Projeto e Implementação de Interfaces para Sistemas de Aplicações Geográficas

Juliano Lopes de Oliveira

Tese de Doutorado

Projeto e Implementação de Interfaces para Sistemas de Aplicações Geográficas

Este exemplar corresponde à redação final da Tese devidamente corrigida e defendida por Juliano Lopes de Oliveira e aprovada pela Banca Examinadora.

Campinas, 11 de dezembro de 1997.

Claudia Bauzer Medeiros (Orientadora)

Tese apresentada ao Instituto de Computação, UNICAMP, como requisito parcial para a obtenção do título de Doutor em Ciência da Computação.

FICHA CATALOGRÁFICA ELABORADA PELA BIBLIOTECA DO IMECC DA UNICAMP

Oliveira, Juliano Lopes de

OL4p Projeto e implementação de interfaces para sistemas de aplicações geográficas / Juliano Lopes de Oliveira -- Campinas, [S.P. :s.n.], 1997.

Orientador : Claudia Bauzer Medeiros

Tese (doutorado) - Universidade Estadual de Cam Instituto de Computação.

1. Interfaces de usuário (Sistema de computador). 2. Sistemas informações geográficas. 3. Projeto de banco de dados. I. Medeiros, Claudia Bauzer. II. Universidade Estadual de Campinas. Instituto de Computação. III. Título

Instituto de Computação
Universidade Estadual de Campinas

Projeto e Implementação de Interfaces para Sistemas de Aplicações Geográficas

Juliano Lopes de Oliveira[1]

11 de dezembro de 1997

Banca Examinadora:

- Claudia Bauzer Medeiros (Orientadora)
- Alberto Henrique F. Laender
  Departamento de Ciência da Computação – UFMG
- Lia Goldstein Golendziner
  Instituto de Informática – UFRGS
- Hans Kurt E. Liesenberg
  Instituto de Computação – UNICAMP
- Neucimar Jerônimo Leite
  Instituto de Computação – UNICAMP
- Jansle Vieira Rocha
  Faculdade de Engenharia Agrícola – UNICAMP (Suplente)
- Ariadne M. B. Rizzoni Carvalho
  Instituto de Computação – UNICAMP (Suplente)

[1]Este trabalho foi realizado com suporte financeiro do CNPq, da UFG, e dos projetos PROTEM GEOTEC do CNPq e ITDC GEOTOOLS da Comunidade Européia.

Tese de Doutorado defendida e aprovada em 11 de dezembro de 1997 pela Banca Examinadora composta pelos Professores Doutores

Profª. Drª. Lia Goldstein Golendziner

Prof. Dr. Alberto Henrique Frade Laender

Prof. Dr. Hans Kurt Edmund Liesenberg

Prof. Dr. Neucimar Jerônimo Leite

Profª. Drª. Claudia Maria Bauzer Medeiros

Dedicatória

"Naître. mourir. renaître encore.
et progresser sans cesse.
Telle est la loi."

Allan Kardec

Este trabalho é dedicado à minha mãe, Nadir, e ao meu pai, Urbano, que mesmo sofrendo com a minha ausência, compreenderam e apoiaram o meu projeto de vida.

Sumário

Esta tese apresenta um conjunto de técnicas e modelos para suporte ao projeto e implementação de interfaces para sistemas de informação geográfica (SIG). A proposta combina conceitos de três áreas da Computação - Bancos de Dados, Engenharia de Software e Interfaces Homem-Computador - a partir de um enfoque inovador, que trata a interface não apenas do ponto de vista da interação com o usuário, mas também com o resto do sistema subjacente. Os aspectos cobertos incluem tanto a arquitetura da interface quanto mecanismos para sua construção.

A base para a integração interface-SIG é um BD geográfico orientado a objetos. A solução apresentada pode ser mapeada para a maior parte das ferramentas de desenvolvimento de interface existentes.

Os principais resultados da tese são: uma arquitetura de software para projeto e implementação de interfaces em sistemas de aplicações geográficas; um modelo de objetos para construção de interfaces com capacidade de incorporar modificações em tempo de execução (interfaces dinâmicas); um mecanismo para personalização de interfaces baseado em bancos de dados ativos; e a criação de componentes reutilizáveis de interface voltados para o domínio de aplicações geográficas.

As técnicas e modelos propostos nesta tese foram utilizados no projeto e implementação de interfaces para dois sistemas de aplicações geográficas, nas áreas urbana e ambiental. Os resultados desta experiência demonstram que este trabalho contribui para diminuição de custos e aumento da eficiência no desenvolvimento de interfaces geográficas.

Abstract

This thesis presents a framework of techniques and models to support the design and implementation of user-interfaces for geographic information systems (GIS). The proposal combines concepts from three areas of computer science – Databases, Software Engineering and Human-Computer Interfaces – in a innovative perspective, considering interactions not only with the user, but also with the underlying software. The framework covers both the architecture of the interface and the mechanisms for its construction.

The basis of the interface-GIS integration is an object-oriented geographic database. The presented solution can be mapped to most of the existing interface development tools.

The main results of the thesis are: a software architecture for the design and implementation of user-interfaces for geographic applications systems; an interface objects model for building user-interfaces which can be modified at run-time (dynamic interfaces); an interface customization mechanism based on active databases; and the creation of reusable interface components geared towards geographic applications.

The techniques and tools introduced in this thesis were applied on the design and implementation of user-interfaces for two geographic applications systems, in urban and environmental areas. The results of this experience showed that this work contributes to diminishing the costs and improving the efficiency of the development of geographic interfaces.

Agradecimentos

Acima de tudo, agradeço a **Deus** pela dádiva da vida.

A realização deste trabalho não foi uma conquista solitária. Durante toda a jornada contei com o apoio de muitas pessoas com as quais eu compartilho a alegria de terminar este projeto. Gostaria de agradecer, em particular, às seguintes pessoas:

- Meus familiares: sobretudo meus pais e irmãos: me sinto abençoado por ter nascido entre pessoas tão maravilhosas.
- Claudia: sua dedicação e atenção ao meu trabalho foram muito além das obrigações esperadas de uma orientadora. Espero transmitir a outros o que aprendi com você.
- Adriana: sua presença tem sido um estímulo constante para que eu supere minhas próprias limitações.
- Jaqueline e Benoît: me acolheram sem restrições em sua casa, durante os três meses em que trabalhei em Paris.
- Grupo de Bancos de Dados do IC: em nossas reuniões foram consolidadas as idéias desta tese. Agradeço em especial a Marcos Antônio, por implementar estas idéias.
- Fábio Lucena (professor da UFG), Geovane e Cleida (pesquisadores do CPqD): tivemos discussões frutíferas sobre muitas problemas tratados nesta tese.
- Professores Bernd Amann e Michel Scholl: me receberam no CNAM, onde trabalhamos em algumas propostas da tese.
- Colegas de doutorado: Mário, Marcus, e Fátima, por tudo o que passamos juntos.
- Professores e funcionários do IC: aprendi muito com vocês ao longo dos sete anos em que estive na Unicamp.

Agradeço ainda ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), à Universidade Federal de Goiás, e aos projetos PROTEM GEOTEC (CNPq) e ITDC GEOTOOLS (*European Community project 116-82152*), pelo suporte financeiro à execução do Programa de Doutorado.

