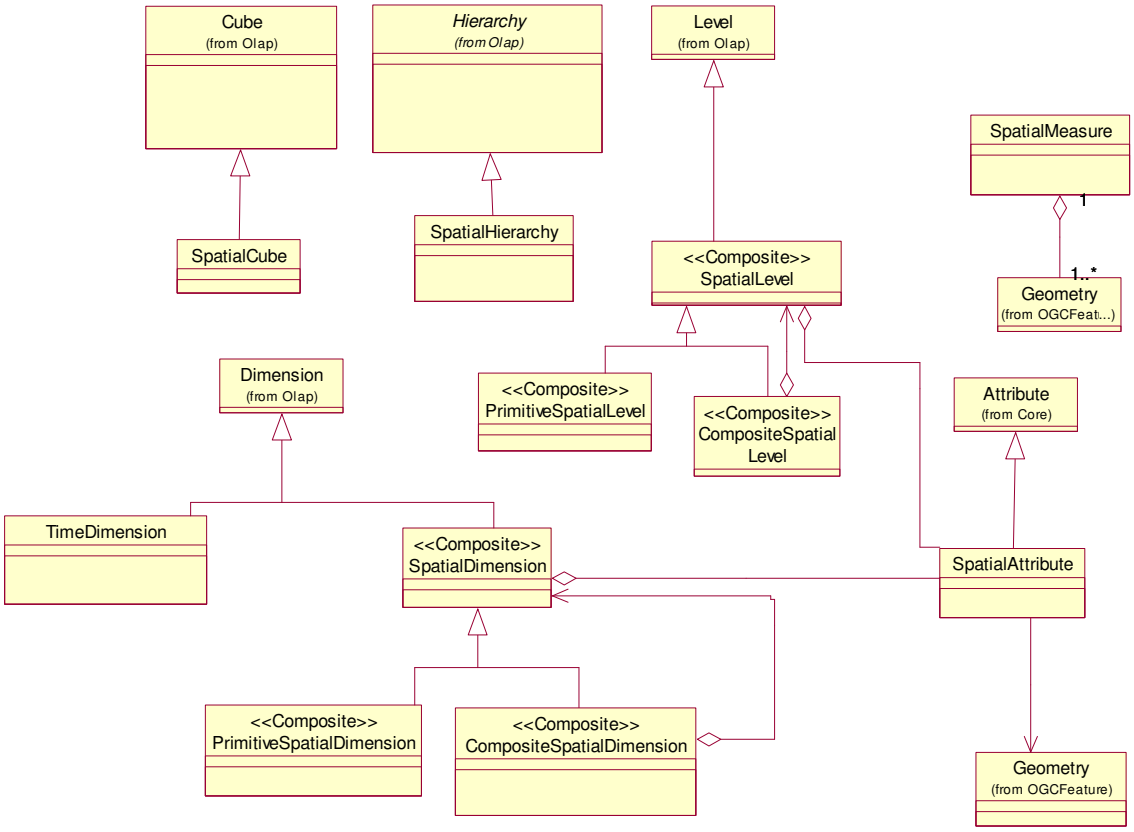
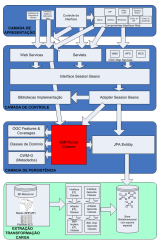


Figura 190 – Diagrama de classes que exibe extensões realizadas sobre o modelo OLAP CWM

B.3.2-Classes EMF/Ecore



O Modelo ECore do *Eclipse Modeling Framework (EMF)* é a base para criação de *abstract factories (*Gamma, 1998) utilizadas para instanciar as classes de domínio de problema. Conforme exibido na Figura 191, o EMF provê um mecanismo eficiente para geração rápida do modelo Ecore e daí, do código Java correspondente segundo o arcabouço Ecore, a partir de um modelo em formato XML, Java anotado ou MDL do *Rational Rose*. Os modelos apresentados no item B.3.1 estão armazenados em formato .MDL. Eles são convertidos em classes e interfaces formatadas conforme o arcabouço *Ecore* do EMF e utilizadas na aplicação via interfaces ISBs e suas implementações correspondentes (ASBs).

Figura 191-Processo de geração de código Java do EMF

A Figura 192 exibe o diagrama de classes geradas a partir do modelo UML editado no Rational Rose, segundo o arcabouço Ecore do EMF. As classes do modelo são convertidas em interfaces Java (*InterfacesDominio)* que são implementadas pelas classes *ImplementacaoDominio* (em essência são Java Beans acessíveis via métodos *getters* e *setters).* Os objetos são criados via fábricas de objetos que são especificadas via *InterfaceFactory* e implementadas via *ImplementacaoFactory.*

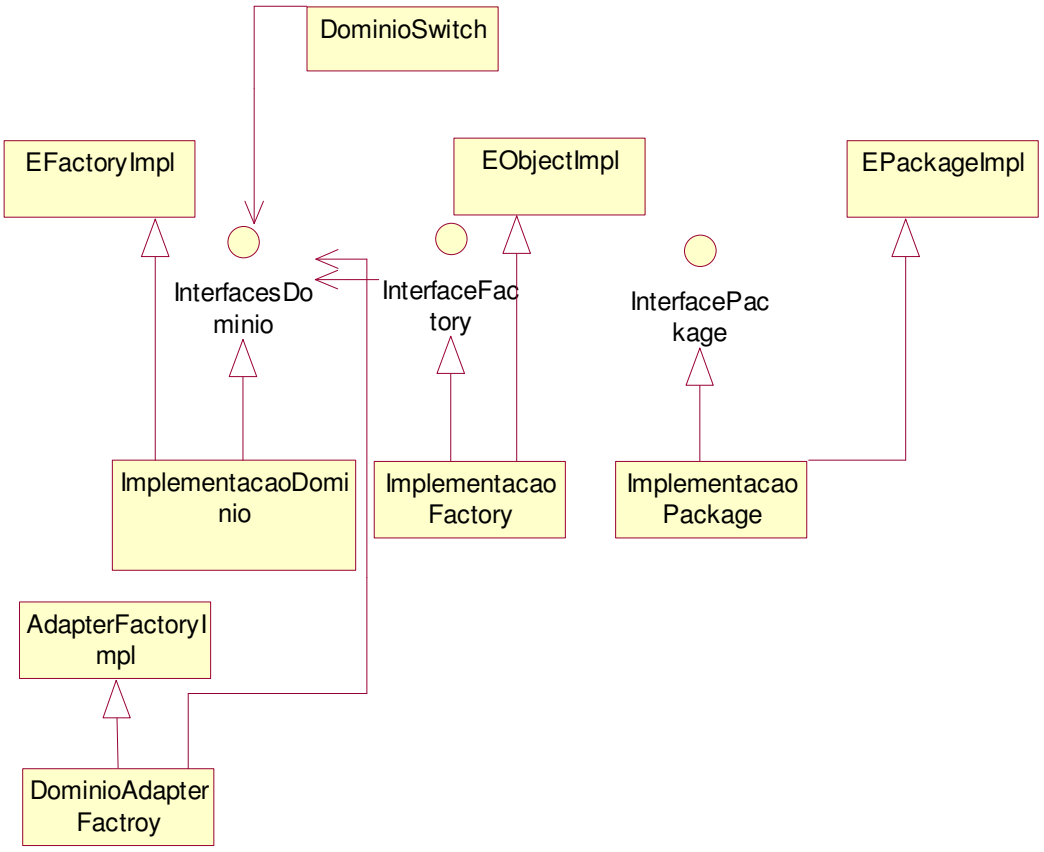
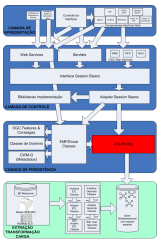


Figura 192-Arcabouço Ecore, gerado a partir de arquivos Rose/UML

B.3.3-JPA *Entity*



A JPA-*Java Persistense API* (Sun, 2007) é a base utilizada para implementação de persistência. As classes JPA provêem um mecanismo para inserção, exclusão e alteração de dados em uma base de dados, bem como um mecanismo de implementação de relacionamentos entre classes e de geração de *queries*. Na aplicação desenvolvida para consulta a bases multidimensionais (ver item 4.5) ,

classes JPA foram criadas para acesso a dados nas tabelas de fatos e dimensões e para a realização das consultas (*queries).*

A implementação da persistência utiliza o arcabouço EMF-Teneo (IBM,2009b), que associa classes geradas pelo EMF com persistência via tabelas de bancos de dados e execução de *queries* providos pelo JPA.

A Figura 193 exibe a estrutura de uma entidade (*entity)* JPA associada à implementação de uma classe de domínio EMF. Os métodos *get* e *set* buscam e armazenam os valores de campos no banco de dados. Podem ser criadas associações entre entidades e realizarem-se *queries* sobre a estrutura criada.

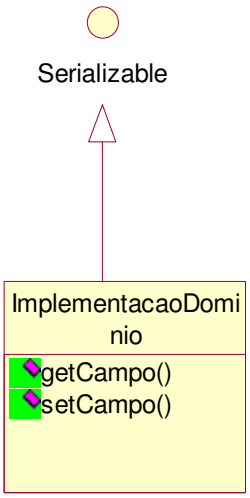
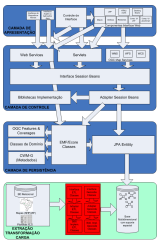


Figura 193- Estrutura essencial de uma entidade JPA

B.4 -Extração-Transformação-Carga

B.4.1-Classes ETL e de Geocodificação



Este módulo especifica interfaces para extração de dados espaciais e georreferenciamento, bem como para realização de operações de tratamento de dados. As interfaces para extração de dados foram implementadas e um aplicativo foi construído para realizar a extração de múltiplas origens e georreferenciamento de dados provenientes de arquivos MIF e SHP. A Figura 194 exibe a interface desta aplicação. No painel da esquerda, escolhem-se as fontes de dados: banco de dados (via ODBC), planilhas Excel (csv), arquivos Mapinfo (MIF) e ArcGis (SHP). Estas fontes são associadas a fatos e dimensões de um *data mart,* listadas no painel da direita. Pressionando-se o botão *V* , a tabela na parte inferior da tela armazena as associações. Pressionando-se o botão extrair, os dados (convencionais e espaciais) são armazenados em um banco de dados.

A Figura 193 exibe o diagrama de classes que mostra as interfaces para extração de dados implementadas no aplicativo.

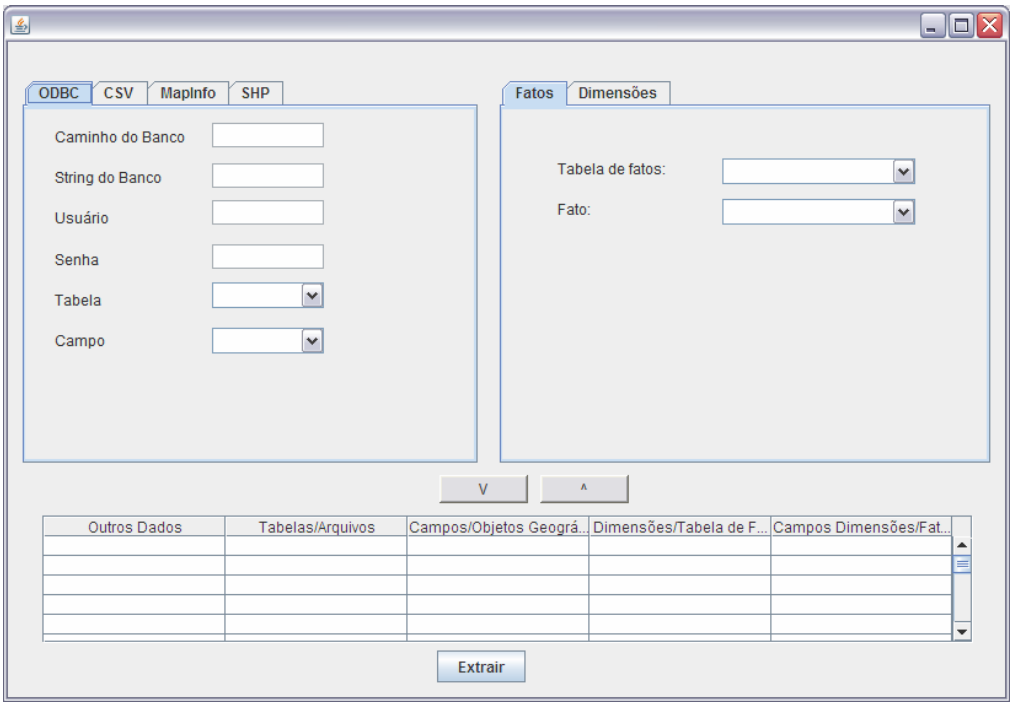


Figura 194-Interface para extração e georreferenciamento de dados

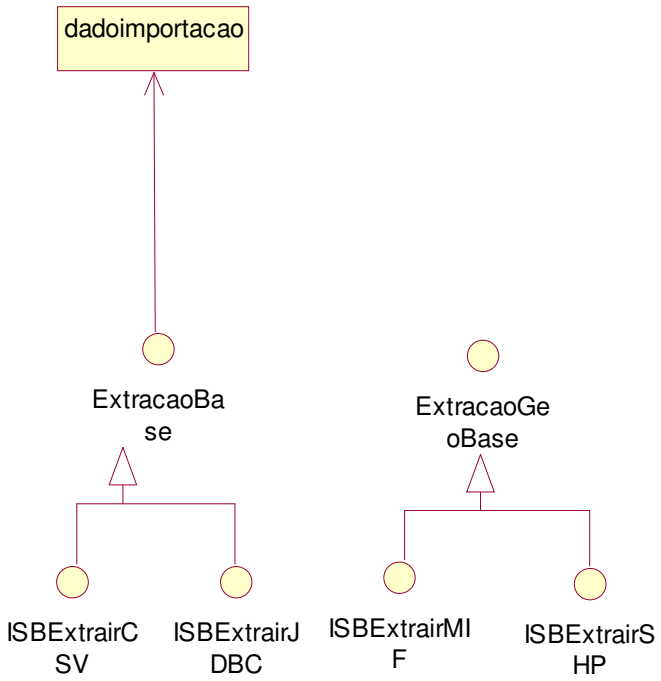
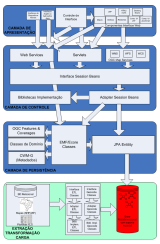


Figura 195-Interfaces para extração de dados convencionais e espaciais

B.4.2-Bases de Dados Multidimensionais Espaciais



Este módulo comporta bases de dados com extensão espacial, que armazenam tabelas de modelos multidimensionais. Bancos de dados como MySQL espacial, PostGIS e Oracle *Spatial* possuem extensões que implementam as especificações espaciais do OGC. As aplicações desenvolvidas nesta tese utilizam o banco de dados MySQL.

APÊNDICE C-APIs-*Application Program Interfaces*

Neste apêndice apresentam-se em formato Javadoc as APIs das ISBs de alguns algoritmos de *data mining* convencional e espacial, modelos de análise espacial, serviços OGC, técnicas de Inteligência Artificial e operações para extração e transformação de dados convencionais e espaciais. Cada Interface, quando for o caso, possui uma introdução teórica que normalmente apresenta o algoritmo, o modelo ou a técnica que especifica. As interfaces aqui apresentadas estão também representadas em diagramas de classes no Apêndice B.

estracaodados

Interface ExtracaoBase

All Known Subinterfaces:

ISBExtrairCSV

```
public interface ExtracaoBase
```

Interface base para extração de dados de arquivos CSV ou de bancos de dados

---

Method Summary

vc	**Extrair**(dadosimportacao[] campos, java.lang.String origem, java.lang.String tabdestino, java.lang.String arquivoSQL) Inicia o processo de extração com os dados necessários

Method Detail

Extrair

```
void Extrair(dadosimportacao[] campos,
             java.lang.String origem,
             java.lang.String tabdestino,
             java.lang.String arquivoSQL)
```

Inicia o processo de extração com os dados necessários

Parameters:

`campos` - Vetor com dados de importacao de campos

`arquivocsv` - Nome da tabela ou arquivo de origem

`tabdestino` - Tabela de destino

`arquivoSQL` - Nome do arquivo SQL onde será gravada a importação

estracaodados

Interface ExtracaoGeoBase

All Known Subinterfaces:
ISBExtrairMIF, ISBExtrairSHP

```
public interface ExtracaoGeoBase
```

Interface base para extração de dados geográficos e convencionais de arquivos MIF ou SHP

Method Summary

vo	**geraarquivosql**() Gera arquivo contendo comandos SQL para carregar dados extraídos em um banco de dados.
vo	**iniciar**(java.lang.String nomearquivorigem, java.lang.String[] nomecamposdados, java.lang.String nomegeometria, java.lang.String nomearquivosql) Inicia o processo de extração de dados geográficos com os dados necessários
java.lang.String	**retornadados**() Retorna vetor de dados lidos no arquivo
Geometry	**retornageometrias**() Retorna vetor de geometrias lidas no arquivo

Method Detail

retornageometrias

```
Geometry[] retornageometrias()
```

Retorna vetor de geometrias lidas no arquivo

Returns:

Vetor de geometrias

---

retornadados

```
java.lang.String[] retornadados()
```

Retorna vetor de dados lidos no arquivo

Returns:

Vetor de dados

---

iniciar

```
void iniciar(java.lang.String nomearquivorigem,
             java.lang.String[] nomecamposdados,
             java.lang.String nomegeometria,
             java.lang.String nomearquivosql)
```

Inicia o processo de extração de dados geográficos com os dados necessários

Parameters:

`nomearquivoorigem` - Nome do arquivo MIF ou SHP
`nomecamposdados[]` - Vetor contendo o nome dos campos de dados a serem buscados nos arquivos
`nomegeometria` - Nome da geometria a ser buscada (por exemplo, Region, Point...)
`nomearquivosql` - Nome do arquivo SQL de destino

---

geraarquivosql
`void` **`geraarquivosql`**`()`
Gera arquivo contendo comandos SQL para carregar dados extraídos em um banco de dados.

analisedados.tecnicasdatamining.dmclusteringespacial

Interface ISBAGM

All Superinterfaces:
ISBBase, ISBClustering, ISBClusteringEspacial

```
public interface ISBAGM
extends ISBClusteringEspacial
```

Interface que especifica métodos para implementação do Algoritmo AGM (Árvore Geradora Mínima)

Em (Assunção, 2000) é apresentado um método para *clustering* de objetos espaciais baseado na teoria de grafos. Nesse contexto define-se o seguinte:

- Um *cluster* será considerado como um subconjunto de áreas internamente homogêneas.
- Um *cluster* é dito *conectado* quando cada área componente é vizinha de pelo menos uma área pertencente ao *cluster* (entretanto, *clusters* formados por uma única área também são considerados conectados).
- Em um grafo G, um *caminho* entre dois nós , v_i e v_k , é uma seqüência de nós v_1 , v_2 ...v_K , que são conectados pelas arestas (v_1,v_2), (v_2,v_3)...(v_{k-1},v_k)
- Um grafo é conexo, se existe ao menos um caminho ligando dois nós quaisquer do mesmo.
- Um circuito em um grafo é um caminho onde os nós inicial e final são os mesmos.
- Uma árvore é um grafo conexo que não contém circuitos.
- Uma árvore geradora para um grafo *G*, é um sub-grafo que é uma árvore que contém todos os nós de *G*..
- Se atribuirmos um custo às arestas de um grafo, o *custo do grafo* é a soma dos custos das arestas do grafo.
- Uma *árvore geradora mínima* é uma árvore geradora com custo mínimo.