Conteúdo

Lista de Figuras

Capítulo 1

Motivação e Conceitos Básicos

1.1 Apresentação

Esta tese aborda um conjunto de problemas que envolvem a interseção entre duas áreas de pesquisa em computação: **interfaces de usuário** e **bancos de dados geográficos**. O objetivo é analisar os problemas existentes e propor novas abordagens para facilitar o projeto e implementação de interfaces de usuário para *Sistemas de Informação Geográfica* (SIG). Especificamente, a tese apresenta um arcabouço teórico – mecanismos, modelos, técnicas e ferramentas – validado experimentalmente, propondo extensões às técnicas já existentes para modelagem, projeto e desenvolvimento de interfaces, visando atender aos requisitos particulares de aplicações geográficas.

As propostas apresentadas visam integrar duas perspectivas conflitantes, onde os usuários demandam interfaces cada vez mais especializadas e sofisticadas, enquanto os projetistas que as constroem trabalham de modo praticamente artesanal. O resultado é um conjunto de ferramentas para que o projetista de interface possa lidar com a complexidade de SIG, desenvolvendo sistemas interativos dentro de patamares de custo e tempo de implementação aceitáveis.

As pesquisas atuais freqüentemente incorporam à interface módulos e responsabilidades que já estão disponíveis em Sistemas Gerenciadores de Bancos de Dados (SGBD). Além disto, os proponentes de interfaces para SIG se preocupam em definir a camada de interface somente para consultas, sem se darem conta que a interface também deve interagir em um conjunto de aplicações que irão ser executadas sobre o SIG.

Estas observações estabelecem o enfoque inovador para as contribuições desta tese, as quais estão norteadas por dois princípios que vão de encontro às correntes de pesquisa atuais na área de interfaces para SIG:

- O projeto e arquitetura de uma interface para SIG deve ser realizado levando em consideração a presença do software geográfico subjacente, e separando claramente

as funções que precisam ser deixadas a cargo da interface das demais. Em especial, a interface não deve ser sobrecarregada com tarefas disponíveis no SGBD espacial. O enfoque tradicional, ao contrário, considera a interface isoladamente e, inclusive, duplica funções de SGBD.

◇ Interfaces para SIG não servem apenas para consultas *ad hoc*, devendo também permitir acoplamento de aplicações do usuário. Todos os trabalhos existentes, em contraste, consideram apenas interfaces de consulta.

Dadas essas premissas, as principais contribuições da tese são:

- a especificação de uma arquitetura geral para interfaces de SIG;
- a definição de um modelo de objetos para construção de interfaces com capacidade de incorporar modificações em tempo de execução (interfaces dinâmicas);
- a definição de um mecanismo para adaptação de interfaces aos usuários baseado em bancos de dados ativos; e
- a criação de componentes de interface voltados para o domínio de aplicações geográficas.

Os mecanismos propostos permitem o desenvolvimento de interfaces especializadas para diferentes tipos de aplicações geográficas, facilitando a reutilização de aplicações já existentes na composição de novas especificações.

O restante deste capítulo introduz os conceitos básicos para o entendimento da tese. A próxima seção mostra características de SIG, com ênfase naquelas que influenciam o desenvolvimento de interfaces. A seção 1.3 descreve as dificuldades para projetar e implementar interfaces de SIG. A seção 1.4 resume as propostas desenvolvidas durante as atividades de tese, enquanto a seção 1.5 descreve o conteúdo desta tese. A seção 1.6 sintetiza as idéias deste capítulo.

1.2 Sistemas de Informação Geográfica

Sistemas de Informação Geográfica são ferramentas com funções para gerenciamento e manipulação integrada de dados convencionais e **geo-referenciados**. Um dado geo-referenciado descreve uma entidade geográfica através de três componentes [CCH+96]:

1. *Posição geográfica*: a localização física, na Terra, da entidade geográfica descrita. Permite inferir relacionamentos geométricos e topológicos com outras entidades.

2. *Atributos não-espaciais*: dados convencionais que estabelecem características e propriedades descritivas de cada entidade geográfica;

3. *Tempo*: a data ou período a que se refere o dado geo-referenciado. O tempo pode se referir à data em que o dado tem validade ou à data de sua captura.

Existe uma grande diversidade de fontes e formatos de dados geo-referenciados, tais como: imagens de satélite; dados de censo; fotografias aéreas; dados de GPS (Sistema de Posicionamento Global); e mapas (analógicos ou digitais). Uma vez armazenados em SIG, esses dados se classificam em *espaciais* (descrevem a geometria e a localização de uma entidade geográfica), *convencionais* (valores tradicionalmente manipulados por SGBD) e *pictóricos* (imagens de diversos tipos).

O principal fator que diferencia SIG de outros tipos de sistemas de informação é o conjunto de funções de análise disponíveis, as quais podem utilizar, de modo integrado, todos os componentes da informação geo-referenciada. Dessa forma, um SIG deve permitir ao usuário desenvolver aplicações que lidem com a natureza espaço-temporal dos dados geo-referenciados, considerando a necessidade de discretização de fenômenos geográficos (e os erros decorrentes deste processo).

1.2.1 Arquitetura de SIG

Uma necessidade comum a todos os SIG é a capacidade de prover busca e armazenamento eficiente de informações. A figura 1.1, adaptada de [CCH+96], mostra uma possível arquitetura para SIG, do ponto de vista de blocos funcionais.

As funções de análise espacial são aplicadas sobre dados armazenados no SIG, enquanto as funções de processamento específico englobam várias bibliotecas dedicadas (por exemplo, de processamento de imagens).. As funções de conversão permitem traduzir formatos e padrões dos dados armazenados dentro do SIG de/para outros SIG. As funções de manipulação e armazenamento estão conceitualmente divididas de acordo com o tipo de dado utilizado (convencional ou geo-referenciado, nos formatos vetorial e matricial). No nível de implementação, um componente do SIG armazena os diferentes formatos de dados em páginas físicas de uma unidade de armazenamento. Um SGBD pode ser utilizado para suportar a implementação de grande parte dessa funcionalidade.

Esta tese se concentra no projeto e implementação da camada de software correspondente à interface, e também na sua interação com o SGBD subjacente. Os SIG modernos são projetados para utilizar as facilidades dos SGBD, os quais lhes oferecem, ademais, funções para controle de concorrência, recuperação de falhas, e segurança e integridade de dados. No entanto, SIG não podem utilizar diretamente SGBD convencionais como plataforma de desenvolvimento, devido à existência de tipos de dados e funções de análise

espacial não previstas nestes sistemas. Diversas propostas de arquitetura de SIG tentam resolver este problema, e podem ser classificadas genericamente em:

- Arquitetura *Proprietária*: um SGBD de propósito específico é implementado para atender às necessidades de gerenciamento de informações do SIG.
- Arquitetura *Dual*: um SGBD tradicional gerencia os componentes convencionais das entidades, enquanto arquivos especiais armazenam seus componentes espaciais.
- Arquitetura *Extensível*: a dimensão espacial é incorporada em um SGBD que disponha de mecanismos que permitam estender sua funcionalidade no nível de sistema.

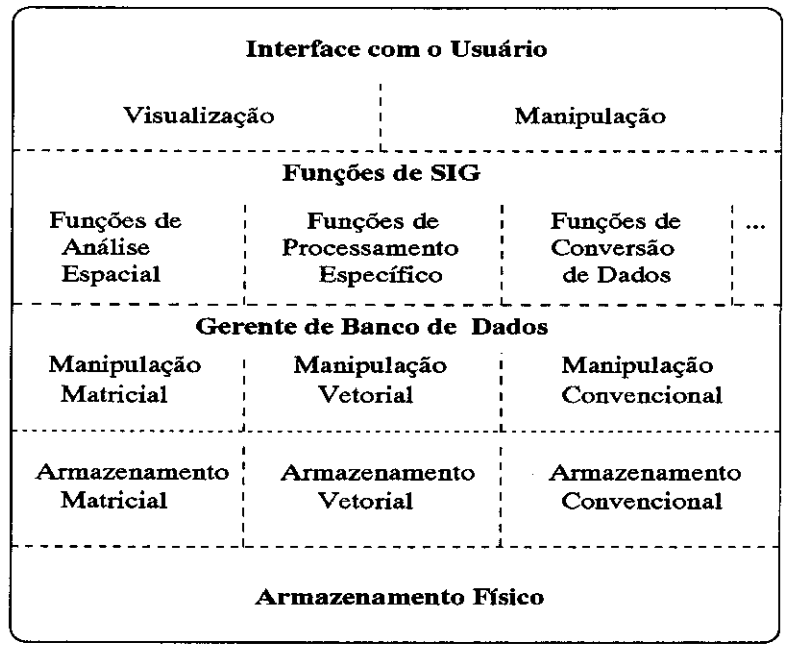

Figura 1.1: Arquitetura funcional genérica de SIG

Cada tipo de arquitetura de SIG delega ao SGBD um determinado subconjunto das funcionalidades apresentadas na figura 1.1. Deste modo, a responsabilidade do SGBD pode se limitar apenas ao armazenamento físico, ou pode se estender a todas as funções básicas do SIG (caso da arquitetura extensível).

Outros aspectos diretamente relacionados com a arquitetura de SIG são o compartilhamento de informações geo-referenciadas e a integração com outros sistemas. A necessidade de compartilhamento e integração de dados entre diferentes SIG é premente, já que o custo de aquisição de dados geo-referenciados é muito elevado. Neste sentido têm surgido propostas para intercâmbio de dados geo-referenciados, tais como SDTS (*Spatial Data Transfer*

Standard) [MW92] e SAIF (*Spatial Archive and Interchange Format*) [Sur94]. Não existe, porém, um padrão *de facto*. Assim, o módulo de conversão de dados deve ser capaz de importar e exportar dados de acordo com as especificações de diversas propostas.

No que diz respeito à integração com outros sistemas, é importante que um SIG tenha a capacidade de interagir com diversos tipos de software, tais como pacotes gráficos e estatísticos, e notadamente outros SIG. Existe uma tendência dos novos SIG em prover esta capacidade de acoplamento, como pode ser observado em [AYA+92, VvO92, GR93, PMP93, VS94]. O consórcio *Open GIS* é um exemplo de esforço conjunto entre usuários e desenvolvedores de recursos de geoprocessamento, incluindo pesquisadores, vendedores, agências governamentais e organizações padronizadoras, com o objetivo de produzir SIG que possuam garantia de interoperabilidade. O consórcio propõe o padrão OGIS (*Open Geodata Interoperability Specification*) que abrange tanto aspectos de desenvolvimento dos SIG (baseado em tecnologia de objetos distribuídos) quanto a definição de um modelo de dados espacial padronizado (baseado em SAIF e compatível com SDTS) [Ope96].

1.2.2 Funcionalidade de SIG

O conjunto de funções oferecidas por SIG é amplo, não havendo um consenso sobre todas as funções que devem estar presentes [MGR91a, MD91, RM92, Wor95]. Diferentes tipos de funções são usadas em aplicações específicas, como por exemplo as funções de processamento de imagem em aplicações ambientais, ou as funções de bancos de dados em sistemas AM/FM (*Automated Mapping and Facilities Management*).

Apesar dessa variedade de funções, existe um conjunto básico de funcionalidades que devem estar presentes em SIG. Uma taxonomia destas funções foi apresentada em [OPM97], segundo a perspectiva de interface de usuário:

1. **Entrada e Conversão**: compreende o conjunto de funções que precisam ser aplicadas *antes* da efetiva utilização dos dados no SIG. Essas funções se caracterizam pelo uso intensivo de recursos computacionais (processamento e entrada/saída), e podem introduzir grandes volumes de dados no sistema.

   Dentro desse conjunto de funções se destacam a captura de dados de diferentes fontes; a transferência de dados previamente capturados para a forma digital adotada pelo SIG; e a validação e tratamento de erros inerentes ao processo de captura de dados geo-referenciados;

2. **Modelagem**: são as funções que definem os conceitos e abstrações importantes para cada tipo de aplicação. Permitem a especificação não apenas dos dados (modelagem de dados), mas também dos processos (modelagem de processos) utilizados nas aplicações. Desta forma, estas funções determinam a organização, integração e

estruturação lógica das informações capturadas, de acordo com as necessidades de aplicações específicas.

Funções de modelagem incluem seleção, associação, generalização, especialização e classificação de dados, e definição de modelos matemáticos que atuam sobre estes dados segundo os objetivos da aplicação. Em geral, estas funções não estão disponíveis em SIG comerciais, ficando sua definição e uso a cargo de ferramentas externas ao sistema, ou embutidas nas aplicações.

3. **Recuperação e Análise**: são as funções que tratam da busca, manipulação e transformação integrada de dados convencionais e geo-referenciados. Elas caracterizam a principal diferença entre SIG e outros sistemas de informação.

   Dentre as funções neste grupo se destacam a navegação, consulta, seleção, e pesquisa de informações geográficas; as transformações efetuadas para adequar os dados aos diferentes tipos de análise possíveis (mudança de escala ou projeção, por exemplo); a identificação de relacionamentos, padrões e variações de entidades espaciais; e a extração de estatísticas e medições geométricas.

4. **Apresentação**: neste grupo estão as funções que transformam a saída dos resultados de consultas e funções de análise para formatos que podem ser mais facilmente interpretados pelo usuário. Estas funções permitem que um mesmo conjunto de resultados seja visualizado de diferentes maneiras, de acordo com o perfil do usuário. Funções de apresentação típicas incluem a produção de mapas, relatórios, diagramas, gráficos e imagens.

A complexidade das funções de entrada e conversão de dados em SIG é agravada pela diversidade de fontes e formatos de dados geo-referenciados. Estas funções precedem o efetivo uso dos dados nos SIG, e apresentam, do ponto de vista de interface, características que diferem profundamente das presentes nas demais classes de funções. A entrada de dados envolve, em geral, a interação com diversos dispositivos de entrada, tais como mesas digitalizadoras, *scanners*, e dispositivos de sensoriamento remoto. Por apresentarem características e problemas tão divergentes das demais, as funções de entrada não serão abordadas nesta tese. Os problemas de interface estudados partem da hipótese de que os dados já foram capturados e convertidos para o formato apropriado do SIG.

1.2.3 Modelos de Dados em SIG

Além da complexidade funcional, SIG se caracterizam pela diversidade de modelos de dados espaciais que comportam. Esta característica pode ser compreendida através da comparação com SGBD relacionais. No modelo relacional existe um único conceito (a *relação*)

que estabelece a estrutura organizacional dos dados, bem como as operações possíveis sobre essa estrutura. Em SIG não existe um modelo de dados consensual; pelo contrário, cada sistema implementa seu próprio modelo de dados. Desta forma, os modelos de dados geográficos existentes na atualidade diferem quanto à *estrutura* (representando tipos de conceitos e relacionamentos distintos) e quanto ao *comportamento* (implementando diferentes conjuntos de operações, com semânticas conflitantes).