Os passos do método são os seguintes:

- 1-O método parte de um mapa contendo regiões poligonais (áreas) vizinhas, que é reduzido a um grafo onde cada nó representa um polígono e polígonos vizinhos são conectados por arestas.

- 2-Atribui-se um *custo* às arestas do grafo, que será a dissimilaridade entre os nós, calculada, por exemplo, pela distância euclidiana entre os valores dos atributos dos dois nós.
- 3-Construir a árvore geradora mínima. Um algoritmo tradicionalmente utilizado é o de *Prim*, que parte de uma árvore T_1 e a incrementa recursivamente:
    - *Passo 1*: Tomar qualquer nó v_i do grafo e fazer $T_k=T1=\{v_i\}$;
    - *Passo 2*: Repetir o seguinte, até que todos os nós tenham sido incluídos na árvore:
        - Encontrar uma aresta e_k de menor custo que ligue qualquer nó pertencente a T_k a um outro nó que não pertença a T_k.
        - Gerar a árvore T_{k+1}, resultante do acréscimo do novo nó a T_k.
- 4-Particionamento da árvore geradora mínima por meio de eliminação de arestas, de forma a se obter os *clusters.*Uma escolha natural seria eliminar sucessivamente as arestas de maior custo, até atingir o número de *clusters* desejado (ou seja, se desejamos *k clusters, k-1* arestas devem ser retiradas). Entretanto, pela topologia final da árvore geradora mínima, que tem arestas de custo maior adicionadas ao final do processo de geração, esse procedimento tende a formar conglomerados com diferença grande de número de polígonos em cada um. Assim sendo, (Assunção, 2003) recomenda que tal particionamento seja feito por meio de uma alteração no custo de cada aresta,que tende a deixar a distribuição mais homogênea e com um número mais uniforme de polígonos, descrita no primeiro passo do algoritmo de particionamento a seguir:

    - *Passo 1***:** Alterar o custo de cada aresta do grafo por meio das seguintes relações:

$C_e = SSD_T - SSD_e$

$$SSD_T = \sum_{j=1}^{m} \sum_{i=1}^{n} (x_{ij} - \bar{x_j})^2$$

$$\bar{x_j} = \frac{1}{n} \sum_{i=1}^{n} x_{ij}$$

$SSD_e = SSD_{Ta} + SSD_{Tb}$

onde:

C_e é o custo de uma aresta e;
SSD_T é a soma dos quadrados dos desvios associados a uma árvore T;
n é o número total de nós em T;
x_{ij} é o atributo j do objeto i;
m é o número de atributos considerados;
x_j *é* o valor médio do atributo j;
$SSD_{Ta} + SSD_{Tb}$ é a soma dos quadrados dos desvios das duas sub-árvores T_a e T_b geradas pela retirada da aresta e da árvore T.

- *Passo 2:* Até que o número de k de *clusters* desejado seja atingido, retirar arestas da árvore, começando pela de custo máximo, recursivamente, ou seja para cada uma das duas árvores resultantes, retira-se a aresta de maior custo *dentre aquelas da árvore em questão.*

É claro que, quanto menor for $SSD_{e,}$ tanto mais homogêneos (com menor desvio entre seus atributos) serão os *clusters* resultantes. Assim sendo, ao retirarmos arestas de maior custo, a tendência é gerar *clusters* mais homogêneos.

No método descrito anteriormente, o custo de todas as arestas é avaliado de forma a se obter a de maior custo. Em casos que envolvam um número elevado de objetos e atributos, o esforço computacional para avaliar todas as soluções cresce significativamente. Para melhorar a eficiência do método e sua aplicabilidade a maiores volumes de dados, (Assunção, 2003) propõe a utilização de técnicas de otimização na fase de particionamento da árvore geradora mínima. Essa otimização é vista como a busca em um espaço de soluções S_i, de forma que seja identificada uma solução aceitável sem necessidade de avaliar todas as

possíveis. A função objetivo a ser minimizada é $f(S_i) = SSD_e$. Uma solução S_i é a aresta escolhida para remoção na árvore. Uma solução vizinha a S_i é obtida pela retirada de uma aresta que possui um dos nós em comum com S_i. A estratégia de busca é a seguinte (Assunção, 2003):

Passo1: escolha da solução inicial *Si* em *S*. Fazer $S^* = Si$; $k^* = k = 0$; e incluir *Si* na lista de soluções visitadas *T*);

Passo 2: Fazer $k = k + 1$. Escolher na lista *T* a solução que terá a vizinhança expandida, gerando o conjunto de soluções promissoras (*V**);

Passo 3: Avaliar as soluções em *V**; Armazenar soluções avaliadas em *T*; escolher a melhor solução, *Sj*, em *V**.

Passo 4: Se $f(Sj) < f(S^*)$, então: $S^* = Sj$; $k^* = k$;

Passo 5: Verificar condição de parada ($k - k^* > 8$). Senão satisfeita, voltar ao *passo 2*.

Segundo (Assunção, 2003), o conjunto de soluções promissoras, *V**, indica as soluções que serão avaliadas na iteração corrente e correspondem à expansão na vizinhança de uma solução já avaliada, escolhida em *T* no passo 2. Para esta escolha é utilizada uma função de avaliação:

$f' = max(SSD_{Ta}, SSD_{Tb})$

Será escolhida a solução que apresentar um menor valor para *f'* e que ainda tenha ao menos uma solução vizinha ainda sem avaliação. Deve-se observar que *f'* evita a busca em ramos da árvore com poucos nós.

(Câmara, 2001), ressalta que esse método diminui o número de possibilidades a serem investigadas em cada iteração.

Method Summary

	obterAGM(GrafoVizinhanca grafoviz) Retorna Árvore Geradora Mínima

Methods inherited from interface analisedados.tecnicasdatamining.dmclusteringespacial.ISBClusteringEspacial
calcularvizinhancaespacial, retornanovosclusteres

Methods inherited from interface analisedados.tecnicasdatamining.dmclustering.ISBClustering
iniciar, LerInstancias, RetornaClassificacaoInstancia, RetornaContagemClusters, RetornaInstanciaPorCluster, RetornaInstancias, RetornaNumClusters, RetornaTamanhoClusters, SetNumClusters

Method Detail

obterAGM

AGM **obterAGM**(GrafoVizinhanca grafoviz)

Retorna Árvore Geradora Mínima

Parameters:

grafoviz - grafo de vizinhança entre instâncias

Returns:

Árvore Geradora Mínima

See Also:

analisedados.tecnicasdatamining.dmclustering

Interface ISBCLARA

All Superinterfaces:
ISBBase, ISBClustering, ISBPAM

```
public interface ISBCLARA
extends ISBPAM
```

Especifica interfaces de operações do algoritmo CLARA

Segundo (Ng,2002), o algoritmo CLARA é bastante adequado a grandes bases de dados, pois ao invés de procurar por objetos representativos em todo o conjunto de dados, ele obtém uma amostra dele, aplica o algoritmo PAM a essa amostra e encontra os *medoids* da mesma. O ponto importante, é que se a amostra é obtida de maneira realmente randômica, os *medoids* da mesma aproximam-se dos *medoids* de todo o conjunto de dados. Para conseguir melhores resultados, o algoritmo utiliza várias amostras e obtém o melhor agrupamento ao final. Para tal, a qualidade dos *clusters* é obtida pela dissimilaridade média de todos os objetos no conjunto de dados, não apenas os das amostras (Ng, 2002). Experimentos citados por (Ng, 2002) indicam que 5 amostras de tamanho 40+2k fornecem resultados satisfatórios. Os passos do algoritmo CLARA são os seguintes:

1-Repetir os passos seguintes por 5 vezes:
2-Obter randomicamente uma amostra de 40+2k objetos em todo o conjunto de dados e chamar o algoritmo PAM para encontrar k *medoids* para a amostra.
3-Para cada objeto de todo o conjunto de dados, determine qual dos k *medoids* é o mais similar ao objeto em questão.
4-Calcular a dissimilaridade média do *clustering* obtido no passo anterior (utilizando (1), por exemplo). Se esse valor for menor que o mínimo atual, faça-o o mínimo atual e mantenha os k *medoids* encontrados no passo 2 como os melhores obtidos até então.
5-Retornar ao passo 1 para iniciar a próxima iteração.

Este algoritmo possui ordem de complexidade $O(k(40+k)^2+k(n-k))$ por iteração, portanto melhor que a do algoritmo PAM (Ng, 2002).

Method Summary

Methods inherited from interface analisedados.tecnicasdatamining.dmclustering.ISBPAM

RetornaMedoidCluster

Methods inherited from interface analisedados.tecnicasdatamining.dmclustering.ISBClustering

iniciar, LerInstancias, RetornaClassificacaoInstancia, RetornaContagemClusters, RetornaInstanciaPorCluster, RetornaInstancias, RetornaNumClusters, RetornaTamanhoClusters, SetNumClusters

analisedados.tecnicasdatamining.dmclustering

Interface ISBCLARANS

All Superinterfaces:

ISBBase, ISBClustering, ISBPAM

```
public interface ISBCLARANS
extends ISBPAM
```

Especifica interfaces de operações do algoritmo CLARANS

Proposto por (Ng, 2002), essa versão melhorada do CLARA também procura pelos k *medoids* em um subconjunto das possíveis combinações de objetos, mas ele realiza uma busca contínua e aleatória por locais ótimos(Câmara, 2001). Segundo (Ng, 2002), é mais adequado à mineração de dados espaciais por convergir mais rapidamente em conjuntos de dados extensos, como os normalmente encontrados em bases de dados espaciais.

Segundo (Ng, 2002) o processo de procura por k *medoids* pode ser abstratamente visto como uma busca em um grafo. Neste grafo, um nó S_i é visualizado como um conjunto de k objetos, os *medoids* selecionados. Dois nós S_1 e S_2 são considerados vizinhos se seus conjuntos diferem por apenas um objeto. Mais formalmente, eles serão vizinhos se $|S_1 \cap S_2| = K - 1$. Assim sendo, cada nó vai possuir k(n-k) vizinhos. Uma vez que um nó representa uma coleção de k *medoids,* corresponderá a um agrupamento (*clustering)* e pode-se, portanto, atribuir um custo a cada um deles. Esse custo pode ser definido como a dissimilaridade total entre cada objeto e o *medoid* de seu *cluster* (calculado por exemplo, pela equação (1) acima. Os passos do algoritmo CLARANS são os seguintes:

1: Entrar com os parâmetros: número de mínimos locais (*numminlocal*) e número máximo de vizinhos avaliados por iteração (*nummaxvizinhos*). Fazer *i = 1* e *customínimo* = valor grande
2: Arbitrar um nó do grafo como *nocorrente*.
3: Fazer *j = 1*.
4: Escolher aleatoriamente um nó vizinho a *nocorrente*, *S*, e calcular o custo diferencial entre os dois nós.
5: Se *S* tem menor custo, tornar *S* o *no_corrente* e voltar ao *passo 3*.
6: Caso contrário, incrementar *j* de 1. Se *j* <= *nummaxvizinhos* , ir
para o *passo* 4.

7: Caso contrário, quando *(j > nummaxvizinhos)* comparar custo de *nocorrente* com *customínimo*. Se menor, fazer *customínimo* igual ao custo de *nocorrente* e atribuir *nocorrente* a *melhorno*. .

8: Incrementar *i* de uma unidade. Se *i* > *numlocal* , informar *melhorno* e sair, senão, voltar ao passo 2;

Algumas observações podem ser feitas, comparando-se os algoritmos PAM, CLARA e CLARANS (Câmara, 2001 & Ng, 2002):

- O algoritmo PAM, pode ser visto como uma busca no grafo, onde em cada passo, todos os vizinhos do nó corrente são examinados de forma a encontrar aquele que minimize f(**π**). Para um grande número de valores de n e k , examinar todos os k(n-k) vizinhos de um nó consome tempo, o que contribui para a ineficiência desse algoritmo
- No algoritmo CLARA temos um sub-grafo do grafo original, contendo um subconjunto de nós, determinados pela combinação dos m objetos da amostra tomados k a k. Portando, somente uma parte menor das partições possíveis são investigadas.
- O CLARANS, em cada passo, verifica apenas parte dos nós vizinhos. Porém, os vizinhos examinados são escolhidos aleatoriamente. Dessa forma, o CLARANS não realiza uma busca em um sub-grafo restrito e pré-definido por uma amostra, e sim, investiga parte dos nós vizinhos definidos dinamicamente, durante a sua execução.

(Ng, 1994) e (Ng, 2002) apresentam evidências experimentais que demonstram a superioridade do algoritmo CLARANS sobre o CLARA e deste sobre o PAM.

Method Summary

vc	**setNumMaxVizinhos**(int nmv) Ajusta número máximo de vizinhos avaliados por iteração
vc	**setNumMinLocal**(int nml) Ajusta número de mínimos locais

Methods inherited from interface analisedados.tecnicasdatamining.dmclustering.ISBPAM

RetornaMedoidCluster

Methods inherited from interface

analisedados.tecnicasdatamining.dmclustering.ISBClustering

iniciar, LerInstancias, RetornaClassificacaoInstancia, RetornaContagemClusters, RetornaInstanciaPorCluster, RetornaInstancias, RetornaNumClusters, RetornaTamanhoClusters, SetNumClusters

Method Detail

setNumMinLocal

`void` **`setNumMinLocal`**`(int nml)`

Ajusta número de mínimos locais

Parameters:

`nml` - número de mínimos locais

See Also:

---

setNumMaxVizinhos

`void` **`setNumMaxVizinhos`**`(int nmv)`

Ajusta número máximo de vizinhos avaliados por iteração

Parameters:

`nml` - número máximo de vizinhos avaliados por iteração

See Also:

analisedados.tecnicasdatamining.dmclustering

Interface ISBClustering

All Superinterfaces:
ISBBase
All Known Subinterfaces:
ISBAGM, ISBCLARA, ISBCLARANS, ISBClusteringEspacial, ISBCOBWEB, ISBDendograma, ISBEM, ISBKMeans, ISBkmeansEspacial, ISBPAM

```
public interface ISBClustering
extends ISBBase
```

Interface que especifica métodos comuns aos algoritmos de clustering

---

Method Summary

vo	**iniciar**() Iniciar a classe
vo	**LerInstancias**(ConjuntoInstancias instancias) Lê um conjunto de instâncias
i	**RetornaClassificacaoInstancia**(Instancia inst) Retorna o cluster ao qual pertence a Instancia
int[][]	**RetornaContagemClusters**() Retorna, para cada cluster a contagem de frequencias para valores de cada atributo
ConjuntoInstancias	**RetornaInstanciaPorCluster**() Retorna cada cluster com as instâncias neles contidas
ConjuntoInstanci	**RetornaInstancias**(int numclusters) Retorna uma instância dado o índice de um cluster
i	**RetornaNumClusters**() Retorna número de clusters
int	**RetornaTamanhoClusters**() Retorna o tamanho de cada cluster
vo	**SetNumClusters**(int n) Armazena número desejado de clusters

Method Detail

iniciar
`void iniciar()`
Iniciar a classe
See Also:

RetornaNumClusters
`int RetornaNumClusters()`

Retorna número de clusters
Returns:
Número de clusters
See Also:

RetornaContagemClusters
`int[][][] RetornaContagemClusters()`
Retorna, para cada cluster a contagem de frequencias para os valores de cada atributo
Returns:
Frequencia dos valores de cada atributo
See Also:

RetornaClassificacaoInstancia
`int RetornaClassificacaoInstancia(Instancia inst)`
Retorna o cluster ao qual pertence a Instancia
Parameters:
`inst` - Instancia cujo cluster deseja-se retornar
Returns:
Índice da instância
See Also:

RetornaTamanhoClusters
`int[] RetornaTamanhoClusters()`
Retorna o tamanho de cada cluster
Returns:
Tamanho de cada cluster
See Also:

SetNumClusters
`void SetNumClusters(int n)`
Armazena número desejado de clusters
Parameters:
`n` - Número de clusters
See Also:

RetornaInstancias
`ConjuntoInstancias RetornaInstancias(int numclusters)`
Retorna uma instância dado o índice de um cluster
Parameters:
`índice` - do cluster cujas instâncias deseja-se retornar
Returns:
O centróide do cluster
See Also:

RetornaInstanciaPorCluster
`ConjuntoInstancias[] RetornaInstanciaPorCluster()`
Retorna cada cluster com as instâncias neles contidas
Returns:

Clusters com respectivas instâncias
See Also:

LerInstancias
`void` **`LerInstancias`**`(`ConjuntoInstancias `instancias)`
Lê um conjunto de instâncias
Parameters:
`instancias` - conjunto de instancias a serem agrupadas
See Also:

analisedados.tecnicasdatamining.dmclusteringespacial

Interface ISBClusteringEspacial

All Superinterfaces:
ISBBase, ISBClustering
All Known Subinterfaces:
ISBAGM, ISBkmeansEspacial

```
public interface ISBClusteringEspacial
extends ISBClustering
```

Interface que especifica métodos comuns aos algoritmos de clustering espacial

A análise de *clustering* espacial possui uma singularidade, que é a associação entre um objeto geográfico georreferenciado(um ponto, uma área ou um polígono por exemplo) e um conjunto de atributos que o caracterizam. Assim sendo, nessa abordagem espacial, consideram-se duas medidas de distância: uma geográfica e outra baseada na medida de distância (dissimilaridade) entre valores de atributos (Assunção, 2003).

Adaptação das Técnicas Convencionais quando se Consideram Informações Georreferenciadas(Câmara, 2001)

(Gordon, 1996) propõe três abordagens para *clustering* quando se considera a contigüidade espacial existente entre objetos:

- *Clustering em dois estágios*. No primeiro estágio, não se considera a informação espacial e um procedimento de *clustering* convencional é executado, utilizando somente os atributos não-espaciais. No segundo estágio, os *clusters* são reavaliados observando as relações de vizinhança entre os objetos. Assim, objetos similares agrupados em um *cluster* no primeiro estágio mas sem contigüidade espacial, serão separados no segundo estágio formando regiões distintas.
- A dissimilaridade entre os objetos é avaliada considerando-se simultaneamente a posição geográfica dos mesmos e seus atributos não-espaciais. As coordenadas do centróide são consideradas atributos adicionais. Utilizam-se algoritmos convencionais de *clustering*, onde a avaliação da dissimilaridade contém duas componentes ponderadas, uma para o espaço de atributos e a outra para a distância geográfica. Se o peso dado para a componente correspondente à distância geográfica for forte o suficiente, os grupos resultantes do processo

de classificação serão contíguos. (Wise, 1997) propõe uma função objetivo que considera três componentes ponderados, dada por:

$$f_0 = w_h f_h + w_c f_c + w_i f_i$$

onde

w_h ,w_c *e* w_i são os pesos referentes às três componentes f_h, f_c *e* f_i

f_h : é uma medida da variância de um ou mais atributos dos objetos nas regiões;

f_c : é uma medida da soma das distâncias dos objetos aos centros dos clusters, considerando todas as regiões;

f_i : é uma medida da soma dos desvios dos valores de um atributo dos objetos membros das regiões em relação ao valor médio.

- Utilização de uma matriz para representar a relação de vizinhança entre os objetos, denominada *matriz de contigüidade*, *C*. Cada elemento cij da matriz indica se os objetos *i* e *j* são contíguos (igual a 1) ou não (igual a 0). Dentro desta terceira abordagem, nos algoritmos hierárquicos aglomerativos, dois clusters mais similares são aglutinados somente se existem dois objetos contíguos, um em cada *cluster*. Nos algoritmos de particionamento adaptados para considerar contigüidade, um objeto só pode ser realocado a *clusters* que possuam ao menos um objeto contíguo a ele.

A verificação da existência de objetos contíguos é bem mais fácil de ser executada nos algoritmos hierárquicos. Uma vez definidos os dois *clusters* com menor dissimilaridade, *Ci* e *Cj*, procuramos nos vizinhos de cada um dos objetos de *Ci*, um objeto pertencente a *Cj*. Encontrado um vizinho pertencente a *Cj*, a busca é interrompida e a fusão efetivada. Não encontrando, o próximo par de *clusters* de menor dissimilaridade é avaliado. Nos algoritmos de particionamento, a verificação da contiguidade espacial é mais complexa, pois ela é dependente da ordem pela qual os objetos vão sendo associados aos *clusters*, necessitando de reavaliações até que todos os objetos possam ser atribuídos a um dos *clusters* contíguos.

Clustering de Objetos Poligonais Convexos (Ng, 2002)

Os algoritmos supra descritos aplicam-se correntemente a objetos espaciais do tipo ponto, para os quais as distâncias euclidiana ou táxi são convenientes. Entretanto, na prática, um sem-número de objetos a serem agrupados são de geometria mais complexa, como polígonos ou áreas(Ng, 2002). A questão aqui colocada é como calcular a distância entre dois polígonos eficientemente, de maneira adequada ao processo de *clustering*.. Aproximar tais polígonos por um ponto-o centróide, por exemplo- não é muito adequado, já que os objetos poligonais podem ter formas e tamanhos bem distintos, produzindo *clusters* de má qualidade. Representá-los por vários pontos também pode ser inadequado, pois tais pontos podem, ao final do processo de *clustering,* não ser alocados ao mesmo *cluster.* Portanto, este item como objetivo apresentar soluções que permitam contornar tais problemas e adaptar os algoritmos de *clustering* para uso em objetos poligonais convexos.

Cálculo da Distância Exata de Separação

Dados dois polígonos A e B, a *distância de separação* entre eles é definida como sendo a menor distância entre quaisquer dois pontos em A e B. Essa distância é computada em duas situações: quando A e B se interceptam ou não. No primeiro caso, a distância de separação é nula e no segundo, calculada conforme descrito acima.

Uma forma mais eficiente para cálculo da distância de separação é apresentada em (Kirkpatrick, 1993). Ao invés de percorrer todos os vértices dos polígonos, esse algoritmo identifica duas cadeias de vértices e segmentos de linha consecutivos (uma cadeia de cada polígono) de tal forma que um vértice de uma cadeia pode “’ver” ao menos um vértice na outra cadeia. Se P é um vértice em A e Q um vértice em B, diz-se que P e Q “vêem” um ao outro se o segmento de linha que liga P a Q não intercepta o interior de nenhum dos polígonos. Logo, por definição, a distância de separação entre os polígonos deve ser a mínima distância entre qualquer par de pontos (não necessariamente vértices) nas duas cadeias. Utilizando-se uma busca binária, o algoritmo encontra então, a distância desejada.

Aproximação pela Mínima Distância Entre Vértices.

A *aproximação MV* entre dois polígonos é a menor distância existente entre os vértices dos mesmos. A complexidade do algoritmo para se obter essa aproximação é menor que a do algoritmo descrito no item anterior, mas na maior parte das vezes ele fornece resultados idênticos com a mesma acuidade, exceto para objetos com mais de 20 vértices e arestas. Como para a grande maioria das aplicações de mineração de dados, o número de vértices e arestas dos objetos espaciais é menor que 20, justifica-se o uso dessa aproximação para otimização do desempenho. Entretanto, a Figura 24 exibida abaixo mostra que a distância de separação entre polígonos nem sempre é igual à mínima distância entre vértices e que a primeira nunca ultrapassa a última. Assim sendo, a aproximação sempre superestima a distância de separação. O ponto chave é verificar se isso não compromete a qualidade do processo de *clustering*.

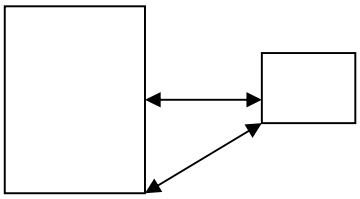


Figura 24-Aproximação MV e distância de separação são diferentes (Fonte: (NG, 2002))

É trivial visualizar que a distância entre centróides também tende a superestimar a distância de separação. O ponto aqui, é verificar se a estimativa por centróides e a aproximação MV têm efeitos similares. A diferença chave é que a primeira depende muito dos tamanhos e formas dos polígonos, produzindo aproximações não uniformes, ao passo que a segunda é mais consistente. Na Figura X3, B está mais próximo de C do que de A, quando se usa a distância de separação exata. Ainda que a distância entre os centróides de A e B seja próxima à distância de separação, a distância entre os centróides B e C é muitas vezes maior que a distância de separação entre B e C. De fato, pela distância entre centróides, B é mais próximo de A do que C, invertendo a ordem induzida pela distância de separação. Geralmente, se uma

coleção de objetos tem uma grande variedade de tamanhos e formas, a aproximação por distância entre centróides produz *clusterings* pobres. Por outro lado, pela Figura 25 pode-se ver que a aproximação MV preserva a relação "B é mais próximo de C do que de A".(Ng, 2002) produziu resultados experimentais que mostram que as aproximações MV geram melhores *clusterings* que a aproximação pela distância entre centróides.

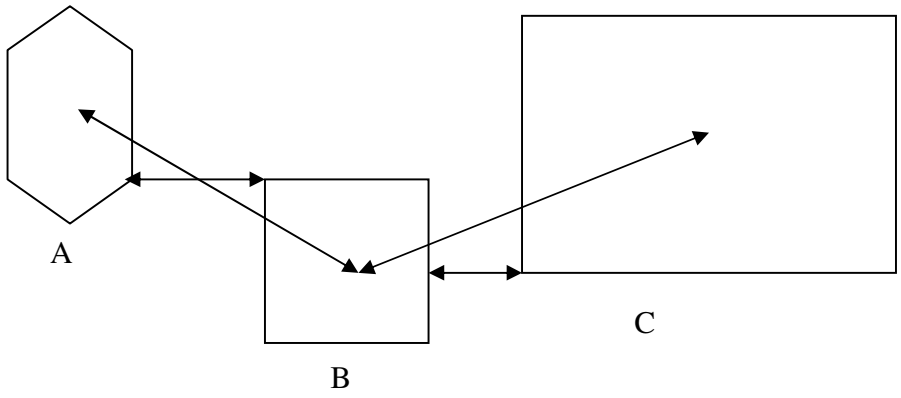


Figura 25- Comparação: distância entre centróides e aproximação MV.(Fonte: NG,2002)

Aproximação pela Distância de Separação Entre Retângulos Isotéticos.

Dado um polígono A, o retângulo isotético I_A do mesmo é o menor retângulo que contém A e cujas arestas são paralelas aos eixos x ou y. Define-se a *aproximação IR* entre dois polígonos A e B como sendo a distância de separação entre os retângulos isotéticos I_A e I_B. O cálculo dessa aproximação é simples e eficiente em termos algorítmicos (ordem de complexidade constante para retângulos isotéticos e logarítmica para polígonos) , podendo também ser usada para reduzir o tempo necessário para se obter a distância entre dois polígonos. A questão aqui também, é avaliar o quanto a aproximação IR afeta a qualidade do processo de *clustering* É trivial verificar-se que a aproximação IR subestima a distância entre polígonos, já que o retângulo isotético contém o polígono. Por esse motivo também, não exige que os polígonos sejam convexos. Resultados experimentais exibidos em (Ng, 2002) mostram que a aproximação IR produz *clusters* de qualidade comparável à obtida com distâncias de separação exata e têm um desempenho superior ao da aproximação MV. Portanto, é a aproximação mais adequada a ser utilizada no processo de *clustering*.

Method Summary

doub	**calcularvizinhancaespacial**(Instancia inst1, Instancia inst2) Retorna vizinhanca espacial entre duas instâncias
ConjuntoInstancias	**retornanovosclusteres**(ConjuntoInstancias[] clusters) Retorna novos clusteres após verificação de vizinhança.

Methods inherited from interface analisedados.tecnicasdatamining.dmclustering.ISBClustering

iniciar, LerInstancias, RetornaClassificacaoInstancia, RetornaContagemClusters, RetornaInstanciaPorCluster, RetornaInstancias, RetornaNumClusters, RetornaTamanhoClusters, SetNumClusters

Method Detail

calcularvizinhancaespacial

```
double calcularvizinhancaespacial(Instancia inst1,
                                  Instancia inst2)
```

Retorna vizinhanca espacial entre duas instâncias

Parameters:

`inst1` - primeira instância

`inst2` - segunda instância

Returns:

distância entre as instâncias

See Also:

---

retornanovosclusteres

```
ConjuntoInstancias[] retornanovosclusteres(ConjuntoInstancias[] clusters)
```

Retorna novos clusteres após verificação de vizinhança. Deve ser invocado após o retorno de todos os métodos de clustering convencional.

Parameters:

`clusters` - clusteres antes da verificação de vizinhança

Returns:

clusteres após verificação de vizinhança

See Also:

analisedados.tecnicasdatamining.dmclustering

Interface ISBCOBWEB

All Superinterfaces:
ISBBase, ISBClustering

```
public interface ISBCOBWEB
extends ISBClustering
```

Interface que especifica métodos do algoritmo COBWEB

A *Utilidade de Categoria*

(Fisher, 1987) define a chamada *utilidade de categoria (category utility-CU),* uma grandeza estatística que mede a qualidade média de uma partição de instâncias em *n clusters,* dada pela seguinte relação:

$$CU(\{C_1, C_{2,\ldots}C_n\}) = \frac{\sum_k P(C_k)\left\{\sum_i \sum_j (P(A_i = V_{ij} \mid C_k))^2 - \sum_i \sum_j (P(A_i = V_{ij}))^2\right\}}{n}$$

Onde:

- $\{C_1, C_2,\ldots C_n\}$ é um conjunto de *clusters*
- $\{A_1, A_2, \ldots A_j\}$ é um conjunto de atributos onde o *i*-ésimo atributo possui valores V_{i1}, $V_{i2},\ldots V_{ij}\ldots$
- $P(A_i = V_{ij} \mid C_k)$ é a probabilidade de um atributo A_i em um *cluster* C_k possuir um valor V_{ij} (Teorema de Bayes)
- $P(A_i = V_{ij})$ é a probabilidade de A_i possuir valor V_{ij} para qualquer membro do conjunto de dados escolhido aleatoriamente
- $P(C_k)$ é a probabilidade de uma determinada instância pertencer ao *cluster* C_k

A relação acima se aplica a atributos não numéricos. Para atributos numéricos, (Witten, 2005) deduz uma relação para cálculo da utilidade de categoria, integrando a função densidade de probabilidade da distribuição normal para um atributo A :

$$f(A) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(\frac{(A-\mu)^2}{2\sigma^2} \right)$$

É evidente que

$$\sum_i \sum_j (P(A_i = V_{ij}))^2 \Leftrightarrow \int f(A_i)^2 dA_i = \frac{1}{\sqrt{2\pi}\sigma_i}$$

Fazendo-se a mesma integração para $\sum_i \sum_j (P(A_i = V_{ij} \mid C_k))^2$ obtém-se a fórmula final:

$$CU(\{C_1, C_2, \ldots C_n\}) = \frac{\sum_k P(C_k) \frac{1}{2\sqrt{\pi}} \sum_i (\frac{1}{\sigma_{ki}} - \frac{1}{\sigma_i})}{n}$$

Onde σ_{ki} é o desvio padrão de cada atributo A_i dentro de um *cluster k* e σ_i é o desvio-padrão de A_i em todos os *clusters.*

***Clustering* Incremental (COBWEB)**

Os algoritmos vistos anteriormente trabalham sobre todo o conjunto de dados. O *clustering* incremental, ou COBWEB (Fisher, 1987), insere instâncias em uma árvore, que se inicia com um agrupamento inicial (raiz). Para cada instância, procura-se o melhor local na árvore onde a mesma pode ser inserida, criando-se ou combinando-se *clusters,* utilizando-se a utilidade de categoria como avaliador. O algoritmo COBWEB pode ser assim descrito:

Para cada instância a ser inserida:
1-Insira-a ao *cluster* que fornece o melhor particionamento, utilizando a utilidade de categoria como avaliador. Esse *cluster* é denominado *melhor hospedeiro.*
2-As seguintes operações deverão ser aplicadas se resultarem em melhor utilidade de categoria*:*

Juntar: unir o *cluster* melhor hospedeiro ao próximo melhor hospedeiro para formar uma classe melhor, quando uma nova instância estiver entre dois *clusters* jé existentes;

Partir: dividir o melhor hospedeiro em dois *clusters* , quando este possuir uma distribuição bi-modal entre seus membros;

Criar novo cluster: criar novo cluster para a instância, quando esta for muito diferente das já inseridas;

A vantagem notória desse algoritmo em relação aos anteriores é que se pode avaliar a qualidade do processo de *clustering* enquanto o mesmo é executado. Entretanto, a ordem em que as instâncias são inseridas produz resultados diferentes, o que exige várias tentativas para se obter o melhor resultado. Percebe-se também que, para situações em que o atributo é numérico e um nó da árvore contém somente uma instância, o desvio-padrão é obviamente zero, produzindo divisões por zero na fórmula da utilidade de categoria. Para que essa situação seja evitada, (Witten, 2005) sugere a imposição de um valor mínimo de variância para cada atributo, o qual representaria o erro mínimo de medida em uma única amostra. Assim, somente *clusters com* esse valor mínimo de variância poderiam ser criados.

Method Summary

flc	**RetornaUtilidadeCategoria**(ConjuntoInstancias[] cj) Retorna a utilidade de categoria de um conjunto de clusters

Methods inherited from interface analisedados.tecnicasdatamining.dmclustering.ISBClustering

iniciar, LerInstancias, RetornaClassificacaoInstancia, RetornaContagemClusters, RetornaInstanciaPorCluster, RetornaInstancias, RetornaNumClusters, RetornaTamanhoClusters, SetNumClusters

Method Detail

RetornaUtilidadeCategoria

`float` **`RetornaUtilidadeCategoria`**`(ConjuntoInstancias[] cj)`

Retorna a utilidade de categoria de um conjunto de clusters

Parameters:

`cj` - Conjunto de clusters

Returns:

Valor da utilidade de categoria

analisedados.tecnicasdatamining.dmclustering

Interface ISBDendograma

All Superinterfaces:
ISBBase, ISBClustering

```
public interface ISBDendograma
extends ISBClustering
```

Calcula o dendograma de um conjunto de instâncias

Métodos hierárquicos de *clustering* permitem obter vários níveis de agrupamento. Segundo (Câmara,2001), eles podem ser classificados em dois tipos:

- *aglomerativos* : em que cada objeto é inicialmente um *cluster,* fundindo-se sucessivamente dois *clusters* mais próximos, até chegar-se a um agrupamento máximo ao final. As fusões entre os *clusters* podem ser exibidas em uma árvore, o *dendrograma* (Figura 23).

Os passos do algoritmo hierárquico aglomerativo são (Câmara, 2001 e Richardson, 1995):

1: Iniciar com *n* clusters, cada um contendo um objeto.

2: Calcular as dissimilaridades entre os objetos.

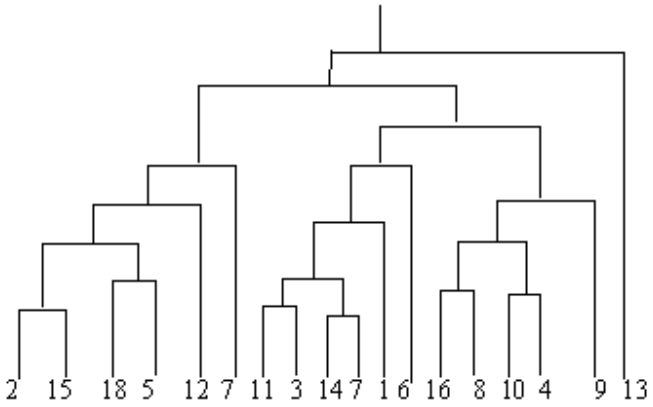
3: Procurar o par de clusters com menor dissimilaridade;

4: Recalcular a dissimilaridade do *cluster* fundido com os demais clusters.

5: Repetir os passos 3 e 4 *n-1* vezes.

- *divisivos*.: inicialmente todos os objetos pertencem a um único grupo, que vai sendo sucessivamente dividido, até que cada *cluster* contenha um único elemento.

O grande inconveniente dos métodos hierárquicos é que eles não proporcionam uma possível modificação dos *clusters* durante o procedimento, de forma a incrementar a qualidade do agrupamento(NG, 1994).