Dentro da diversidade de modelos de dados existentes em SIG, existe uma clara dicotomia entre representações (ou visões) distintas da realidade geográfica [Cou92, FG90]:

- ◇ Visão de **campo**: considera o espaço geográfico como uma superfície sobre a qual diferentes fenômenos são caracterizados por valores distribuídos continuamente. Cada fenômeno é caracterizado por um único valor em cada ponto do espaço descrito. Dessa forma, um campo pode ser representado por uma função associada a cada ponto do espaço, mapeando características de um fenômeno em valores que as representam. Diferentes fenômenos são representados por diferentes funções, e os campos resultantes podem ser considerados como camadas temáticas que descrevem aspectos específicos da realidade geográfica original.

- ◇ Visão de **objeto**: captura a realidade geográfica através de objetos com características bem definidas, entre as quais estão a sua localização e a sua geometria. Cada objeto possui uma identidade que o diferencia de todos os demais objetos, ocupa determinada posição do espaço descrito, e possui valores para um conjunto de características que ocorrem na sua localização (mas não necessariamente se referem a fenômenos geográficos). Dois objetos podem se sobrepor no espaço, de modo que a cada ponto podem estar associados diferentes valores para uma determinada característica.

Em geral, a visão de campo representa melhor os fenômenos geográficos naturais (como vegetação, solo, e clima), enquanto a visão de objetos é mais adequada para representar artefatos humanos geo-referenciados (como edificações e redes de infraestrutura urbana). O modelo de dados é o aspecto isolado de SIG mais importante para a interface de usuário. A capacidade do modelo em representar conceitos familiares ao usuário é um fator determinante na complexidade dos mapeamentos realizados pela interface.

1.2.4 Aplicações de SIG

Uma **aplicação geográfica** é um programa que manipula dados geo-referenciados armazenados em um SIG. [ABC+91] e [MGR91b] apresentam coletâneas de exemplos que revelam a diversidade de empregos possíveis e a importância econômica deste tipo de

aplicação. As aplicações geográficas definem requisitos de informação que permitem classificá-las em três categorias:

1. Aplicações *urbanas* ou *sócio-econômicas*: voltadas para aspectos de infraestrutura urbana, de controle populacional, e de administração de propriedades. Exemplos englobam (i) gerência de redes (água, energia, telecomunicações, transportes); (ii) localização e distribuição de serviços públicos (hospitais, escolas); e (iii) controle de censo, análise mercadológica, e administração de impostos sobre propriedades.

2. Aplicações *ambientais*: voltadas para o aproveitamento e conservação de recursos naturais. Exemplos incluem (i) modelagem da natureza (estudos climáticos, controle de agentes poluidores, análise de processos de desertificação e de destruição de coberturas vegetais); (ii) previsão, detecção, e prevenção de cataclismos e fenômenos devastadores (terremotos, furacões); e (iii) inferência de fatores que originam fenômenos naturais (ocorrência de minérios, mudanças no ecossistema).

3. Aplicações *Gerenciais*: envolvem informações qualitativas sobre aplicações ambientais e urbanas. O objetivo é apoiar a formulação e o acompanhamento de políticas de desenvolvimento urbano e de uso de recursos naturais. Aplicações típicas envolvem tomada de decisão baseada em informações geo-referenciadas.

Aplicação	Consulta Típica
Urbana	O que se encontra no local ...? O que está adjacente a ...? Qual o caminho para ...?
Ambiental	Onde ocorre o fenomeno ...? O que mudou desde ...? Onde as mudanças ocorreram?
Gerencial	Que padrões espaciais existem? Quais as anomalias neste padrão? O que ocorre se ...?

Figura 1.2: Consultas típicas em diferentes categorias de aplicações geográficas

A figura 1.2 exemplifica, através de questões simples e genéricas, a natureza dos requisitos e os aspectos mais importantes da informação geo-referenciada de acordo com os diferentes tipos de aplicação.

Além dos aspectos aqui mencionados (arquiteturas, modelos de dados, funcionalidades, e aplicações), SIG possuem um vasto conjunto de características e particularidades que não foram abordadas por fugirem ao objetivo deste trabalho. O leitor interessado

em discussões sobre outros aspectos de SIG (tais como projeções cartográficas, qualidade de informação, integração de dados, estruturas de armazenamento e indexação espacial) encontrará subsídios adequados em [CCH+96, SVP+96, Wor95, MGR91a].

1.3 Interfaces de Usuário para SIG

A interação entre usuário e sistema computacional abrange, por um lado, aspectos técnicos de processamento de dados e, por outro lado, questões relacionadas ao comportamento humano na realização de uma tarefa. Um **sistema de interface usuário–computador** é uma ferramenta que estabelece um meio de ligação entre estes dois aspectos da interação.

Apesar de notáveis avanços, o uso efetivo de SIG ainda não está suficientemente disseminado. De fato, um conjunto muito restrito de usuários tem se beneficiado com o progresso da tecnologia de geoprocessamento. Um fator chave para explicar essa subutilização do potencial de SIG é, sem dúvida, a *interface com o usuário*.

Em geral, o sistema de interface implementa um filtro de conversão sobre os fluxos de informação entre computador e usuário, facilitando o processo de comunicação entre eles. Neste sentido, avanços significativos já foram obtidos no desenvolvimento de padrões de projeto de interfaces [Shn87, Sun90, Ope91], na construção de modelos que representam a interação entre homem e computador [MD91, Nor91, GG96], e na implementação de sistemas gerenciadores de interface [HH89, Mra91, Mye95].

1.3.1 Características de Interfaces para SIG

Do ponto de vista da interface, pode-se considerar um SIG como um SGBD sofisticado, com capacidade de gerenciar dados geo-referenciados e de executar funções de análise integrada sobre estes dados. De acordo com esta visão, a interface de SIG deve prover todas as qualidades desejáveis de interfaces de SGBD, incluindo aspectos de transparência, adaptabilidade, geração automática de apresentações, com suporte a diferentes níveis de abstração, e independência entre interface e SGBD [Wel88, Oli94].

A dificuldade para garantir estas propriedades de interface é consideravelmente maior em SIG do que em SGBD, devido às características daqueles sistemas. Por exemplo, a geração de apresentações é um problema simples em SGBD relacionais, mas não trivial em SGBD orientados a objetos [MM91, OA93a]. Em SIG, a complexidade desta tarefa é ainda maior, pois envolve a transformação de informações alfanuméricas em estruturas gráficas apropriadas para a produção de resultados cartográficos.

Além dessas características comuns com interfaces de SGBD, as interfaces de SIG possuem particularidades que as diferenciam de outros tipos de interface, e que contribuem

para a complexidade de sua implementação, dentre as quais se destacam:

1. *Ausência de metáforas universais*: não existe um paradigma de apresentação e manipulação de informações em SIG que represente um consenso geral, como o que representa a metáfora de *tabela* em interfaces para SGBD relacionais. A metáfora de *mapa* geralmente adotada não tem o mesmo grau de padronização em termos de definição de dados e operações. Conseqüentemente, as interfaces de propósito geral em SIG possuem um potencial bastante restrito.

2. *Complexidade funcional*: SIG modernos disponibilizam centenas de operações aos seus usuários. Uma abordagem comum para suportar este conjunto extensível de operações adota a modularização funcional da interface, criando os chamados "*modos de operação*". Operações de interface semelhantes podem levar a resultados bastante diferentes, dependendo do modo de operação em que forem executadas.