Exemplo de um dendograma

See Also:

Method Summary

Dendogra	**retornaDendograma**(ConjuntoInstancias Cisnt) Retorna o dendograma de um conjunto de instâncias

Methods inherited from interface analisedados.tecnicasdatamining.dmclustering.ISBClustering

iniciar, LerInstancias, RetornaClassificacaoInstancia, RetornaContagemClusters, RetornaInstanciaPorCluster, RetornaInstancias, RetornaNumClusters, RetornaTamanhoClusters, SetNumClusters

Method Detail

retornaDendograma

Dendograma **retornaDendograma**(ConjuntoInstancias Cisnt)

Retorna o dendograma de um conjunto de instâncias

Parameters:

Cinst - Conjunto de Instâncias para o qual se deseja o dendograma

Returns:

O dendograma

analisedados.tecnicasdatamining.dmclustering

Interface ISBEM

All Superinterfaces:
ISBBase, ISBClustering

```
public interface ISBEM
extends ISBClustering
```

Interface que especifica métodos do algoritmo EM

Abordagens estatísticas constituem uma forma mais fundamentada para obtenção de *clusters,* em relação aos métodos anteriormente vistos. Em essência, elas consideram a probabilidade de uma instância pertencer a um *cluster.* A base dessas abordagens são modelos estatísticos denominados *misturas finitas.* Cada *cluster* pode ser matematicamente representado por uma distribuição de probabilidades, como a normal (gaussiana) ou a de Poisson. O conjunto de dados é então modelado como uma *mistura destas distribuições*, que governam os valores de atributos nos *clusters.* Ou seja, cada distribuição fornece a probabilidade de que uma instância tenha um certo conjunto de valores de atributo, se for conhecido que ela seja um membro de um *cluster.* O problema de *clustering* consiste então, em alocar instâncias a um certo número de *clusters,* obtendo-se os parâmetros das distribuições de probabilidades que formam os melhores *clusters.*

O algoritmo EM (*expectation-maximization*) proposto por (Dempster, 1977) é uma forma iterativa de se obter os parâmetros da distribuição de probabilidades, a partir do cálculo dos valores esperados das probabilidades de cada *cluster* (*expectation)* e uma posterior obtenção dos valores dos parâmetros da distribuição de probabilidades que maximizam (*maximization)* a chamada *função de verossimilhança,* definida da seguinte forma:

Dados um conjunto de dados independentes $X = \{x_1, x_2, ..., x_n\}$, com função densidade de probabilidade $f(x, \theta)$, onde θ é um vetor de parâmetros da mesma, a função de verossimilhança é

$$L(X, \theta) = \prod_{i=1}^{n} f(x_i, \theta)$$

Como transformações monótonas não alteram a maximização, aplicar o logaritmo à expressão acima torna-a analiticamente mais simples, transformando o produtório em somatório e obtendo –se:

$$L^*(X,\theta) = \sum_{i=1}^{n} \log(f(x_i,\theta))$$

O algoritmo EM pode ser assim expresso:

1-Definir o número n de *clusters* desejado

2-Definir valores iniciais para os parâmetros θ da distribuição (se for a gaussiana, vetores de média, desvio-padrão e probabilidade de pertinência a cada *.cluster*);

3-Repetir até que a diferença entre duas iterações, do valor da função de verossimilhança, for menor que um mínimo ε :

Passo E (*Expectation):* calcular as probabilidades de cada *cluster* a partir dos valores atuais dos parâmetros θ das distribuições. Esta probabilidade é a seguinte(Teorema de Bayes):

Para cada *cluster k temos* $P(C_k \mid x_i,\theta) = \dfrac{P(x_i \mid C_k,\theta)P(C_k)}{P(x_i)}$, que após normalizada torna-se

$P(C_k \mid x_i,\theta) = P(x_i \mid C_k,\theta)P(C_k)$

Passo M (*Maximization):* utilizar as probabilidades obtidas no passo E para encontrar novos valores de parâmetros θ da distribuição de forma a maximizar a função de verossimilhança (demonstra-se que esse valor aumenta a cada iteração e que graças a isso o algoritmo converge). A função de verossimilhança é $L^*(X,\theta) = \sum_{i=1}^{n} \log(f(x_i,\theta))$. Mas $f(x_i,\theta)$ é dada por $f(x_i,\theta) = \sum_{k=1}^{n} P(x_i \mid C_k,\theta)P(C_k)$.

Portanto

$$L^*(X,\theta) = \sum_{i=1}^{n} \log(\sum_{k=1}^{n} P(x_i \mid C_k,\theta)P(C_k))$$

Derivando-se parcialmente *L** em relação a cada parâmetro θ e igualando-se o resultado a zero, podem-se resolver as equações resultantes para obter os novos valores dos parâmetros θ a serem usados na próxima iteração. Se existirem *j* parâmetros a serem estimados, então:

$$\frac{\partial L^*(X,\theta_j)}{\partial \theta_j} = 0$$

Method Summary

double	**RetornaLogDensidadePorCluster**(Instancia inst) Retorna o logaritmo da função densidade de probabilidade por cluster, da uma instancia
vo	**setParametrosGaussiana**(float media, float sdev, float probab) Retorna o logaritmo da função densidade de probabilidade por cluster, da uma instancia

Methods inherited from interface analisedados.tecnicasdatamining.dmclustering.ISBClustering

iniciar, LerInstancias, RetornaClassificacaoInstancia, RetornaContagemClusters, RetornaInstanciaPorCluster, RetornaInstancias, RetornaNumClusters, RetornaTamanhoClusters, SetNumClusters

Method Detail

RetornaLogDensidadePorCluster

```
double[] RetornaLogDensidadePorCluster(Instancia inst)
```

Retorna o logaritmo da função densidade de probabilidade por cluster, dada uma instancia

Parameters:

`inst` - Instância para a qual se deseja o cálculo

Returns:

Logaritmo da função densidade de probabilidade por cluster

---

setParametrosGaussiana

```
void setParametrosGaussiana(float media,
                            float sdev,
                            float probab)
```

Retorna o logaritmo da função densidade de probabilidade por cluster, dada uma instancia

Parameters:

`media` - Média

`sdev` - Desvio-padrão

`probab` - Probabilidade de pertinência a cada cluster

estracaodados

Interface ISBExtrairCSV

All Superinterfaces:

ExtracaoBase

```
public interface ISBExtrairCSV
extends ExtracaoBase
```

Interface para extração de dados de arquivos CSV

Method Summary

Methods inherited from interface estracaodados.ExtracaoBase
Extrair

estracaodados

Interface ISBExtrairJDBC

```
public interface ISBExtrairJDBC
```

Interface para extração de dados de arquivos JDBC

Method Summary

vc	**iniciarbanco**(java.lang.String caminhobanco, java.lang.String stringbanco, java.lang.String usuario, java.lang.String senha) Inicializa dados do banco

Method Detail

iniciarbanco

```
void iniciarbanco(java.lang.String caminhobanco,
                  java.lang.String stringbanco,
                  java.lang.String usuario,
                  java.lang.String senha)
```

Inicializa dados do banco

Parameters:

`caminhobanco` - Caminho do banco (por exemplo:jdbc:mysql://localhost/wayfrota)

`stringdobanco` - String característica do banco (como org.gjt.mm.mysql.Driver, do MySQL)

`usuario` - Nome de usuário do banco

`senha` - Senha de acesso ao banco

estracaodados

Interface ISBExtrairMIF

All Superinterfaces:

ExtracaoGeoBase

```
public interface ISBExtrairMIF
extends ExtracaoGeoBase
```

Interface para extração de dados geográficos e convencionais de arquivos MIF

Method Summary

Methods inherited from interface estracaodados.ExtracaoGeoBase
geraarquivosql, iniciar, retornadados, retornageometrias

estracaodados

Interface ISBExtrairSHP

All Superinterfaces:
ExtracaoGeoBase

```
public interface ISBExtrairSHP
extends ExtracaoGeoBase
```

Interface para extração de dados geográficos e convencionais de arquivos SHP

Method Summary

Methods inherited from interface estracaodados.ExtracaoGeoBase
geraarquivosql, iniciar, retornadados, retornageometrias

analisedados.tecnicasdeia.geneticos

Interface ISBGeneticos

`public interface` **`ISBGeneticos`**

Interface que especifica métodos que implementam algoritmos genéticos

Os algoritmos genéticos (AG) são uma família de algoritmos computacionais inspirados na teoria da evolução, que incorporam conceitos semelhantes aos de cromossomo, seleção, reprodução e mutação para resolver, principalmente, problemas de otimização.
O conjunto de estados possíveis de um problema é denominado *população.* Cada estado constitui um indivíduo e é codificado como um *cromossomo.* Usualmente, os cromossomos são representados por vetores binários cujos elementos indicam a presença (1) ou ausência (0) de uma característica. Entretanto, representações por meio de vetores inteiros, reais ou de caracteres, também são possíveis. A combinação dessas características determina o aspecto (fenótipo) final do indivíduo.
Os indivíduos se reproduzem, gerando novas populações. Nessas populações os indivíduos são selecionados de forma que os mais aptos têm mais chance de se reproduzir, transmitindo sua herança genética às populações seguintes.. Para quantificar essa aptidão, uma *função de avaliação* é utilizada, de forma que retorne valores maiores para os melhores indivíduos. Essa função depende do problema abordado. No processo de reprodução, tal como no mundo real, aplicam-se aos cromossomos operações de *crossing-over* e mutação, que vão gerar os novos indivíduos.

Seleção

A seleção é feita de forma a privilegiarem-se indivíduos mais aptos (maiores valores da função de avaliação). Um dos métodos mais populares é o denominado *método da roleta,* em que cada cromossomo, em uma roleta, possui uma área proporcional a sua aptidão. O "ponteiro" da roleta é impulsionado e a área apontada pelo mesmo ao parar, corresponde a um indivíduo selecionado. Assim, a área a_i de um indivíduo i que possui uma determinada $nota_i$ proporcional a sua aptidão é dada por:

$$a_i = \frac{nota_i}{\sum_{j=1}^{N} nota_p}$$

Onde N é o número de indivíduos a selecionar (número de vezes em que o "ponteiro" é impulsionado).

Contudo, se estas áreas forem muito próximas, pode não haver favorecimento dos mais aptos. (Pádua, 2000), sugere que a fatia seja determinada pela posição que o cromossomo ocupa no *ranking* de todas os notas. A fatia correspondente a cada cromossomo é definida pela escolha de um valor entre 0,0 e 1,0. Cada indivíduo, em ordem crescente do *ranking,* ocupa, da área total da roleta ainda não ocupada, uma fatia proporcional ao valor escolhido. O indivíduo de menor *ranking* ocupará a área que restou na roleta.

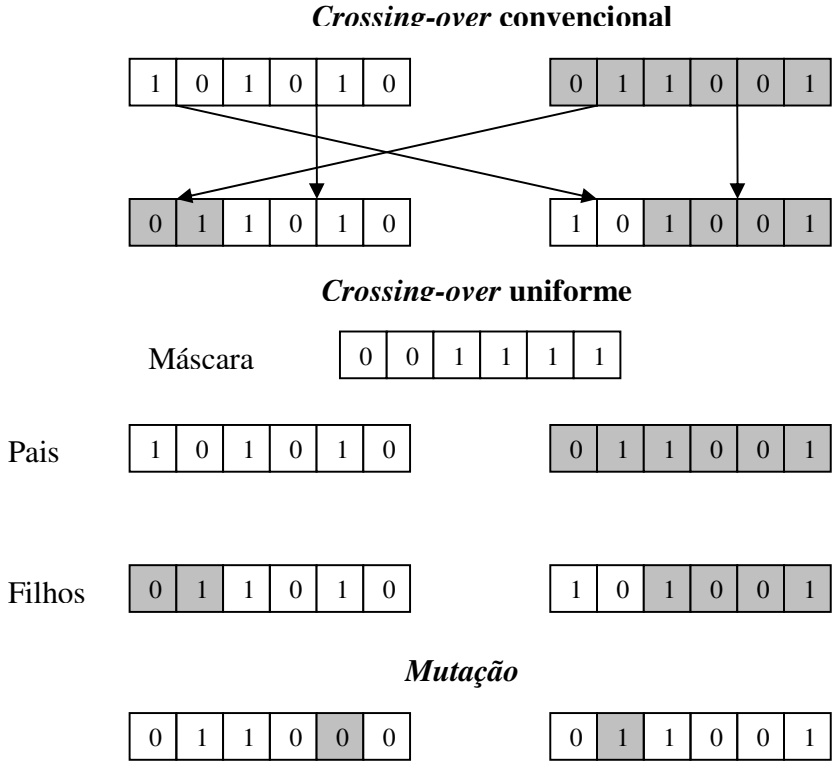
De forma a preservarem-se os melhores indivíduos, pode-se passá-los, sem cruzamento e alteração, para a geração seguinte. Este procedimento, denominado *elitismo*, melhora o desempenho do algoritmo, pois evita que tais indivíduos sejam destruídos no cruzamento ou na ocorrência de mutação. A % dos indivíduos restantes que se cruzam é denominada *taxa de cruzamento.* Em geral essa elite é constituída por aproximadamente 2% da população. O restante da população efetua o cruzamento. Entretanto, em determinadas situações, o elitismo pode gerar populações muito homogêneas, com bons indivíduos, mas com chances menores de produzir indivíduos ainda melhores.

Reprodução

O processo de reprodução combina cromossomos entre si, de forma a produzir indivíduos diferentes na próxima geração, para consequentemente aumentar a possibilidade de aparecimento de melhores variações. Esse processo ocorre por meio de dois operadores sucessivos:

- Cruzamento. É o denominado *crossing-over,* em que dois cromossomos fornecem aos filhos partes extraídas a partir de pontos de corte escolhidos aleatoriamente. Pode haver um único, ou múltiplos cortes. Uma outra forma de *crossing-over* utiliza uma máscara (vetor) de

bits, com o mesmo tamanho dos cromossomos, para determinar que genes serão herdados do pai e da mãe. Nessa máscara, se o í-ésimo elemento for 1, o í-ésimo gene do filho 1 será igual ao do pai, se for 0, será igual ao da mãe. O inverso ocorre no filho 2. A figura abaixo exibe esquematicamente esses processos.



Operações de *crossing-over* e mutação (Inspirado em (Pádua, 2000))

- Mutação. Consiste em uma mudança aleatória na cadeia de genes dos filhos, que ocorre após a operação de *crossing-over.* É a responsável pela variabilidade genética na população e ocorre em uma freqüência determinada pela *taxa de mutação.* A Figura X10 exibe o funcionamento desse operador.

O algoritmo genético

Criar um conjunto Inicial de cromossomos;
Repetir
Utilizar uma função de avaliação para atribuir uma nota a cada cromossomo;
    Selecionar os cromossomos mais aptos;
    Se for realizar elitismo
De acordo com a taxa de cruzamento, retirar os mais aptos para passar diretamente à geração seguinte;
Sobre os cromossomos restantes:
    Aplicar *crossing-over;*
    Aplicar mutação considerando-se a taxa de mutação;
    Senão
Sobre os cromossomos mais aptos:
    Aplicar *crossing-over;*
    Aplicar mutação considerando-se a taxa de mutação;
Até que sejam obtidos cromossomos adequados, ou um certo número de mutações.

Desempenho do algoritmo

A escolha de parâmetros do algoritmo pode influenciar no desempenho e no comportamento do mesmo e devem ser determinados, geralmente por tentativa e erro, em função das características e peculiaridades do problema em questão. Segundo (Jong, 1980) e (Pádua, 2000) os parâmetros mais importantes a serem considerados são os seguintes:

- *Tamanho da população:* Pequenas populações comprometem o desempenho do algoritmo por representarem um espaço de busca pequeno. Grandes populações diminuem a chance de ocorrência de mínimos locais, mas devem ser usadas com parcimônia, por demandarem muito esforço computacional.
- *Taxa de cruzamento:* Um valor maior acarreta maior variabilidade genética. Entretanto, se for muito alta, aumenta-se a chance de eliminação de bons indivíduos. Se for muito baixa, a população tende a estagnar, diminuindo-se a chance de produção de bons indivíduos.

- *Taxa de mutação:* Uma baixa taxa de mutação previne estagnações de determinadas posições. Quando muito alta, introduz-se uma aleatoriedade excessiva no processo de busca.
- *Intervalo de geração:* Controla a porcentagem da população a ser substituída na próxima geração. Um valor alto aumenta a chance de perda de bons indivíduos. Um valor muito baixo torna o algoritmo lento, pois pode ser necessário um grande número de gerações até convergir.

Method Summary

Cromossc	**retornarmelhorsolucao**() Retorna o melhor indivíduo após o treinamento
Genc	**treinar**(Genoma genoma, int numite, double taxamutacao, int numelite) Efetua treinamento sobre uma população inicial

Method Detail

treinar

```
Genoma treinar(Genoma genoma,
               int numite,
               double taxamutacao,
               int numelite)
```

Efetua treinamento sobre uma população inicial

Parameters:

`genoma` - População inicial

`numite` - Número de iterações

`taxamutacao` - Taxa de mutação

`numelite` - Número de indivíduos de elite

Returns:

População selecionada

---

retornarmelhorsolucao

```
Cromossomo retornarmelhorsolucao()
```

Retorna o melhor indivíduo após o treinamento

Returns:

Melhor indivíduo

analisedados.modelosanaliseespacial

Interface ISBHagerstrand

```
public interface ISBHagerstrand
```

Interface que especifica métodos que implementam o modelo de Hagerstrand.

A dimensão espacial é a essência da representação da informação geográfica. Esta essência é válida quando considerados fenômenos e eventos estáticos, ao menos em uma escala de tempo macroscópica. Entretanto, quando se consideram as atividades humanas, a dimensão tempo torna-se relevante.

O geógrafo Törsten Hägerstrand, na década de 60, notabilizou-se por integrar o tempo ao comportamento espacial como uma dimensão intrínseca, através de um diagrama tempo-espacial. Entretanto, segundo (Adams, 1996), Hägerstrand restringiu sua visão das atividades humanas ao movimento do corpo, limitando suas interações espaciais aos objetos ao seu alcance, ignorando a natureza social do ser humano, capaz de estender tais interações a escalas muito maiores.

Extensões ao diagrama tempo-espacial de Hägerstrand têm sido propostas. (Adams, 1996) com seu diagrama de extensibilidade, representa em uma escala temporal as atividades realizadas por um indivíduo e a escala de transação de tais atividades nos níveis próximo, metripolitano, estadual, regional, nacional e internacional.. (Kwan, 1995) criou os chamados caminhos espaço-tempo individuais em múltiplas escalas, uma variação do trabalho de (Adams, 1996).

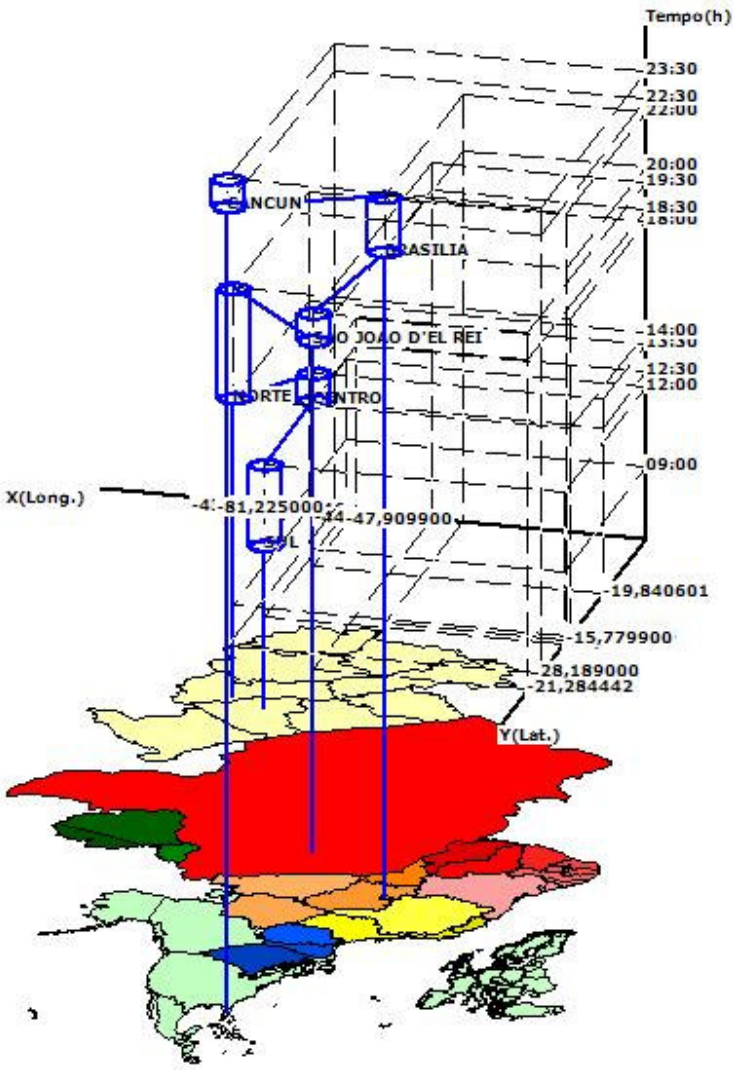
O diagrama tempo-espacial multi-escopo

..

O diagrama tempo-espacial multi-escopo representa as transações de um indivíduo em termos das dimensões tempo e espaço em até 4 escopos de interação, cada um deles a uma escala espacial distinta:

- Escopo local (próximo)
- Escopo estadual
- Escopo nacional
- Escopo internacional

Três eixos de coordenadas são utilizados para representar a informação, um para tempo e os outros dois para latitude e longitude. Cada escopo é representado em uma camada diferente do diagrama, de forma que todas as transações podem ser simultaneamente visualizadas. A Figura 45 apresenta o aspecto geral do diagrama. Os cilindros em azul representam os tempos gastos em cada atividade, ou seja, a base do cilindro corresponde ao início da atividade e o topo ao fim (que podem ser lidos no eixo vertical). As linhas que partem da base dos cilindros para os mapas nas camadas indicam o local exato onde ocorreu a transação. As linhas azuis conectando os cilindros representam a seqüência em que as atividades ocorreram. Neste exemplo, as camadas representam os mapas de Belo Horizonte, do Estado de Minas Gerais, do Brasil e das Américas do Norte/Central e Europa, correspondentes ao escopo das transações que o indivíduo realizou.



Aspecto Geral do Diagrama

Method Summary	
vc	**armazenaestacao**(Estacoes estacao) Armazena uma estacao no modelo a ser criado
ConjuntoEstacc	**retornacamada**(java.lang.String nomecamada) Retorna uma camada com suas estações
Poi	**retornacoordenada**(java.lang.String nomestacao)

	Retorna a coordenada de uma estação
ConjuntoEstacc	**retornaestacoes**() Retorna todas as estações do modelo

Method Detail

retornacoordenada
Point **retornacoordenada**(java.lang.String nomestacao)
Retorna a coordenada de uma estação
Parameters:
nomeestacao - Nome da estação
Returns:
Coordenada da estação

---

armazenaestacao
void **armazenaestacao**(Estacoes estacao)
Armazena uma estacao no modelo a ser criado
Parameters:
estacao - Estação a ser armazenada

---

retornaestacoes
ConjuntoEstacoes **retornaestacoes**()
Retorna todas as estações do modelo
Returns:
Conjunto de estações

---

retornacamada
ConjuntoEstacoes **retornacamada**(java.lang.String nomecamada)
Retorna uma camada com suas estações
Parameters:
nomecamada - Nome da camada
Returns:
Conjunto de Estações

analisedados.tecnicasdatamining.dmclustering

Interface ISBKMeans

All Superinterfaces:
ISBBase, ISBClustering
All Known Subinterfaces:
ISBkmeansEspacial

```
public interface ISBKMeans
extends ISBClustering
```

Interface que especifica métodos do algoritmo KMeans

É considerado o algoritmo clássico de *clustering*. Os passos são os seguintes (Witten, 2005):

1-Definir o número k de *clusters* desejado

2-Definir aleatoriamente k pontos, que serão os centros dos *clusters*.

3-Associar cada ponto ao centro do *cluster* mais próximo, utilizando como métrica a distância euclidiana.

4-Calcular a média (centróide) dos pontos em cada *cluster*

5-Tornar estes centróides novos centros de *cluster*.

6-Repetir os passos 3,4 e 5 até que haja estabilidade, ou seja, quando em iterações consecutivas os centróides não se alterarem mais.

Esse método é simples e bastante efetivo, já que a alocação dos centros de *clusters* ao centróide minimiza a distância quadrática média entre cada ponto do *cluster* e seu centro. Entretanto, este mínimo é local, sem garantia de globalidade. Assim sendo, resultados diferentes obtêm-se com seleções distintas de centros de *cluster* inicial. Portanto, deve-se realizar diversas rodadas do algoritmo, com centros diferentes, selecionando-se ao final a configuração que possua a menor distância quadrática média (Witten, 2005).

Method Summary

ConjuntoInstanci	**RetornaCentroideCluster**(int cluster) Retorna o centróide de um cluster

Methods inherited from interface analisedados.tecnicasdatamining.dmclustering.ISBClustering

iniciar, LerInstancias, RetornaClassificacaoInstancia, RetornaContagemClusters, RetornaInstanciaPorCluster, RetornaInstancias, RetornaNumClusters, RetornaTamanhoClusters, SetNumClusters

Method Detail

RetornaCentroideCluster

ConjuntoInstancias **RetornaCentroideCluster**(int cluster)

Retorna o centróide de um cluster

Parameters:

cluster - índice do cluster cujo centróide deseja-se obter

Returns:

O centróide do cluster

analisedados.tecnicasdatamining.dmclusteringespacial

Interface ISBkmeansEspacial

All Superinterfaces:

ISBBase, ISBClustering, ISBClusteringEspacial, ISBKMeans

```
public interface ISBkmeansEspacial
extends ISBClusteringEspacial, ISBKMeans
```

Versão espacial da interface que especifica métodos do algoritmo KMeans

É considerado o algoritmo clássico de *clustering*. Os passos são os seguintes (Witten, 2005):

1-Definir o número k de *clusters* desejado

2-Definir aleatoriamente k pontos, que serão os centros dos *clusters*.

3-Associar cada ponto ao centro do *cluster* mais próximo, utilizando como métrica a distância euclidiana.

4-Calcular a média (centróide) dos pontos em cada *cluster*

5-Tornar estes centróides novos centros de *cluster*.

6-Repetir os passos 3,4 e 5 até que haja estabilidade, ou seja, quando em iterações consecutivas os centróides não se alterarem mais.

Esse método é simples e bastante efetivo, já que a alocação dos centros de *clusters* ao centróide minimiza a distância quadrática média entre cada ponto do *cluster* e seu centro. Entretanto, este mínimo é local, sem garantia de globalidade. Assim sendo, resultados diferentes obtêm-se com seleções distintas de centros de *cluster* inicial. Portanto, deve-se realizar diversas rodadas do algoritmo, com centros diferentes, selecionando-se ao final a configuração que possua a menor distância quadrática média (Witten, 2005).

Method Summary

Methods inherited from interface analisedados.tecnicasdatamining.dmclusteringespacial.ISBClusteringEspacial
calcularvizinhancaespacial, retornanovosclusteres

Methods inherited from interface analisedados.tecnicasdatamining.dmclustering.ISBKMeans
RetornaCentroideCluster

Methods inherited from interface analisedados.tecnicasdatamining.dmclustering.ISBClustering
iniciar, LerInstancias, RetornaClassificacaoInstancia, RetornaContagemClusters, RetornaInstanciaPorCluster, RetornaInstancias, RetornaNumClusters, RetornaTamanhoClusters, SetNumClusters

Interface ISBLISA

```
public interface ISBLISA
```

Interface que especifica métodos que implementam autocorrelação

Autocorrelação

A autocorrelação expressa quanto o valor observado de um atributo numa região é dependente dos valores deste mesmo atributo nas localizações vizinhas. O índice de Moram I é o mais utilizado para estimar a força da correlação entre valores observados em função da distância que os separam. Este índice é dado pela expressão:

$$I = \frac{\sum_{i=1}^{n}\sum_{j=1}^{n} w_{ij}(z_i - \bar{z})(z_j - \bar{z})}{\sum_{i=1}^{n}(z_i - \bar{z})^2} \quad (17)$$

onde

z_i = o valor normalizado do atributo considerado na área i

$\bar{z}$ = é o valor médio do atributo na região sob estudo.

n = número de áreas

w_{ij} = peso da área i em relação à área j. Se forem vizinhas, seu valor pode ser 1, caso contrário, 0. Este peso pode ser também a distância entre as áreas.

Valores positivos (entre 0 e +1) do índice de Moran indicam uma correlação direta, valores negativos (entre -1 e 0) indicam correlação inversa.

Outro importante indicador de autocorrelação é o LISA (*Local Indicator of Spatial Associatin)* (Anselin, 1995), que é o equivalente local do índice de Moran (a soma de todos os índices locais de uma região é proporcional ao índice de Moran). Ele é obtido pela seguinte expressão:

$$I_i = z_i \sum_j^n w_{ij} z_j \qquad (18)$$

onde I_i é o índice LISA de correlação espacial local para cada área i na região considerada.

Para cada área, os índices LISA permitem avaliar sua similaridade em relação às áreas vizinhas e testar sua significância. Desta análise, quatro cenários podem emergir:

- Áreas com valores altos e vizinhos similares: *high-high*. São denominadas *hot spots.*
- Áreas com valores baixos e vizinhos similares: *low-low*. São denominadas *cold spots.*
- Áreas com valores altos e vizinhos com valores baixos: *high-low*. São denominadas *potential spatial outliers.*
- Áreas com valores baixos e vizinhos com valores altos: *low-high*. São também denominadas *potential spatial outliers.*
- Áreas com autocorrelação não significativa .

Para estimar a significância do índice LISA, utiliza-se o chamado *teste de pseud-significância:* para cada área, permutam-se os valores dos atributos nas diversas regiões ao seu redor e calcula-se um novo índice *I* para cada permutação. Obtém-se assim uma nova distribuição empírica do índice LISA para cada área da região. O valor real de cada índice é testado em relação à distribuição obtida pelas diversas situações, através do chamado *p-value,* que é definido como "a probabilidade de qualquer média da amostra ser mais extrema que a média da amostra $\overline{I}$ extraída para o teste". O *p-value* é calculado da seguinte forma:

Para cada área i tem-se I_i e a média $\overline{I}$ dos valores permutados obtém-se o *p-value* da seguinte forma:

Se $\overline{I} >= I_i$ então p-value = $2\ x\ P(Z >= Z_o)$

e $\overline{I} <= I_i$ então p-value = $2\ x\ P(Z <= Z_o)$

Onde P é a distribuição normal de probabilidades e $Z_o = (I_i - \overline{I})/\sigma$ (valor normalizado da variável).

Method Summary

double	**calcularindicemoranglobal**(ConjuntoInstanciasLISA[] cluster Retorna índice de MORAN global
double[]	**calcularindicesLISA**(ConjuntoInstanciasLISA[] clusters) Retorna índices LISA para cada cluster
ConjuntoInstanciasLISA	**retornaclusterslisa**(ConjuntoInstancias instancias) Retorna conjunto de clusters LISA

Method Detail

retornaclusterslisa
ConjuntoInstanciasLISA[]
retornaclusterslisa(ConjuntoInstancias instancias)
Retorna conjunto de clusters LISA
Parameters:
instancias - Conjunto de instâncias
Returns:
Clusters LISA
See Also:

---

calcularindicemoranglobal
double **calcularindicemoranglobal**(ConjuntoInstanciasLISA[] clusters)
Retorna índice de MORAN global
Parameters:
clusters - Clusteres LISA
Returns:
Índice de Moran
See Also:

---

calcularindicesLISA
double[] **calcularindicesLISA**(ConjuntoInstanciasLISA[] clusters)
Retorna índices LISA para cada cluster

Parameters:
`clusters` - Clusteres LISA
Returns:
Índices LISA para cada cluster

analisedados.tecnicasdeia.neurais

Interface ISBMLP

```
public interface ISBMLP
```

Interface que especifica métodos que implementam redes neurais MLP

(Texto teórico está em 4.4.1)

Method Summary

double	**consultar**(RedeMLP redemlp, double[] teste) Efetua a consulta em uma rede MLP
vo	**salvarrede**(RedeMLP redemlp, java.lang.String arquivo) Salva uma rede MLP em um arquivo
RedeM	**trazerrede**(java.lang.String arquivo) Carrega uma rede MLP armazenada em um arquivo
doub	**treinar**(RedeMLP redemlp, int epocas, double rmseminimo) Efetua o treinamento de uma rede MLP

Method Detail

treinar

```
double treinar(RedeMLP redemlp,
               int epocas,
               double rmseminimo)
```

Efetua o treinamento de uma rede MLP

Parameters:

`redemlp` - Rede MLP devidamente estruturada

`epocas` - Número de iterações

`rmseminimo` - Erro quadrático médio mínimo

Returns:

O Erro quadrático médio

---

consultar

```
double[] consultar(RedeMLP redemlp,
                   double[] teste)
```

Efetua a consulta em uma rede MLP

Parameters:

`redemlp` - Rede MLP devidamente estruturada

`teste` - Entrada submetida à rede

Returns:

Resposta da rede

---

trazerrede

```
RedeMLP trazerrede(java.lang.String arquivo)
```

Carrega uma rede MLP armazenada em um arquivo

Parameters:
`arquivo` - Nome do arquivo onde foi armazenada a rede MLP
Returns:
A rede MLP

---

salvarrede

```
void salvarrede(RedeMLP redemlp,
                java.lang.String arquivo)
```

Salva uma rede MLP em um arquivo
Parameters:
`arquivo` - Nome do arquivo onde foi armazenada a rede MLP
`redemlp` - Rede MLP a ser salva

analisedados.tecnicasdatamining.dmclustering

Interface ISBPAM

All Superinterfaces:
ISBBase, ISBClustering
All Known Subinterfaces:
ISBCLARA, ISBCLARANS

```
public interface ISBPAM
extends ISBClustering
```

Especifica interfaces de operações do algoritmo PAM

O algoritmo PAM (*Partitioning Around Medoids)* foi desenvolvido por Kaufmann e Rousseeuw (Kaufmann, 1990). Para encontrar k *clusters,* esse algoritmo utiliza um objeto representativo de cada *cluster,* o denominado *medoid,* que é o objeto mais central possível desses *clusters.*

Esse algoritmo pode ser considerado como um problema de classificação a partir da busca por uma partição ótima do conjunto de dados (Câmara, 2001). Do ponto de vista da otimização, temos que**:**

$$V = \{1,2,3,\ldots,n\},$$

$$\pi = \{C_1, C_2, \ldots, C_k\},$$

$$f(\pi),$$

onde: V é o conjunto de n objetos a serem particionados em classes, C_K é o *cluster K*, π é uma dada partição em k *clusters* e f (π) é uma função objetivo que fornece uma medida de qualidade da partição. O objetivo final é conseguir uma partição que maximize (ou minimize, dependendo do caso) a função objetivo. Um exemplo simples para uma função objetivo é dado por:

$$f(\pi) = \sum_{i=1}^{k} D_i \quad (1)$$

onde Di é a soma das distâncias euclidianas entre cada *cluster* i e o *medoid* correspondente Assim sendo, os passos do algoritmo PAM são os seguintes (Câmara, 2001 & NG, 1994):

1-Selecionar arbitrariamente k *medoids* dentre os objetos

2-Agrupar os objetos em função do *medoid* mais similar, definindo a partição corrente, $\pi_{corrente}$, e calcular f($\pi_{corrente}$)

3-Verificar todas as trocas possíveis entre um *medoid* e um objeto não selecionado, escolhendo um novo conjunto de k *medoids,* que produza uma partição π, tal que o valor para a $f(\pi)$ é o menor dentre todas as possibilidades.

4- Se o $f(\pi_{corrente}) < f(\pi)$, efetivar a troca, $\pi = \pi_{corrente}$ e voltar ao *passo 2*.

5- Senão, fim do procedimento.

(Ng, 1994) afirma que as vantagens dos métodos baseados em *k-medoids* em relação aos métodos baseados em centros médios são a maior robustez à presença de *outliers* (objetos fora do padrão dos demais objetos) e a independência da ordem em que os objetos são verificados. (Ng, 1994) afirma também que o PAM é mais adequado a pequenos conjuntos de dados, afirmação que pode ser comprovada pelo alta ordem de complexidade desse algoritmo: $O(k(n-k)^2)$ por iteração.