3. *Múltiplas representações*: a complexidade do espaço geográfico demanda o armazenamento de diferentes *representações* da mesma entidade geográfica. Por exemplo, uma propriedade rural pode ser vista pela perspectiva de um mapa que descreve classes de uso de solo, ou através de um conjunto de polígonos que definem os limites e a extensão espacial da propriedade. A interface deve permitir associar operações sobre diferentes abstrações de uma mesma entidade real às suas respectivas representações armazenadas nos bancos de dados.

4. *Modelo de dados intermediário*: os modelos de dados de SIG estão direcionados aos aspectos de implementação, tais como a descrição de estruturas de dados e índices espaciais, e a manipulação eficiente de geometrias e relacionamentos topológicos. Interfaces para SIG necessitam de um modelo intermediário para mapear *representações* espaciais descritas pelos modelos de dados dos SIG para *conceitos* espaciais compreendidos pelo usuário final. O modelo intermediário da interface deve ser semanticamente próximo do modelo mental do usuário, mantendo um mapeamento eficiente para as estruturas das representações usadas pelos SIG.

5. *Manipulação gráfica e textual*: interfaces de SIG precisam prover visualização espacial das informações armazenadas textualmente através de recursos gráficos (não faz sentido apresentar o conjunto de coordenadas X e Y de um polígono, por exemplo). A elaboração de consultas pode ser feita através de diagramas que representam os objetos espaciais, ou através de valores alfanuméricos para medidas e relacionamentos espaciais (tais como distância ou adjacência).

Prover as características identificadas acima não é uma tarefa trivial. O projeto e a implementação de interfaces para sistemas complexos, como os que envolvem aplicações de SIG, pode consumir grande parte dos recursos destinados ao sistema como um todo.

1.3.2 Dificuldades de Projeto e Implementação

Interfaces para SIG enfrentam todos os problemas existentes para implementação de interfaces para SGBD. Além disso precisam tratar uma série de dificuldades oriundas das características de SIG e do uso de dados geo-referenciados. As principais dificuldades enfrentadas pelos desenvolvedores de interface são apresentadas a seguir.

A ligação entre interface e SIG Existem duas abordagens básicas para esse problema. Na primeira, o sistema de interface faz parte do SIG, e se beneficia de conhecimentos sobre estruturas físicas de armazenamento. Na segunda abordagem, o sistema de interface é um módulo externo que se comunica com o SIG, sendo independente de sua implementação. A escolha de uma abordagem deve ser uma solução de compromisso entre *independência entre interface e* SIG e *desempenho e simplicidade da interface* [Voi94].

A visualização de dados geo-referenciados Esta é uma área de pesquisa por si própria, que trata da transformação dos dados armazenados em objetos gráficos que apresentam a informação espacial para o usuário. Esta transformação deve resolver problemas de cartografia inerentes ao projeto de mapas, tais como o uso de escalas e projeções, e também a *generalização cartográfica* [LR93] e *abstrata* [Rig95] de entidades espaciais.

O aspecto temporal Todos os problemas de visualização de dados espaciais tornam-se mais complexos quando o componente temporal da informação geo-referenciada está envolvido. A visualização da evolução de uma dada entidade geográfica ao longo do tempo é, em geral, restrita a mapas estáticos, referentes a um único ponto ou intervalo de tempo, apresentados em diferentes janelas da tela. Técnicas de animação de imagens vêm sendo exploradas para melhorar a visualização da variação temporal dos dados espaciais, mas ainda há muito trabalho a ser feito nesta área [Peu94].

O modelo mental do usuário O projeto de interfaces se baseia em suposições feitas sobre os seus potenciais usuários, expressas através de um *modelo mental de usuário*. Os problemas para definição de modelos mentais em SIG englobam desde a definição de conceitos espaciais básicos até a proposta de metáforas adequadas para representação desses conceitos. Um agravante para estes problemas é que a comunidade de usuários de SIG é formada por especialistas em diferentes áreas de aplicação de geoprocessamento, mas que raramente possuem conhecimentos suficientes de computação [MSH93].

A diversidade de linguagens e álgebras espaciais Existe uma grande dificuldade para adaptação de uma interface a diferentes SIG, pois cada sistema especifica e implementa suas próprias linguagens de definição e manipulação de dados. Mais ainda, a

diferença das linguagens não se restringe aos aspectos sintáticos, mas inclui também a semântica de operações. Cada SIG define operações próprias, e mesmo que dois sistemas ofereçam a mesma operação espacial, não há garantia de que sua implementação (e semântica) seja a mesma nos dois SIG [BM92].

A adaptação da interface a diferentes contextos O processo de adaptação da interface a um contexto de utilização será designado nesta tese pelo termo *personalização*. *Personalizar* uma interface existente para atender diferentes tipos de usuários representa uma tentativa de reduzir os custos de implementação da interface. Entretanto, as técnicas existentes para personalização são insuficientes para proporcionar um ganho real em termos de esforço de desenvolvimento e de facilidade de uso da interface [LS92, Gou93].

1.4 Principais Propostas da Tese

As propostas desta tese contribuem para a resolução de alguns dos problemas apontados na seção anterior. Deve-se notar que muitos desses problemas são multi-disciplinares, envolvendo diversas áreas de estudo, tais como psicologia, educação, ergonometria, lingüística e computação. Esta tese se concentra nos problemas diretamente relacionados com a Ciência da Computação e, em particular, nos aspectos que envolvem Interfaces de Usuário, Bancos de Dados, e Engenharia de Software.

Dentro deste contexto, a tese introduz novas abordagens para os problemas de concepção e implementação de interfaces em um **sistema de aplicações geográficas** (SAG). Um SAG é um conjunto de aplicações geográficas inter-relacionadas, implementadas sobre um SIG, e que cooperam para realizar tarefas complexas. A interface geográfica de um SAG é formada pelo conjunto de componentes interativos de suas aplicações. Um conjunto de aplicações geográficas que permitem gerenciar redes de telecomunicações é um exemplo de SAG, conforme discute a seção 7.2.

A figura 1.3 mostra de forma esquemática o contexto em que as propostas da tese estão situadas. Os principais resultados da tese no sentido de suportar o desenvolvimento de interfaces para SAG são:

- A especificação de uma *arquitetura de interface* [OM95] que organiza os componentes do sistema de interface em camadas, definindo suas funcionalidades, interoperabilidade, e ligação com o SIG (e demais softwares de suporte). O objetivo é organizar o software de interface de modo a facilitar o projeto, implementação e manutenção dos componentes interativos dos SAG.

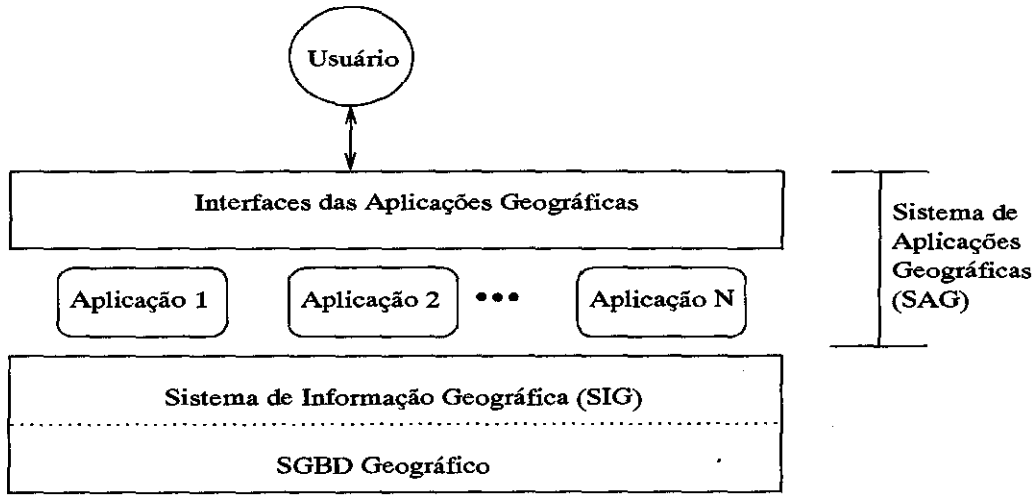

Figura 1.3: Sistema de Aplicações Geográficas (SAG)

◇ Um *modelo de objetos* para construção de *interfaces dinâmicas*. O modelo é uma extensão da proposta de [OCM95] e permite a especificação de componentes de interação da camada mais alta da arquitetura da interface. Estes componentes variam em forma e comportamento de acordo com o tipo de aplicação, de modo que a camada precisa ser reestruturada para atender os requisitos de uma aplicação específica. O modelo facilita esta reestruturação pela capacidade de definição de componentes de interface em tempo de execução.

◇ Um *modelo de dados intermediário* para interfaces geográficas [OCM97, OPM97]. Uma das maiores dificuldades para construção do modelo intermediário está no mapeamento do modelo conceitual do usuário para o modelo físico adotado no SIG. A solução proposta para esse problema utiliza uma analogia com o conceito de *visões* em bancos de dados para diminuir a distância semântica entre o modelo mental do usuário e o modelo de dados do SIG.

◇ Um *mecanismo de personalização* de interface baseado no paradigma de bancos de dados ativo [OMC97]. O mecanismo permite adaptar uma interface genérica, construída com base na arquitetura e no modelo de objetos propostos, para necessidades específicas de um conjunto abrangente de aplicações, levando em conta as preferências do usuário e os requisitos da aplicação.

◇ Uma *proposta de interface* para uma ferramenta de projeto de aplicações ambientais [OPM97]. Esta interface facilita o acesso do usuário aos dados existentes, provendo ainda a capacidade de definição de informações específicas para a aplicação projetada. A arquitetura de interface e o modelo de objetos são utilizados para

implementar uma interface com estas características.

Estas propostas são validadas através da sua utilização no projeto e implementação de interfaces para dois tipos de aplicação geográfica, nas áreas de infraestrutura urbana e de planejamento e controle ambiental. Uma contribuição adicional do trabalho de tese é a revisão detalhada (capítulo 3) das atividades de pesquisa recentes em interfaces para SIG [OM96a, OM96b].

1.5 Conteúdo da Tese

Este capítulo introduziu os problemas abordados, oferecendo uma visão geral das propostas apresentadas pela tese. O capítulo discutiu ainda as características básicas dos SIG, enfatizando o seu relacionamento com a interface de usuário. O restante do texto está organizado como se segue.

O capítulo 2 analisa as principais arquiteturas de interface na literatura, que são referenciadas no resto da tese para comparação com as propostas apresentadas.

O capítulo 3 apresenta o *estado da arte* em pesquisas relacionadas a interfaces para SIG. As principais linhas de pesquisa são identificadas e, dentro de cada linha, é feita uma análise comparativa das principais propostas, visando a identificação de aspectos positivos e de deficiências nas abordagens atuais.

O capítulo 4 introduz uma *nova abordagem para projeto* de um sistema de interface para SIG. Esta abordagem está baseada em uma arquitetura de interface organizada em camadas que abstraem os problemas de implementação subjacentes. Cada camada trata de problemas específicos, utilizando os serviços oferecidos pelas demais camadas.

O capítulo 5 apresenta um *modelo de objetos para construção de interfaces* visuais dinâmicas em SIG. O modelo está associado a uma metodologia de desenvolvimento de interface, cujo objetivo é direcionar os esforços para a implementação da camada superior da arquitetura proposta no capítulo 4.

O capítulo 6 propõe a utilização de *mecanismos de bancos de dados ativos* para auxiliar o desenvolvimento e a personalização de interfaces para SIG. A idéia é trazer parte do trabalho de construção da interface para o sistema de bancos de dados subjacente.

O capítulo 7 descreve a *utilização das propostas* apresentadas nesta tese para a implementação de dois sistemas de interface para SIG. O primeiro sistema faz parte de uma aplicação geográfica de grande porte na área de redes de telecomunicações. A segunda interface foi incorporada a um protótipo que suporta o projeto de aplicações ambientais.

O último capítulo apresenta as *conclusões* obtidas do trabalho realizado, através de uma análise das propostas e da comparação com outras abordagens existentes. O capítulo termina com uma discussão das contribuições alcançadas e com sugestões para a extensão deste trabalho.

1.6 Resumo

Este capítulo apresentou a motivação, escopo e objetivos da tese, que se concentra na interseção entre as áreas de interface de usuário e de bancos de dados geográficos. O principal objetivo é a definição de mecanismos para reduzir os custos de projeto e desenvolvimento de interfaces em sistemas de aplicações geográficas baseados em SIG.

Foram definidos conceitos que serão usados nos capítulos seguintes, através de uma introdução a SIG e às características de suas interfaces. Os principais conceitos discutidos incluem:

- Sistemas geográficos: componentes de um dado geo-referenciado; tipos de arquitetura e funcionalidade de SIG; modelos de dados espaciais (visões de campos e de objetos); e aplicações geográficas (urbanas, ambientais e gerenciais);
- Sistemas de interface: características marcantes de interfaces para sistemas de aplicações geográficas.

O capítulo discutiu, ainda, as dificuldades para desenvolvimento de interfaces para SIG, onde se destacam problemas relacionados à ligação entre interface e SIG, à representação múltipla de dados espaciais, e aos modelos mentais de usuários.

Capítulo 2

Organização e Arquitetura do Software de Interface

2.1 Apresentação

A arquitetura, organização e implementação do software de interface é, atualmente, uma importante área de pesquisa em Computação. Este capítulo apresenta uma descrição resumida das principais arquiteturas existentes, que serão cotejadas com as propostas da tese nos capítulos 4, 5, e 6. A seção 2.2 discute critérios para classificação de sistemas de interface. A seção 2.3 analisa a organização e composição do software de interface que será utilizada como referência na tese. A seção 2.4 descreve algumas das principais arquiteturas de software de interface propostas na literatura. A seção 2.5 resume o capítulo.

2.2 Taxonomia de Interfaces

Sistemas de interface podem ser classificados segundo sua complexidade, interatividade, e estilo de interação. A *complexidade* de uma interface pode ser medida pela riqueza (características, variedade, volume, tipo) das informações fornecidas pelo sistema ao usuário, e pela forma com que este expressa as funções e informações desejadas do sistema. É tarefa do projetista de interface prover uma grande variedade de funções (que garantam a utilidade da interface) sem aumentar demasiadamente a complexidade de aprendizagem e de uso do sistema.

A *interatividade* de uma interface é medida pela taxa de informações intercambiadas entre usuário e interface, levando-se em conta, ainda, a granularidade das informações comunicadas. Uma interface com baixa interatividade pode requerer conjuntos completos de comandos antes de qualquer processamento, enquanto uma com alta interatividade interpreta e processa comandos à medida que eles são informados. Interfaces gráficas pos-

suem, em geral, alta interatividade, ao passo que interfaces textuais tendem a apresentar baixa interatividade.