Method Summary

ConjuntoInstanci	**RetornaMedoidCluster**(int cluster) Retorna medoid de um cluster

Methods inherited from interface analisedados.tecnicasdatamining.dmclustering.ISBClustering

iniciar, LerInstancias, RetornaClassificacaoInstancia, RetornaContagemClusters, RetornaInstanciaPorCluster, RetornaInstancias, RetornaNumClusters, RetornaTamanhoClusters, SetNumClusters

Method Detail

RetornaMedoidCluster

`ConjuntoInstancias` **`RetornaMedoidCluster`**`(int cluster)`

Retorna medoid de um cluster

Parameters:

`cluster` - índice do cluste

Returns:

Medoid do cluster

See Also:

analisedados.tecnicasdatamining.dmregraassociacao

Interface ISBRegraAssociacao

All Known Subinterfaces:

ISBRegraAssociacaoEspacial

`public interface` **`ISBRegraAssociacao`**

Interface que especifica métodos para obtenção de regras de associação

Regras de Associação

Regras de associação descrevem relacionamentos lógicos do tipo causa-e-efeito entre itens em um conjunto de dados. Trata-se de uma ferramenta bastante útil no processo de descoberta de conhecimento. Uma regra de associação pode ser definida da seguinte forma (Agrawal, 1994): seja $I = \{i_1, i_2, ..., i_n\}$ um conjunto de *itens.* Seja D um conjunto de transações onde cada transação T é um conjunto de itens tal que $T \subseteq I$. Cada transação possui um identificador único denominado *TID.* Seja X um conjunto de itens tal que $X \subseteq T$. Uma regra de associação é uma implicação da forma $X \rightarrow Y$ onde $X \subset I$, $Y \subset I$ e $X \cap Y = 0$. A regra $X \rightarrow Y$ possui uma *confiança c* no conjunto de transações D se $c\%$ das transações em D que contém X também contêm Y, ou seja, é a proporção instâncias corretamente preditas sobre instâncias analisadas (Witten, 2005). A regra $X \rightarrow Y$ possui *suporte s* no conjunto de transações D se $s\%$ das transações em D contêm $X \cup Y$, ou seja, o suporte diz respeito ao número de instâncias que são corretamente preditas (Witten, 2005). Dado um conjunto D de transações, o problema de encontrar regras de associação constitui-se em gerar todas as regras de associação que tenham suporte e confiança maiores que um mínimo (respectivamente *minsup* e *minconf).*

(Agrawal, 1994) propõe que a descoberta de regras de associação seja decomposta em dois subproblemas:

- 1-Encontrar todos os conjuntos de itens (*itemsets)* que possuam suporte acima de um mínimo estabelecido. Estes são denominados *itemsets freqüentes* ou *itemsets grandes.* O algoritmo Apriori apresentado na próxima seção propõe-se a resolver esse problema.
- 2-Utilizar os *itemsets* freqüentes para gerar as regras de associação. Um algoritmo geral para isso é o seguinte: para cada *itemset* freqüentes f encontre todos os subconjuntos

não vazios do mesmo. Para cada subconjunto a produzir uma regra $a \rightarrow (l - a)$ se a razão *suporte(l)/suporte(a)* for maior que *minconf.*

O Algoritmo *Apriori*

Proposto por (Agrawal,1994), esse algoritmo é bastante adequado ao tratamento de grandes volumes de dados. Para descrevê-lo, assume-se o seguinte (Agrawal, 1994):

- Itens em cada transação são mantidos ordenados em ordem lexicográfica.
- Cada registro da base de dados *D* é um par < *TID, item*>
- A notação utilizada para representar o tamanho (número de itens) *k* de um *itemset* é *k-itemset.*
- Itens em um *itemset* são mantidos em ordem lexicográfica. *c[1].c[2].....c[k]* representa um *k-itemset* constituído de itens *c[1],c[2],...,c[k]* onde *c[1]<c[2]<.....<c[k]*
- Se *c=X.Y* e *Y* é um *m-itemset , Y* é denominada uma *m-extension* de *X.*
- L_k é um conjunto de *k-itemsets* grandes, constituído de dois campos: o *itemset* e a *contagem do suporte* (*count*).
- C_k é um conjunto de *k-itemsets* candidatos (potencialmente grandes) , constituído dos campos *itemset* e a *contagem do suporte* (*count*).
- $\overline{C}_K$ é um conjunto de *k-itemsets* candidatos quando os TIDs das transações geradas são mantidas associadas aos candidatos.

O algoritmo *Apriori* é descrito da seguinte forma:

L_1 = {*large 1-itemsets};* //conjunto com somente um *itemset* grande
Para (*k=2;* $L_{k-1} \neq 0$*;k++*) faça:
    C_k= *apriorigen*(L_{k-1}*);* //Obtém novas candidatas
      Para todas as transações $t \in D$ faça:
          C_t= *subconjunto*(C_k,*t);* //candidatas contidas em t
           Para todas candidatas $c \in C_t$ faça //contagem das candidatas

c.count++;

fim;

fim;

$L_k = \{c \in C_k \mid c.count \geq \text{minsup}\}$

fim;

Retornar $\bigcup_k L_k$;

A função *apriorigen(L)* é utilizada para obter as novas transações candidatas em dois passos:

Passo de junção entre L_{k-1} e L_{k-1} (geração de um conjunto candidato Ck, através da eliminação de itens) :

Insert into C_k

Select $p.item_1$, $p.item_2$, ...,$p.item_{k-1}$, $q.item_{k-2}$

From L_{k-1} p, L_{k-1} q

Where $p.item_1 = q.item_1, \ldots, p.item_{k-2} = q.item_{k-2}, p.item_{k-1} < q.item_{k-1}$

Passo de poda (excluir todos os itemsets $c \in C_k$ de tal forma que algum (k-1)-subconjunto de c não está em L_{k-1}):

Para todos *itemsets* $c \in C_k$ faça

Para todos (k-1)-subconjuntos *s* de *c* faça

Se ($s \notin L_{k-1}$)

Então remova *c* de C_k ;

Fim;

Fim;

O seguinte exemplo ilustra os passos de junção e poda (Agrawal, 1994). Seja L_3 ={ {1 2 3}, {1 2 4}, {1 3 4}, {1 3 5}, {2 3 4} }. Após o passo de junção tem-se C_4={{1 2 3 4}, {1 3 4

5}}. Após o passo de poda {1 3 4 5} será excluído porque os *itemsets* {1 4 5} e {3 4 5} não estão em L_3.Portanto C_4={1 2 3 4}.

Sugere-se que função *subconjunto(Ck,t)* seja estruturada de tal forma que os *itemsets* candidatos C_k sejam armazenados em uma árvore *hash.* Um nó da árvore *hash* pode conter um nó-folha(uma lista de *itemsets)* ou um nó interior (uma tabela de *hash* contendo *itemsets)* . Esta função encontra todas as candidatos C_k em uma transação *t* da seguinte forma: se estamos em uma folha, procuramos quais dos *itemsets* na mesma estão contidos em em *t* e adicionamos referências a eles no conjunto-resposta. Se estamos em um nó interior e o alcançamos buscando o item *i* na tabela de *hash,* buscamos nesta cada item que venha depois de *i* em *t* e recursivamente aplicamos esse procedimento ao nó na entrada correspondente.

(Agrawal, 1994) apresenta uma variação do algoritmo Apriori denominada *a priori TID.* A característica interessante desse algoritmo é que a base de dados *D* não é utilizada para contabilizar o suporte após a primeira passagem. Para tal, é utilizado o conjunto $\overline{C}_K$. Cada membro do conjunto $\overline{C}_K$ é utilizado na forma < *TID, {Xk}* > onde cada X_k é um *k-itemset* potencialmente grande presente na transação com identificador *TID.* Esse algoritmo pode ser descrito da seguinte forma:

L_1 = {*large 1-itemsets};* //conjunto com somente um *itemset* grande

$\overline{C}_1$ = base de dados *D.*

Para (k=2; $L_{k-1} \neq 0$;k++) faça:

C_k= *apriorigen*(L_{k-1}); //Obtém novas candidatas

$\overline{C}_K$ =0;

Para todas as transações $t \in \overline{C}_{K-1}$ faça:

//Determinar *itemsets* candidatos em C_k contidos em *t.TID*

$C_t = \{c \in C_k \mid (c - c[k]) \in t..conjunto\text{-}de\text{-}itemsets \wedge (c - c[k-1]) \in t..conjunto\text{-}de\text{-}itemsets\}$;

Para todas candidatas $c \in C_t$ faça //contagem das candidatas

c.count++;

fim;

Se ($C_t \neq 0$) então $\overline{C}_K$ += <*t.TID*, C_t>;

fim;

$L_k = \{c \in C_k \mid c.count \geq \text{minsup}\}$

fim;

Retornar $\bigcup_k L_k$;

Uma Abordagem Alternativa: Conjuntos Freqüentes Fechados

Esta abordagem reduz o número de *itemsets* freqüentes e conseqüentemente, o número de regras a gerar. Inicialmente, realiza-se o cômputo dos *itemsets* freqüentes para, em seguida, eliminar os que não sejam fechados (Bogorny, 2006). Um TIDSET é definido como o conjunto das transações onde determinados *itemsets* freqüentes ocorrem. Por definição, um *itemset* freqüente é considerado *fechado* se não houver um *itemset* com cardinalidade maior que a sua no TIDSET que o contém. Assim sendo, os *itemsets* freqüentes fechados constituem os *itemsets freqüentes mínimos não redundantes.* Segundo (Bogorny,2006), *itemsets* freqüentes não fechados possuem o mesmo suporte que seus respectivos *itemsets* freqüentes fechados, que são então *itemsets* freqüentes máximos. Isso garante que regras geradas a partir de *itemsets* freqüentes não fechados sejam redundantes em relação a seus respectivos *itemsets* freqüentes fechados. Por conseguinte, de acordo com (Pasquier,1999), todos os *itemsets* freqüentes que ocorrem na mesma transação geram regras com o mesmo suporte e a mesma confiança. Assim, como o *itemset* freqüente máximo contém todos os outros *itemsets* freqüentes, estes geram regras redundantes, que podem então ser descartadas.

Method Summary

RegraAssociacao[]	**retornarregras**(ConjuntoInstancias conjunto, double minsup, double minconf) Retorna conjunto de regras de associação em um conjunto de instâncias

Method Detail

retornarregras

```
RegraAssociacao[] retornarregras(ConjuntoInstancias conjunto,
                                 double minsup,
                                 double minconf)
```

Retorna conjunto de regras de associação em um conjunto de instâncias

Parameters:

`conjunto` - conjunto de instâncias

`minsup` - suporte mínimo

`minconf` - confiança mínima

Returns:

Conjunto de regras de associação

See Also:

analisedados.tecnicasdatamining.dmregraassociacaoespacial

Interface ISBRegraAssociacaoEspacial

All Superinterfaces:
ISBRegraAssociacao

```
public interface ISBRegraAssociacaoEspacial
extends ISBRegraAssociacao
```

Interface que especifica métodos para obtenção de regras de associação espacial

Uma regra de associação espacial é uma regra na forma $X \rightarrow Y$ onde *X* e *Y* são conjuntos de predicados e alguns destes são *espaciais* (Koperski, 1995). Nesse caso, cada linha é uma instância de uma classe e as colunas são *predicados* (Bogorny,2006). Cada predicado pode estar associado a um atributo não espacial ou a uma relação espacial. Predicados espaciais representam relações espaciais entre entidades geográficas, que podem ser:

- *Topológicas:* por exemplo, *toca(A,B), dentrode(A,B), disjuntos(A,B), contém(A,B), sobrepõe(A,B), cruza(A,B), interesecciona(A,B), etc.*
- *De distância: distância(A,B)=10;*
- *De ordem: nortede(A,B), sudoeste(A,B), etc.*

Relacionamentos espaciais podem ter a seguinte classificação (Bogorny, 2006):

- *Obrigatórios:* a relação entre objetos é uma dependência obrigatória entre os mesmos,como por exemplo, *contida(ILHA, PORÇÃO DE ÁGUA);*
- *Proibidos:* a relação entre objetos é impossível no mundo real, como por exemplo, *dentrode(ESTRADA, RIO);*
- *Possíveis:* são relacionamentos não definidos previamente, como por exemplo, *cruza(ESTRADA, RIO);*

Em mineração de regras espaciais, interessam-nos as *relações possíveis*, os outros tipos são descartados.

Como exemplo de regra de associação tem-se: $rio(X) \wedge rio(Y) \wedge intersecciona(X,Y) \rightarrow afluentede(X,Y)$.

Processo para Geração de Regras de Associação Espacial

(Bogorny,2006) propõe um processo para geração de regras de associação espacial que tem sido aclamado como um dos mais eficientes e que proporciona melhor suporte à eliminação de relacionamentos não relevantes. Tal processo é decomposto nas seguintes etapas:

a)Pré-Processamento de Dados

Esta etapa é dividida nos seguintes passos:

a1-*Extrair as dependências geográficas entre pares de feições geográficas* (denominadas restrições de conhecimento). Tais dependências são extraídas da base de dados da seguinte forma (Bogorny,2006): procura-se por todas as chaves estrangeiras sendo que, para cada uma delas, é recuperado o nome da tabela que referencia e o nome da tabela onde é especificada. Os dois nomes são armazenados em um conjunto ϕ de restrições de conhecimento e são utilizados nos passos seguintes para melhorar o pré-processamento de dados geográficos.

a2-*Recuperação de metadados da base de dados*. Neste passo, informações são recuperadas através de um esquema de metadados do *Open Geospatial Consortium (OGC),* incluindo todas as relações na base de dados (tipos de feições). A partir destas, seleciona-se o tipo de feição-alvo *T* (por exemplo, cidade), os atributos não espaciais *A* da feição-alvo *T* (por exemplo, população), o conjunto *S* de tipos de feição relevantes (por exemplo, rio) que possam ter alguma influência sobre *T,* as relações espaciais *R* (por exemplo, *toca, contém, perto, longe, etc)* e o nível de granularidade *g* para cada tipo de feição relevante *O* em *S,* quando uma hierarquia conceitual H_0 é dada para *O.* A granularidade *g* é um número inteiro com tamanho *1...n,* onde *1* é o nível mais geral e *n* o mais específico. Aqui, o nível *1* corresponde á granularidade do tipo de feição e o nível *2* à granularidade da instância do tipo de feição

a3-*Eliminação de dependências*, que verifica as associações entre o tipo de feição-alvo e todos os tipos de feição relevantes. O conjunto ϕ de restrições de conhecimento é pesquisado

e se é encontrada uma dependência entre T e $O \subseteq S$ então O é eliminado de S. Para cada tipo de feição relevante removido de S, não se requer qualquer junção espacial para extrair relações espaciais (Bogorny, 2006). Consequentemente não serão gerados nem conjuntos freqüentes, nem regras de associação espacial com esse tipo de feição relevante.

a4-*Junção espacial,* que obtém as relações R entre T e O para todos os tipos de feição relevantes $O \subseteq S$. Este passo extrai de uma base de dados D, via operações espaciais providas pelo banco de dados, as relações espaciais R entre as instâncias de T e todas as instâncias de $O \subseteq S$, ao nível de granularidade $g_0 \subseteq G$, de acordo com uma hierarquia $h_0 \subseteq H$ e para cada O em S.

a5- *Transformação,* que transpõe e discretiza o conjunto de dados obtido Ψ em uma tabela em que cada linha corresponde a uma instância do tipo de feição-alvo e os atributos são os predicados (relações) obtidos.

b)*Geração dos conjuntos freqüentes*

Segundo (Bogorny, 2006) a melhor forma de se eliminarem as dependências geográficas bem conhecidas é excluindo conjuntos candidatos cujas dependências apareçam em primeiro lugar. Com esse propósito, (Bogorny, 2006) modificou o algoritmo Apriori (Agrawal,1994), visto anteriormente, de forma a considerar as restrições de conhecimento ϕ obtidas na etapa de pré-processamento. O algoritmo modificado, denominado *Apriori-KC*, é descrito da seguinte forma (Bogorny, 2006):

L_1 = {*large 1-conjunto de predicados};*
Para (*k=2;* $L_{k-1} \neq 0$ *;k++*) faça:
    C_k= *apriorigen*(L_{k-1}); //Obtém novas candidatas
      Se (*k=2)*
        //remover pares com dependência
     (Passo 1) *Exclua de* C_2 *todos os pares com uma dependência em* ϕ*;*

//Remover pares com dependência hierárquicas

(Passo 2) *Exclua de* C_2 *todos os pares com uma dependência hierárquica H em* ϕ*;*

//Remover pares com mesmo tipo de feição e diferentes relações topológicas //(por exemplo, {contém(água), toca(água)}

(Passo 3) *Exclua de* C_2 *todos os pares com o mesmo tipo de feição;*

//Remover pares com diferentes tipo de feição e que tenham o mesmo pai em // H, por exemplo {toca(rio), toca(lago)}

(Passo 4) *Exclua de* C_2 *todos os pares com o mesmo pai em H;*

Para todas as linhas $w \in \Psi$ faça:

C_t= *subconjunto*(C_k,*w*); //candidatas contidas em t

Para todas candidatas $c \in C_w$ faça //contagem das candidatas

c.count++;

fim;

fim;

$L_k = \{c \in C_k \mid c.count \geq \text{minsup}\}$

fim;

Retornar $\bigcup_k L_k$;

c)*Geração de Conjuntos Freqüentes Não Redundantes Máximos*

(Bogorny, 2006) considera que, como a abordagem conjuntos freqüentes fechados gera menos conjuntos freqüentes porque elimina conjuntos freqüentes geradores de regras de associação espacial redundantes, seria interessante supor que as dependências geográficas possam também ser eliminadas a partir de conjuntos freqüentes fechados. Entretanto, conclui a partir de um exemplo, que a eliminação das dependências geográficas resulta em conjuntos freqüentes que não são fechados em relação ao conjunto de dados, ainda que sejam em relação aos conjuntos freqüentes, tornando possível a geração de regras redundantes. Assim sendo, propõe a geração de *conjuntos freqüentes não redundantes máximos,* por meio do algoritmo *Max-FGP,* apresentado a seguir.

A idéia básica do algoritmo *Max-FGP* é inicialmente eliminar as dependências geográficas bem conhecidas e para todos os conjuntos freqüentes que ocorram na mesma transação, todos são eliminados, exceto os que sejam redundantes máximos. Segundo (Bogorny, 2006) um padrão geográfico freqüente *L* é máximo quando não possui uma dependência geográfica bem conhecida em ϕ tal que $L\text{-}\phi = L$ e $M(L) = L$. O operador *M* associa um conjunto de predicados freqüentes *L* com o conjunto máximo de predicados comuns a todas as transações que contenham *L,* sem dependências geográficas bem conhecidas, ou seja, *L* será máximo se não houver um conjunto de predicados freqüentes *L'* nas mesmas transações de *L* tal que $L \subset L'$. A seguir, o pseudocódigo para o algoritmo Max-FGP (Bogorny, 2006).