Estilo	Características, Vantagens e Desvantagens
Linguagem Textual	diálogo se dá através de comandos textuais digitados pelo usuário; oferece acesso rápido e eficiente para usuários experientes; permite a realização de operações complexas através de uma seqüência de comandos relativamente pequena; para usuários novatos requer esforço de prática e memorização.
Tela Formatada	o usuário se restringe a completar campos livres de telas pré-formatadas; utilizado tipicamente em aplicações de entrada de dados, e em manipulação de informações comerciais; diálogo de natureza extremamente limitada; vantajoso na interação com usuários novatos ou casuais.
Sistema de Menu	um conjunto de opções é apresentado para o usuário, e a seleção de uma opção modifica o estado da interface; elimina a necessidade de memorização de comandos; guia usuários inexperientes, reduzindo a possibilidade de erros; usuários experientes devem se submeter à navegação repetitiva; utiliza mais espaço de tela do que o necessário para digitar comandos.
Manipulação Direta	baseado na metáfora de movimentação de objetos gráficos; realiza operações complexas com um reduzido conjunto de ações; a metáfora substitui a memorização da sintaxe de comandos; apresentação de opções na tela (representadas pelos objetos gráficos); o fluxo de controle não se limita a uma única ação por vez; dificuldade para obter um modelo gráfico que represente a tarefa; um novo ícone pode ser difícil de assimilar; o usuário pode tirar conclusões erroneas sobre a representação analógica e sobre as operações possíveis no modelo.

Figura 2.1: Os principais estilos de interação em interfaces

O *estilo de interação* (ou *tipo de diálogo*) classifica interfaces segundo a maneira como as informações fluem entre usuário e sistema computacional. A maioria das interfaces adota estilos de interação baseados nas abordagens descritas na figura 2.1, embora existam propostas que empregam outros paradigmas de interação (sistemas de reconhecimento de voz, por exemplo). Cabe observar que a maior parte das interfaces utiliza um estilo híbrido de interação, combinando propriedades de diferentes tipos de diálogo.

2.3 Organização do Software de Interface

Um **programa interativo** se caracteriza por realizar intercâmbio de informações com o usuário durante o seu processamento. Este tipo de programa possui dois componentes lógicos: o **componente semântico**, que define a funcionalidade e a semântica da aplicação (em termos de transformação e manipulação de informações), e o **componente interativo**, responsável pelo diálogo com o usuário (através da apresentação de informações e de mecanismos de entrada de dados).

A interação entre os componentes interativo e semântico se dá, no nível lógico, através de um **protocolo de comunicação** que estabelece, entre outras coisas, o conjunto de operações disponíveis, o modo de interação (síncrono ou assíncrono), os tipos de dados intercambiados (simples ou complexos), e o mapeamento entre objetos (de interação e do domínio semântico).

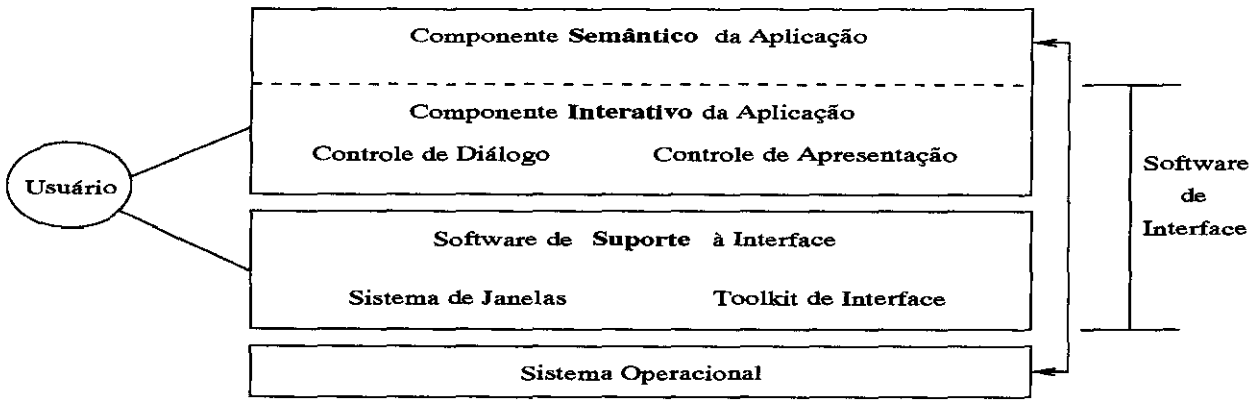

Figura 2.2: Organização do software de um programa interativo

A figura 2.2 apresenta uma divisão lógica do software envolvido em um programa de aplicação interativo de acordo com as funções executadas. Essa organização reflete os produtos oferecidos pelas ferramentas de interface atualmente disponíveis, definindo dois módulos para o software de interface propriamente dito: o componente interativo da aplicação e o software de suporte à interface.

2.3.1 Software de Suporte à Interface

O software de suporte à interface é formado por um conjunto de ferramentas que provê facilidades genéricas, utilizadas em praticamente todos os tipos de interface. Entre estas ferramentas destacam-se os sistemas de janelas e os *toolkits* de interface.

Sistemas de Janelas

Um **sistema de janelas** interage diretamente com o sistema operacional do computador para permitir que a tela seja dividida em regiões distintas (as janelas). Existem dois componentes principais em um sistema de janelas: a *camada básica*, que implementa a funcionalidade do sistema, e o *gerente de janelas* (ou *window manager*), que cuida dos seus aspectos interativos. Em ambientes Unix, o sistema *X windows* é o padrão *de facto* para a camada básica, suportando diferentes tipos de gerente de janelas (Motif Window Manager e OpenLook Window Manager, por exemplo).

A camada básica oferece uma interface procedimental que permite que as aplicações desenhem gráficos em cada janela, e também recebam eventos do usuário. O gerente de janelas define o modelo de apresentação das janelas (por exemplo, se elas podem ser sobrepostas) e também o modo de interação com o usuário, que pode mover, modificar o tamanho, ou destruir janelas, sob a coordenação do sistema de janelas.

Apesar de sua utilidade, os sistemas de janelas oferecem apenas serviços de criação e controle de eventos básicos (mudança de tamanho e de posição, por exemplo) sobre janelas. Os componentes de interação contidos em cada janela são criados através de *toolkits gráficos* de interface que utilizam as facilidades e serviços do sistema de janelas.

Toolkits de Interface

Um **toolkit de interface** pode ser entendido como uma biblioteca de classes de *window gadgets* (**widgets**) que podem ser chamadas dentro de um programa de aplicação. Cada classe de *widget* define um determinado tipo de componente de interface capaz de armazenar e apresentar valores de um certo tipo de dado, e de receber eventos de um dispositivo de entrada (em geral teclado ou *mouse*). Existem diversos *toolkits* de interface, tais como MFC, Xview, e Motif, oferecendo diferentes tipos de *widgets*. Exemplos típicos são as classes que definem botões, menus e campos textuais.

Em geral, *toolkits* permitem a ligação entre *widgets* e programas de aplicação através de funções *callback*. Estas funções são definidas pelo programador e chamadas pelo componente gerenciador de execução de interface do *toolkit* quando ocorre um evento sobre um determinado *widget*. Cada classe de *widgets* especifica um conjunto de eventos que podem ser detectados por suas instâncias e associados a funções *callback*.

Um *toolkit* pode ser implementado sobre diferentes sistemas de janelas, e um sistema de janelas pode suportar diferentes *toolkits* gráficos. O sistema de janelas *X* (ou *X windowing system*) é o padrão *de facto* em estações de trabalho atuais. Além de suportar diversos *toolkits*, como XView e Motif, o sistema *X* permite também a utilização de diferentes gerentes de janela (Motif e OpenLook, por exemplo).