L_k *:* conjuntos freqüentes gerados pelo algoritmo Apriori-KC

Ψ*:* conjunto de dados gerado na etapa anterior

M=L;

Para ($k=2$; $M_k \neq 0$;k++) faça:

Para ($j=K+1$; $M_J \neq 0$;j++) faça:

Se (TIDSET(M_k)= TIDSET(M_j)

Se ($M_k \subset M_j$) exclua M_k de *M*;

Fim;

Fim;

Retorne M;

d)*Gerar as regras de associação espacial*

Nesta última etapa, as regras são geradas de forma convencional, ou seja, para cada conjunto freqüente *f* encontrar todos os subconjuntos não vazios do mesmo. Para cada subconjunto *a* produzir uma regra $a \rightarrow (l - a)$ se a razão *suporte(l)/suporte(a)* for maior que *minconf.*

Method Summary

PredicadoEspacial	**obterpredicadosespaciais**(ConjuntoInstancias instancia Retorna predicados espaciais dados um conjunto de intância

Methods inherited from interface analisedados.tecnicasdatamining.dmregraassociacao.ISBRegraAssociacao

retornarregras

Method Detail

obterpredicadosespaciais

PredicadoEspacial[]
obterpredicadosespaciais(ConjuntoInstancias instancias)

Retorna predicados espaciais dado um conjunto de intâncias

Parameters:

instancias - conjunto de instâncias

-

Returns:

conjunto de predicados espaciais

See Also:

analisedados.tecnicasdeia.neurais

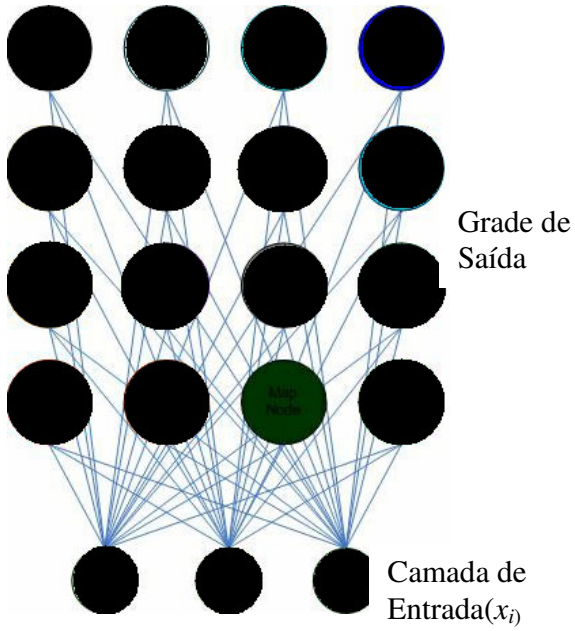
Interface ISBSOM

```
public interface ISBSOM
```

Interface que especifica métodos que implementam redes neurais SOM

As redes *self-organizing maps* (SOM - mapas auto-organizáveis) ou redes de Kohonen (Kohonen, 2001), diferem bastante das redes MLP, pela estrutura, por serem em essência um mapeamento da camada de entrada para a camada de saída, uma grade reticular usualmente bidimensional, e pelo treinamento, que é não-supervisionado.

Os neurônios da rede SOM não possuem função de ativação. Sua estrutura é tal que os neurônios em sua entrada são todos conectados a cada unidade da saída por meio de pesos. No treinamento, padrões apresentados na camada de entrada fazem com que os nós na camada de saída se ajustem aos mesmos, produzindo um mapa em que padrões semelhantes sejam identificados por grupos de nós próximos.



Estrutura da rede SOM

Cada nó *j* da grade de saída possui uma posição topológica específica (suas coordenadas *x* e *y)* e contém em sua entrada o vetor x_i da camada de entrada, com um peso w_{ji} que conecta

este nó j ao elemento i do vetor de entrada. A saída y_j de cada nó j é dada pela distância euclidiana entre as entradas x_i e os pesos w_{ji}:

$$y_j = \sqrt{\sum_{i=1}^{n} (x_i - w_{ji})^2}$$

Treinamento das redes SOM

Dado um conjunto de treinamento constituído de p entradas x_i , cada vez que um elemento deste conjunto é apresentado à rede, ocorre uma competição entre os nós da camada de saída, pela maior saída,dada pela distância euclidiana (ver acima). O nó vencedor (BMU-*Best Matching Unit)* e seus vizinhos dentro de um raio (para determinação do qual existem várias formas), têm seus pesos atualizados. A cada iteração do algoritmo, a taxa de aprendizado e o raio de vizinhança são decrementados.
O algoritmo de treinamento da rede SOM pode ser assim formalizado (inspirado em (Pádua, 2000):

Seja:

$\lambda(t)$ a vizinhança do nó vencedor, que é função da iteração t
$\eta(t)$ a taxa de aprendizado, que diminui também em função da iteração t . (Ver abaixo redução da taxa de aprendizado)

1-Iniciar pesos com valores randômicos pequenos;
2-Até que o mapa de características fique inalterado ou que um certo número de iterações t ocorram, faça:

2.1-Para cada um dos p elementos do conjunto de treinamento faça:
2.1.1-Para cada nó j calcular a distância euclidiana y_j entre o peso w_{ji} e a entrada x_i :

$$y_j = \left\| x_i - w_{ji} \right\| = \sqrt{\sum_{i=1}^{n} (x_i - w_{ji})^2}$$

2.1.2-Definir como nó vencedor aquele tiver a menor distância euclidiana y_j;

2.1.3-Determinar os nós vizinhos ao nó vencedor (ver formas possíveis abaixo);

2.1.4-Atualizar os pesos do nós vencedores e dos nós vizinhos, da seguinte maneira:

$$w_{ji}(t+1) = \begin{cases} w_{ji}(t+1) + \eta(t)(x_i(t) - w_{ji}(t)) & se \;\; j \in \lambda(t) \\ w_{ji}(t) \;\; se \;\; j \notin \lambda(t) \end{cases}$$

Determinação dos nós vizinhos ao nó vencedor

(Schalkoff, 1997) sugere que a vizinhança seja inicialmente grande (do tamanho da grade), sendo reduzida progressivamente, em função do número de ciclos (de diversas formas possíveis, linear e exponencial são as mais comuns), até um limite pré-estabelecido. Essa vizinhança pode assumir diversos formatos diferentes, por exemplo, um quadrado ao redor do nodo, um hexágono, ou até mesmo um círculo. O formato ideal, geralmente, é determinado por tentativa-e-erro e depende do problema e da distribuição dos dados.

Redução da taxa de aprendizado

(Pádua, 2000) sugere que em uma primeira fase, as primeiras 1000 iterações, a taxa de aprendizado deve ser alta, próxima de 1, ou seja, os pesos sofrem grandes modificações, sendo gradualmente reduzida até um aproximadamente 0,1 (essa redução pode ser linear, mais comum, ou exponencial).

Em uma segunda fase, que requer de 100 a 1000 vezes mais ciclos que a anterior, a taxa de aprendizado é baixa (da ordem de 0,01 ou menos) e o raio de vizinhança envolve um ou nenhum vizinho.

Melhorias no desempenho

Os pesos iniciais de uma rede SOM são definidos aleatoriamente. Assim sendo, a distância entre os nodos e o vetor de entrada pode se tornar muito grande, diminuindo a velocidade de convergência. As sugestões a seguir podem diminuir esse problema (Pádua, 2000):

- Utilizar valores iniciais de pesos iguais;
- Se um nó for selecionado, diminuir a chance de que o mesmo seja novamente selecionado;
- Reduzir a vizinhança.

Method Summary

double	**consultar**(RedeSOM redesom, double[] teste) Efetua a consulta em uma rede MLP
vo	**salvarrede**(RedeSOM redesom, java.lang.String arquivo) Salva uma rede SOM em um arquivo
RedeS	**trazerrede**(java.lang.String arquivo) Carrega uma rede SOM armazenada em um arquivo
i	**treinar**(RedeSOM redesom, int epocas) Efetua o treinamento de uma rede SOM

Method Detail

treinar

```
int treinar(RedeSOM redesom,
            int epocas)
```

Efetua o treinamento de uma rede SOM

Parameters:

`redemlp` - Rede SOM devidamente estruturada

`epocas` - Número de iterações

Returns:

Número de iterações necessário ao treinamento

---

consultar

```
double[] consultar(RedeSOM redesom,
                   double[] teste)
```

Efetua a consulta em uma rede MLP

Parameters:

`redesom` - Rede SOM devidamente estruturada

`teste` - Entrada submetida à rede

Returns:

Resposta da rede

---

trazerrede

```
RedeSOM trazerrede(java.lang.String arquivo)
```

Carrega uma rede SOM armazenada em um arquivo

Parameters:
`arquivo` - Nome do arquivo onde foi armazenada a rede SOM
Returns:
A rede SOM

---

salvarrede

```
void salvarrede(RedeSOM redesom,
                java.lang.String arquivo)
```

Salva uma rede SOM em um arquivo
Parameters:
`arquivo` - Nome do arquivo onde foi armazenada a rede SOM
`redemlp` - Rede SOM a ser salva

analisedados.modelosanaliseespacial

Interface ISBTaxi

`public interface` **`ISBTax`**

Interface que especifica métodos que implementam aspectos da geometria táxi

(Teoria apresentada no item 4.3.1)

Method Summary

void	**armazenaestrutura**(Rua[] ruas, Reta[] avenidastransv PontoInteresse[] pontosinteresse) Armazena estrutura da região a ser analisada
PontoInteresse[]	**executaquery**(Expressao query) Executa query traduzida por uma expressão Expressão=[(](Equação)[),and, or,(] [Expressão] Equação tem o formato geral: Equação= [(](Membro1 [=, >,<,<=,>=, <>] Membro 2) [),and, or,(] [Equação] Membro tem o formato geral Membro = [(] (Termo) [),+,-,*,/,(] [Membro] Termo tem o formato geral: Termo = [(] (Multiplicador* [DT ([Ponto,reta],[Ponto, reta])]) [),+,-,*,/,(][Termo] onde Multiplicador é um número real.
int	**retornadistanciataxi**(ObjetoGeograficoTaxi obj1, ObjetoGeograficoTaxi obj2) Retorna distância táxi entre dois objetos

Method Detail

armazenaestrutura

```
void armazenaestrutura(Rua[] ruas,
                       Reta[] avenidastransv,
                       PontoInteresse[] pontosinteresse)
```

Armazena estrutura da região a ser analisada

Parameters:

`ruas` - Ruas verticais e horizontais que formam o reticulado

`avenidastransv` - Avenidas transversais que cruzam o reticulado

`pontosinteresse` - Pontos de interesse que compõe a região a ser estudada

---

executaquery

```
PontoInteresse[] executaquery(Expressao query)
```

Executa query traduzida por uma expressão
Expressão=[(](Equação)[),and, or,(] [Expressão]
Equação tem o formato geral: Equação= [(](Membro1 [=, >,<,<=,>=, <>] Membro 2) [),and, or,(] [Equação]
Membro tem o formato geral: Membro = [(] (Termo) [),+,-,*,/,(] [Membro]

Termo tem o formato geral: Termo = [(] (Multiplicador* [DT ([Ponto,reta],[Ponto, reta])]) [),+,-,*,/,(][Termo] onde Multiplicador é um número real.
DT ([Ponto,reta],[Ponto, reta]) é uma função que retorna a distância táxi entre geometrias ponto e reta.

Parameters:

`expressao` - Expressão que contém a query a ser executada

Returns:

Pontos que satisfazem à query

See Also:

---

retornadistanciataxi

```
int retornadistanciataxi(ObjetoGeograficoTaxi obj1,
                         ObjetoGeograficoTaxi obj2)
```

Retorna distância táxi entre dois objetos

Parameters:

`obj1,` - obj2 Objetos cuja distância entre eles deseja-se calcular

Returns:

Distância táxi

ogcservices

Interface ISBWCS

`public interface` **`ISBWCS`**

Interface que especifica métodos de acesso ao WCS

Method Summary

vo	**DescribeCoverage**(java.lang.String url) Descreve a estrutura de uma coverage
Java.lang.Stri	**DescribeCoverageXML**(java.lang.String url) Descreve a estrutura de uma coverage
vo	**getCapabilities**(java.lang.String url) Tem como objetivo a obtenção de metadados do serviço, ou sej uma descrição do conteúdo das informações do servidor e valores dos parâmetros de requisição.
Java.lang.Stri	**getCapabilitiesXML**(java.lang.String url) Tem como objetivo a obtenção de metadados do serviço, ou sej uma descrição do conteúdo das informações do servidor e valores dos parâmetros de requisição.
vo	**getCoverage**(java.lang.String url) Recupera propriedades de um coverage
Java.lang.Stri	**getCoverageXML**(java.lang.String url) Recupera propriedades de um coverage

Method Detail

getCapabilities

`void` **`getCapabilities`**`(java.lang.String url)`

Tem como objetivo a obtenção de metadados do serviço, ou seja, uma descrição do conteúdo das informações do servidor e valores dos parâmetros de requisição.

Parameters:

`url` - URL da requisição, que deve possuir os seguintes parâmetros:

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=version	M	Número de versão
SERVICE=WCS	M	Tipo de serviço
REQUEST=DescribeCoverage	M	Nome da requisição
IDENTIFIERS=identifier	M	Lista de identificadores de geo-campos a serem descritos, separados por vírgula

See Also:

getCapabilitiesXML

`java.lang.String` **`getCapabilitiesXML`**`(java.lang.String url)`

Tem como objetivo a obtenção de metadados do serviço, ou seja, uma descrição do conteúdo das informações do servidor e valores dos parâmetros de requisição.

Parameters:

`url` - URL da requisição, que deve possuir os seguintes parâmetros:

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=version	M	Número de versão
SERVICE=WCS	M	Tipo de serviço
REQUEST=DescribeCoverage	M	Nome da requisição
IDENTIFIERS=identifier	M	Lista de identificadores de geo-campos a serem descritos, separados por vírgula

Returns:

String do código XML contendo os metadados

See Also:

DescribeCoverage

`void` **`DescribeCoverage`**`(java.lang.String url)`

Descreve a estrutura de uma coverage

Parameters:

`url` - URL da requisição, que deve possuir os seguintes parâmetros:

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=version	M	Número de versão
SERVICE=WCS	M	Tipo de serviço
REQUEST=DescribeCoverage	M	Nome da requisição
IDENTIFIERS=identifier	M	Lista de identificadores de geo-campos a serem descritos, separados por vírgula

See Also:

DescribeCoverageXML

`java.lang.String` **`DescribeCoverageXML`**`(java.lang.String url)`

Descreve a estrutura de uma coverage

Parameters:

`url` - URL da requisição, que deve possuir os seguintes parâmetros:

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=version	M	Número de versão
SERVICE=WCS	M	Tipo de serviço
REQUEST=DescribeCoverage	M	Nome da requisição
IDENTIFIERS=identifier	M	Lista de identificadores de geo-campos a serem descritos, separados por vírgula

Returns:
String do código XML contendo a descrição da estrutura da coverage
See Also:

---

getCoverage
`void` **`getCoverage`**`(java.lang.String url)`
Recupera propriedades de um coverage
Parameters:

`url` - URL da requisição, que deve possuir os seguintes parâmetros:

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=version	M	Número de versão
SERVICE=WCS	M	Tipo de serviço
REQUEST=GetCoverage	M	Nome da requisição
IDENTIFIERS=identifier	M	Lista de identificadores de geo-campos a serem descritos, separados por vírgula
BBOX=minx,miny,maxx,maxy	M	Requisição de um subconjunto de *coverages* definido pelos cantos da caixa de contorno do mapa (inferior esquerdo, superior direito)
TIMESEQUENCE=tempo	O	Requisição de um subconjunto correspondente aos instantes de tempo, separados por vírgula.
RANGESUBSET	O	Requisição de somente alguns campos ou subconjuntos de alguns campos
FORMAT= image/netcdf	M	Formato de saída
STORE= true	O	Especifica se a resposta deve ser armazenada.

See Also:

---

getCoverageXML

`java.lang.String` **`getCoverageXML`**`(java.lang.String url)`

Recupera propriedades de um coverage

Parameters:

`url` - URL da requisição, que deve possuir os seguintes parâmetros:

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=version	M	Número de versão
SERVICE=WCS	M	Tipo de serviço
REQUEST=GetCoverage	M	Nome da requisição
IDENTIFIERS=identifier	M	Lista de identificadores de geo-campos a serem descritos, separados por vírgula
BBOX=minx,miny,maxx,maxy	M	Requisição de um subconjunto de *coverages* definido pelos cantos da caixa de contorno do mapa (inferior esquerdo, superior direito)
TIMESEQUENCE=tempo	O	Requisição de um subconjunto correspondente aos instantes de tempo, separados por vírgula.
RANGESUBSET	O	Requisição de somente alguns campos ou subconjuntos de alguns campos
FORMAT= image/netcdf	M	Formato de saída
STORE= true	O	Especifica se a resposta deve ser armazenada.

Returns:
String do código XML contendo as propriedades da coverage
See Also:

ogcservices

Interface ISBWFS

`public interface` **`ISBWFS`**

Interface que especifica métodos de acesso ao WFS

Method Summary

vo	**getCapabilities**(java.lang.String url) Tem como objetivo a obtenção de metadados do serviço, ou sej uma descrição do conteúdo das informações do servidor e valores dos parâmetros de requisição.
java.lang.Stri	**getCapabilitiesXML**(java.lang.String url) Tem como objetivo a obtenção de metadados do serviço, ou sej uma descrição do conteúdo das informações do servidor e valores dos parâmetros de requisição.
vo	**getFeature**(java.lang.String url) Retorna instâncias das feições espaciais contidas na base de dad
vo	**getFeatureType**(java.lang.String url) Descreve a estrutura de uma feição geográfica
java.lang.Stri	**getFeatureTypeXML**(java.lang.String url) Descreve a estrutura de uma feição geográfica
vo	**getFeatureWithLock**(java.lang.String url) Bloqueia uma ou mais instâncias que foram selecionadas
java.lang.Stri	**getFeatureWithLockXML**(java.lang.String url) Bloqueia uma ou mais instâncias que foram selecionadas
java.lang.Stri	**getFeatureXML**(java.lang.String url) Retorna instâncias das feições espaciais contidas na base de dad
vo	**LockFeature**(java.lang.String url) Bloqueia uma feição para alteração
java.lang.Stri	**LockFeatureXML**(java.lang.String url) Bloqueia uma feição para alteração
vo	**Transaction**(java.lang.String url) Utilizada em operações de inserção, exclusão ou alteração de feições.
java.lang.Stri	**TransactionXML**(java.lang.String url) Utilizada em operações de inserção, exclusão ou alteração de feições.

Method Detail

getCapabilities

`void` **`getCapabilities`**`(java.lang.String url)`
Tem como objetivo a obtenção de metadados do serviço, ou seja, uma descrição do conteúdo das informações do servidor e valores dos parâmetros de requisição.
Parameters:
`url` - URL da requisição, que deve possuir os seguintes parâmetros:

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=version	O	Número de versão
SERVICE=WFS	M	Tipo de serviço
REQUEST=GetCapabilities	M	Nome da requisição
NAMESPACE = namespace	O	Especifica um namespace e seus prefixos

See Also:

---

getCapabilitiesXML
`java.lang.String` **`getCapabilitiesXML`**`(java.lang.String url)`
Tem como objetivo a obtenção de metadados do serviço, ou seja, uma descrição do conteúdo das informações do servidor e valores dos parâmetros de requisição.
Parameters:
`url` - URL da requisição, que deve possuir os seguintes parâmetros:

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=version	O	Número de versão
SERVICE=WFS	M	Tipo de serviço
REQUEST=GetCapabilities	M	Nome da requisição
NAMESPACE = namespace	O	Especifica um namespace e seus prefixos

Returns:
String do código XML contendo os metadados
See Also:

---

getFeatureType
`void` **`getFeatureType`**`(java.lang.String url)`
Descreve a estrutura de uma feição geográfica
Parameters:

`url` - URL da requisição, que deve possuir os seguintes parâmetros:

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=version	O	Número de versão
SERVICE=WFS	M	Tipo de serviço
REQUEST=DescribeFeatureType	M	Nome da requisição
NAMESPACE = namespace	O	Especifica um namespace e seus prefixos
TYPENAME=namespace:featuretype	O	Uma lista de elementos separados por vírgula onde Namespace é o domínio e featuretype é o nome do tipo de feição
OUTPUTFORMAT=myme_type	O	Formato de saída

See Also:

---

getFeatureTypeXML

`java.lang.String` **`getFeatureTypeXML`**`(java.lang.String url)`

Descreve a estrutura de uma feição geográfica

Parameters:

`url` - URL da requisição, que deve possuir os seguintes parâmetros:

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=version	O	Número de versão
SERVICE=WFS	M	Tipo de serviço
REQUEST=DescribeFeatureType	M	Nome da requisição
NAMESPACE = namespace	O	Especifica um namespace e seus prefixos
TYPENAME=namespace:featuretype	O	Uma lista de elementos separados por vírgula onde Namespace é o domínio e featuretype é o nome do tipo de feição
OUTPUTFORMAT=myme_type	O	Formato de saída

Returns:

String do código XML contendo a estrutura da feição

See Also:

---

getFeature

`void` **`getFeature`**`(java.lang.String url)`

Retorna instâncias das feições espaciais contidas na base de dados

Parameters: `url` - URL da requisição, que deve possuir os seguintes parâmetros:

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=version	O	Número de versão
SERVICE=WFS	M	Tipo de serviço
REQUEST=DescribeFeatureType	M	Nome da requisição
NAMESPACE = namespace	O	Especifica um namespace e seus prefixos
TYPENAME=namespace:featuretype	M	Lista de tipos de feições a selecionar, onde Namespace é o domínio e featuretype é o nome do tipo de feição
FEATUREID=feature	O	Filtro para de determinada feição
MAXFEATURES= N	O	Número máximo de feições a serem retornadas
SORTBY=property	O	Determina uma lista de atributos (property) para os quais será feita uma ordenação no formato "PropertyName [A|D][,PropertyName [A|D],...]"
PROPERTYNAME=property	O	Lista de propriedades a serem selecionadas, separadas por vírgula
FEATUREVERSION=[all,N]	O	Versões das feições
FILTER	O	Filtros a serem aplicados às feições, separados por conectores lógicos *and*. Deve haver um filtro para cada TYPENAME
BBOX=minx,miny,maxx,maxy	M	Se FEATUREID ou FILTER forem usados, pode-se especificar uma caixa de contorno.
RESULTTYPE= resulttype	O	Usado para indicar se um WFS deve gerar um documento de resposta completo (*results*) ou um documento contendo

See Also:

getFeatureXML

`java.lang.String` **`getFeatureXML`**`(java.lang.String url)`

Retorna instâncias das feições espaciais contidas na base de dados

Parameters: `url` - URL da requisição, que deve possuir os seguintes parâmetros:

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=version	O	Número de versão
SERVICE=WFS	M	Tipo de serviço
REQUEST=DescribeFeatureType	M	Nome da requisição
NAMESPACE = namespace	O	Especifica um namespace e seus prefixos
TYPENAME=namespace:featuretype	M	Lista de tipos de feições a selecionar, onde Namespace é o domínio e featuretype é o nome do tipo de feição
FEATUREID=feature	O	Filtro para de determinada feição
MAXFEATURES= N	O	Número máximo de feições a serem retornadas
SORTBY=property	O	Determina uma lista de atributos (property) para os quais será feita uma ordenação no formato "PropertyName [A|D][,PropertyName [A|D],...]"
PROPERTYNAME=property	O	Lista de propriedades a serem selecionadas, separadas por vírgula
FEATUREVERSION=[all,N]	O	Versões das feições
FILTER	O	Filtros a serem aplicados às feições, separados por conectores lógicos *and*. Deve haver um filtro para cada TYPENAME
BBOX=minx,miny,maxx,maxy	M	Se FEATUREID ou FILTER forem usados, pode-se especificar uma caixa de contorno.
RESULTTYPE= resulttype	O	Usado para indicar se um WFS deve gerar um documento de resposta completo (*results*) ou um documento contendo

Returns:

String do código XML contendo a estrutura da feição

See Also:

getFeatureWithLock

`void` **`getFeatureWithLock`**`(java.lang.String url)`

Bloqueia uma ou mais instâncias que foram selecionadas

Parameters: `url` - URL da requisição, que deve possuir os seguintes parâmetros:

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=version	O	Número de versão
SERVICE=WFS	M	Tipo de serviço
REQUEST=DescribeFeatureType	M	Nome da requisição
NAMESPACE = namespace	O	Especifica um namespace e seus prefixos
TYPENAME=namespace:featuretype	M	Lista de tipos de feições a selecionar, onde Namespace é o domínio e featuretype é o nome do tipo de feição
FEATUREID=feature	O	Filtro para de determinada feição
MAXFEATURES= N	O	Número máximo de feições a serem retornadas
SORTBY=property	O	Determina uma lista de atributos (property) para os quais será feita uma ordenação no formato "PropertyName [A|D][,PropertyName [A|D],...]"
PROPERTYNAME=property	O	Lista de propriedades a serem selecionadas, separadas por vírgula
FEATUREVERSION=[all,N]	O	Versões das feições
FILTER	O	Filtros a serem aplicados às feições, separados por conectores lógicos *and*. Deve haver um filtro para cada TYPENAME
BBOX=minx,miny,maxx,maxy	M	Se FEATUREID ou FILTER forem usados, pode-se especificar uma caixa de contorno.
RESULTTYPE= resulttype	O	Usado para indicar se um WFS deve gerar um documento de resposta completo (*results*) ou um documento contendo

.gif">

See Also:

---

getFeatureWithLockXML

`java.lang.String` **`getFeatureWithLockXML`**`(java.lang.String url)`

Bloqueia uma ou mais instâncias que foram selecionadas

Parameters: `url` - URL da requisição, que deve possuir os seguintes parâmetros:

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=version	O	Número de versão
SERVICE=WFS	M	Tipo de serviço
REQUEST=DescribeFeatureType	M	Nome da requisição
NAMESPACE = namespace	O	Especifica um namespace e seus prefixos
TYPENAME=namespace:featuretype	M	Lista de tipos de feições a selecionar, onde Namespace é o domínio e featuretype é o nome do tipo de feição
FEATUREID=feature	O	Filtro para de determinada feição
MAXFEATURES= N	O	Número máximo de feições a serem retornadas
SORTBY=property	O	Determina uma lista de atributos (property) para os quais será feita uma ordenação no formato "PropertyName [A|D][,PropertyName [A|D],...]"
PROPERTYNAME=property	O	Lista de propriedades a serem selecionadas, separadas por vírgula
FEATUREVERSION=[all,N]	O	Versões das feições
FILTER	O	Filtros a serem aplicados às feições, separados por conectores lógicos *and*. Deve haver um filtro para cada TYPENAME
BBOX=minx,miny,maxx,maxy	M	Se FEATUREID ou FILTER forem usados, pode-se especificar uma caixa de contorno.
RESULTTYPE= resulttype	O	Usado para indicar se um WFS deve gerar um documento de resposta completo (*results*) ou um documento contendo

Returns:
String do código XML
See Also:

LockFeature
`void` **`LockFeature`**`(java.lang.String url)`
Bloqueia uma feição para alteração

Parameters:

`url` - URL da requisição, que deve possuir os seguintes parâmetros:

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=version SERVICE=WFS	O M	Número de versão Tipo de serviço
REQUEST=LockFeature	M	Nome da requisição
TYPENAME=namespace:featuretype	M	Lista de tipos de feições a selecionar, onde Namespace é o domínio e featuretype é o nome do tipo de feição
OPERATION=delete LOCKACTION=[ALL|SOME]	M O	Operação a executar ALL tenta bloquear toda as feições. SOME tenta bloquear o máximo possível.
FEATUREID=feature	O	Filtro para de determinada feição
FILTER	O	Filtros a serem aplicados às feições, separados por conectores lógicos *and*. Deve haver um filtro para cada TYPENAME
BBOX=minx,miny,maxx,maxy	M	Se FEATUREID ou FILTER forem usados, pode-se especificar uma caixa de contorno.

.gif">

See Also:

LockFeatureXML

`java.lang.String` **`LockFeatureXML`**`(java.lang.String url)`

Bloqueia uma feição para alteração

Parameters:

`url` - URL da requisição, que deve possuir os seguintes parâmetros:

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=version SERVICE=WFS	O M	Número de versão Tipo de serviço
REQUEST=LockFeature	M	Nome da requisição
TYPENAME=namespace:featuretype	M	Lista de tipos de feições a selecionar, onde Namespace é o domínio e featuretype é o nome do tipo de feição
OPERATION=delete LOCKACTION=[ALL|SOME]	M O	Operação a executar ALL tenta bloquear toda as feições. SOME tenta bloquear o máximo possível.
FEATUREID=feature	O	Filtro para de determinada feição
FILTER	O	Filtros a serem aplicados às feições, separados por conectores lógicos *and*. Deve haver um filtro para cada TYPENAME
BBOX=minx,miny,maxx,maxy	M	Se FEATUREID ou FILTER forem usados, pode-se especificar uma caixa de contorno.

Returns:
String do código XML
See Also:

Transaction
`void` **`Transaction`**`(java.lang.String url)`
Utilizada em operações de inserção, exclusão ou alteração de feições.
Parameters: `url` - URL da requisição, que deve possuir os seguintes parâmetros:

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=version	O	Número de versão
SERVICE=WFS	M	Tipo de serviço
REQUEST=Transaction	M	Nome da requisição
OPERATION=delete	M	Operação a executar
RELEASEACTION=[ALL|SOME]	O	ALL denota que todos os bloqueios de feições devem ser liberados quando a transação terminar. SOME, indica que somente os modificados devem ser liberados quando a transação terminar.
TYPENAME=namespace:featuretype	M	Lista de tipos de feições a selecionar, onde Namespace é o domínio e featuretype é o nome do tipo de feição
FEATUREID=feature	O	Filtro para de determinada feição
FILTER	O	Filtros a serem aplicados às feições, separados por conectores lógicos *and*. Deve haver um filtro para cada TYPENAME
BBOX=minx,miny,maxx,maxy	M	Se FEATUREID ou FILTER forem usados, pode-se especificar uma caixa de contorno.

.gif">
See Also:

TransactionXML

`java.lang.String` **`TransactionXML`**`(java.lang.String url)`

Utilizada em operações de inserção, exclusão ou alteração de feições.

Parameters:

`url` - URL da requisição, que deve possuir os seguintes parâmetros:

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=version	O	Número de versão
SERVICE=WFS	M	Tipo de serviço
REQUEST=Transaction	M	Nome da requisição
OPERATION=delete	M	Operação a executar
RELEASEACTION=[ALL|SOME]	O	ALL denota que todos os bloqueios de feições devem ser liberados quando a transação terminar. SOME, indica que somente os modificados devem ser liberados quando a transação terminar.
TYPENAME=namespace:featuretype	M	Lista de tipos de feições a selecionar, onde Namespace é o domínio e featuretype é o nome do tipo de feição
FEATUREID=feature	O	Filtro para de determinada feição
FILTER	O	Filtros a serem aplicados às feições, separados por conectores lógicos *and*. Deve haver um filtro para cada TYPENAME
BBOX=minx,miny,maxx,maxy	M	Se FEATUREID ou FILTER forem usados, pode-se especificar uma caixa de contorno.

Returns:
String do código XML
See Also:

ogcservices

Interface ISBWMS

Interface que especifica métodos de acesso ao WMS

```
public interface ISBWMS
```

Method Summary

vo	**getCapabilities**(java.lang.String url) Tem como objetivo a obtenção de metadados do serviço, ou sej uma descrição do conteúdo das informações do servidor e valores dos parâmetros de requisição.
java.lang.Stri	**getCapabilitiesXML**(java.lang.String url) Tem como objetivo a obtenção de metadados do serviço, ou sej uma descrição do conteúdo das informações do servidor e valores dos parâmetros de requisição.
vo	**getFeatureInfo**(java.lang.String url) Proporcionar aos clientes de um serviço WMS informações adicionais sobre feições nos mapas retornados pelas requisições.
java.lang.Stri	**getFeatureInfoXML**(java.lang.String url) Proporcionar aos clientes de um serviço WMS informações adicionais sobre feições nos mapas retornados pelas requisições.
vo	**getMap**(java.lang.String url) Retorna visualização do mapa.
java.lang.Stri	**getMapXML**(java.lang.String url) Retorna dados XML do mapa.

Method Detail

getCapabilities

```
void getCapabilities(java.lang.String url)
```

Tem como objetivo a obtenção de metadados do serviço, ou seja, uma descrição do conteúdo das informações do servidor e valores dos parâmetros de requisição.

Parameters:

`url` - URL da requisição, que deve possuir os seguintes parâmetros:

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=version	O	Número de versão
SERVICE=WMS	M	Tipo de serviço
REQUEST=GetCapabilities	M	Nome da requisição
FORMAT=MIME_type	O	Formato de Saída
UPDATESEQUENCE=string	O	Núm. de seqüência

See Also:

---

getCapabilitiesXML

`java.lang.String` **`getCapabilitiesXML`**`(java.lang.String url)`

Tem como objetivo a obtenção de metadados do serviço, ou seja, uma descrição do conteúdo das informações do servidor e valores dos parâmetros de requisição.

Parameters:

`url` - URL da requisição, que deve possuir os seguintes parâmetros:

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=version	O	Número de versão
SERVICE=WMS	M	Tipo de serviço
REQUEST=GetCapabilities	M	Nome da requisição
FORMAT=MIME_type	O	Formato de Saída
UPDATESEQUENCE=string	O	Núm. de seqüência

Returns:

String do código XML contendo os metadados

See Also:

---

getMap

`void` **`getMap`**`(java.lang.String url)`

Retorna visualização do mapa.

Parameters:

`url` - URL da requisição, que deve possuir os seguintes parâmetros:

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=1.3.0	M	Versão da requisição
REQUEST=GetMap	M	Nome da requisição
LAYERS=layer_list	M	Lista de camadas separadas por vírgula
STYLES=style_list	M	Lista de estilos separados por vírgula, um para cada camada
CRS=namespace:identifier	M	Sistema de coordenadas de referência
BBOX=minx,miny,maxx,maxy	M	Cantos da caixa de contorno do mapa (inferior esquerdo, superior direito)
WIDTH=output_width	M	Largura do mapa em pixels do mapa
HEIGHT=output_height	M	Altura do mapa em pixels do mapa
FORMAT=output_format	M	Formato de saída do mapa
TRANSPARENT=TRUE|FALSE	O	Transparência de fundo do mapa
BGCOLOR=color_value	O	Cor de fundo do mapa em hexadecimal (RGB)
EXCEPTIONS=exception_format	O	Formato em que exceções devem ser reportadas ao WMS (o default é XML)
TIME=time	O	Valor de tempo da camada desejada
ELEVATION=elevation	O	Elevação da camada desejada
Other sample dimension(s)	O	Valor de outras dimensões, se for apropriado

See Also:

---

getMapXML

`java.lang.String` **`getMapXML`**`(java.lang.String url)`

Retorna dados XML do mapa.

Parameters:

`url` – UR L da requisição, que deve possuir os seguintes parâmetros:

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=1.3.0	M	Versão da requisição
REQUEST=GetMap	M	Nome da requisição
LAYERS=layer_list	M	Lista de camadas separadas por vírgula
STYLES=style_list	M	Lista de estilos separados por vírgula, um para cada camada
CRS=namespace:identifier	M	Sistema de coordenadas de referência
BBOX=minx,miny,maxx,maxy	M	Cantos da caixa de contorno do mapa (inferior esquerdo, superior direito)
WIDTH=output_width	M	Largura do mapa em pixels do mapa
HEIGHT=output_height	M	Altura do mapa em pixels do mapa
FORMAT=output_format	M	Formato de saída do mapa
TRANSPARENT=TRUE|FALSE	O	Transparência de fundo do mapa
BGCOLOR=color_value	O	Cor de fundo do mapa em hexadecimal (RGB)
EXCEPTIONS=exception_format	O	Formato em que exceções devem ser reportadas ao WMS (o default é XML)
TIME=time	O	Valor de tempo da camada desejada
ELEVATION=elevation	O	Elevação da camada desejada
Other sample dimension(s)	O	Valor de outras dimensões, se for apropriado

Returns:
String do código XML contendo o mapa
See Also:

---

getFeatureInfo
`void` **`getFeatureInfo`**`(java.lang.String url)`
Proporcionar aos clientes de um serviço WMS informações adicionais sobre feições nos mapas retornados pelas requisições.

Parameters:
`url` - URL da requisição, que deve possuir os seguintes parâmetros:

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=1.3.0	M	Versão da requisição
REQUEST=GetFeatureInfo	M	Nome da requisição
Parte a requisição de Mapa	M	Cópia parcial dos parâmetros de requisição de mapas que geraram o mapa para o qual informação é desejada.
QUERY_LAYERS=layer_list	M	Lista de camadas a serem consultadas, separadas por vírgula
INFO_FORMAT=output_format	M	Formato de retorno da informação acerca da feição.
FEATURE_COUNT=number	O	Número de feições sobre as quais se deseja informação
I=pixel_column	M	Coordenada i, em pixels, da feição no mapa
j=pixel_column	M	Coordenada j. em pixels, da feição no mapa
EXCEPTIONS=exception_format	O	Formato em que exceções devem ser reportadas ao WMS (o default é XML)

See Also:

---

getFeatureInfoXML
`java.lang.String` **`getFeatureInfoXML`**`(java.lang.String url)`
Proporcionar aos clientes de um serviço WMS informações adicionais sobre feições nos mapas retornados pelas requisições.
Parameters:

`url` - URL da requisição, que deve possuir os seguintes parâmetros:

Parâmetro	Obrigatório(M)/Opcional(O)	Descrição
VERSION=1.3.0	M	Versão da requisição
REQUEST=GetFeatureInfo	M	Nome da requisição
Parte a requisição de Mapa	M	Cópia parcial dos parâmetros de requisição de mapas que geraram o mapa para o qual informação é desejada.
QUERY_LAYERS=layer_list	M	Lista de camadas a serem consultadas, separadas por vírgula
INFO_FORMAT=output_format	M	Formato de retorno da informação acerca da feição.
FEATURE_COUNT=number	O	Número de feições sobre as quais se deseja informação
I=pixel_column	M	Coordenada i, em pixels, da feição no mapa
j=pixel_column	M	Coordenada j. em pixels, da feição no mapa
EXCEPTIONS=exception_format	O	Formato em que exceções devem ser reportadas ao WMS (o default é XML)

Returns:
String do código XML/GML
See Also:

Livros Grátis

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

Milhares de Livros para Download:

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