Ferramentas de Alto Nível

Construir uma interface gráfica utilizando somente os procedimentos oferecidos pela camada básica do sistema de janelas é uma tarefa complexa. Mesmo com o auxílio de *toolkits*, existem certas aplicações cujas interfaces gráficas não podem ser devidamente projetadas. Interfaces que variam dinamicamente e aquelas que fazem uso intensivo de desenhos gráficos são exemplos onde *toolkits* não podem contribuir de modo efetivo.

A principal tarefa de uma **ferramenta de alto nível** é suprir esta deficiência, ajudando o projetista na especificação de uma interface complexa. As ferramentas existentes utilizam quatro abordagens para realizar esta tarefa [Mye95]:

1. *Estruturas de aplicação predefinidas*, que definem classes de objetos para as partes importantes da interface de uma aplicação alvo;
2. *Ferramentas de geração automática baseada em modelo*, utilizando modelos de alto nível que descrevem a estrutura das informações e a semântica da aplicação;
3. *Linguagens de definição de interface*, abrangendo variações de diagramas de estados e diversos tipos de linguagens declarativas, de restrições, de eventos, e de programação visual;
4. *Editores para projeto interativo de interface*, que permitem definição parcial da interface através de manipulação direta de *widgets*.

As propostas apresentadas nos capítulos 4, 5 e 6 desta tese implementam mecanismos e modelos que combinam variações das três primeiras abordagens citadas acima.

2.3.2 Componente Interativo da Aplicação

O componente interativo de uma aplicação é formado pelos módulos de **Controle de Diálogo** e de **Controle de Apresentação**, que definem, respectivamente, o comportamento e a aparência específica da interface. Desta forma, o chamado *"look-and-feel"* de uma interface é parcialmente definido pelo gerente de janelas (forma de apresentação de títulos de janelas, bordas, rodapés, cores e padrões de fundo), pelo *toolkit* utilizado para desenvolver a interface (aparência e funcionalidade de cada *widget*), e pelo componente interativo da aplicação (leiaute de *widgets* e funções *callback*).

Controle de Apresentação

Um **objeto de interação** é uma instância de uma determinada classe de *widgets*. Um *botão simples* é um exemplo típico de instância de *widget*: ele pode apresentar um valor

ao usuário (o rótulo do botão) e permite a chamada de uma função da interface (*callback*) em resposta a um evento de ativação com o *mouse*. Cada evento permitido em um objeto de interação pode estar associado a uma ou mais funções *callback*. No exemplo anterior, a função a ser chamada quando o botão é ativado é a função *callback* daquele botão.

Cabe ao projetista de interface escolher os *widgets* apropriados para as finalidades de cada aplicação, e organizá-los em um leiaute que facilite a execução da tarefa alvo. Estas decisões são implementadas no módulo de *Controle de Apresentação*. Os *widgets* usados em uma interface podem estar associados entre si; um **objeto de interface** representa um conjunto de objetos de interação inter-relacionados. É importante observar que um objeto de interação corresponde diretamente a um *widget*, enquanto um objeto de interface pode não possuir um objeto correspondente em *toolkits* de interface. Além disto, objetos de interface definem os controles necessários para a composição de objetos de interação, a fim de construir um objeto de interface complexo. A diferença entre objeto de interação e objeto de interface é fundamental para o entendimento das propostas desta tese, notadamente do capítulo 5.

Controle de Diálogo

O módulo de *Controle de Diálogo* é responsável pela sintaxe de interação com o usuário e pelas conversões entre elementos do domínio da interface e da aplicação. O diálogo com o usuário e com o componente semântico da aplicação é determinado parcialmente pelo comportamento associado a cada objeto de interação (funções *callback*). O Controle de Diálogo é ainda responsável por manter informações sobre o "estado" da interface, e utiliza estas informações para determinar as seqüências de ações necessárias para a ativação de uma função da interface em cada contexto. A ativação da função pode, por sua vez, modificar o estado do sistema.

Desta forma, o controle de diálogo pode ser expresso através de uma *máquina de estados*. O projetista de interface conta com diversos formalismos para definição deste controle, entre os quais se destacam os baseados em *Statecharts* (diagramas de estado) [Har87]. Um diagrama de estado é um grafo onde os nós representam estados do sistema e arestas são eventos de transição entre estados. A implementação destes diagramas é feita no módulo de Controle de Diálogo.

É importante notar que a organização de interface apresentada nesta seção descreve apenas uma visão lógica dos diversos tipos de funções executadas. Ela não especifica, por exemplo, os mecanismos utilizados para troca de informações entre os módulos e componentes da interface. Estas e outras decisões devem ser tomadas durante a *fase de projeto* do software do sistema interativo. Os projetistas são responsáveis pela definição de uma *arquitetura de software* que expresse estes aspectos de maneira clara, conforme discute a próxima seção.

2.4 Arquiteturas de Interfaces

Uma *arquitetura de software de interface* é um modelo abstrato que estabelece a organização (lógica e funcional) do software de um programa interativo. Ela identifica componentes, divide funções e estabelece um protocolo de comunicação entre eles. Estas decisões têm implicações diretas nos custos de criação e manutenção da interface resultante.

Existem diversos tipos de arquitetura de interface, destacando-se as arquiteturas modulares convencionais (por exemplo Seeheim [Gre85] e Slinky/Arch [BFL+92]) e as arquiteturas baseadas em agentes (tais como MVC [KP88] e PAC [BC91]). Uma visão genérica das principais características e diferenças notáveis entre estas arquiteturas é apresentada a seguir.

2.4.1 Arquiteturas Modulares Convencionais

Nestas arquiteturas as funções realizadas pelo componente interativo são particionadas em módulos de software especializados. Cada módulo centraliza e executa um conjunto amplo de funções, mas todas as funções dentro de um módulo são direcionadas para o mesmo tipo de tarefa. Desta forma, as arquiteturas modulares seguem a divisão lógica e modular do software proposta pelas metodologias de desenvolvimento convencionais.

Arquitetura de Seeheim

Uma das arquiteturas de interface mais conhecidas é a de Seeheim [Gre85]. Ela divide uma interface em três módulos: Apresentação, Controle de Diálogo, e Interface com a Aplicação (figura 2.3).

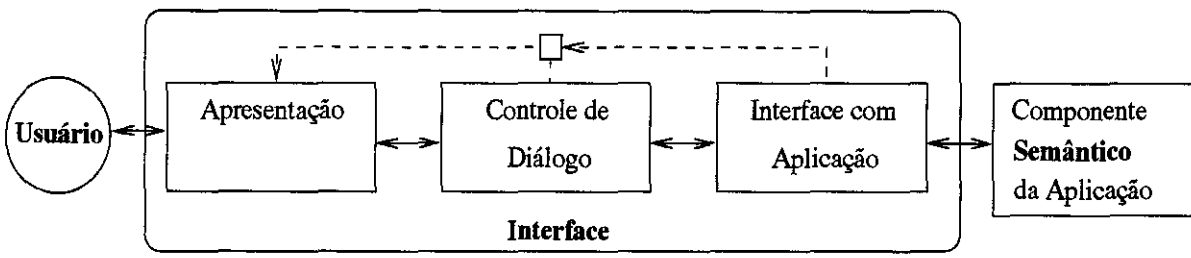

Figura 2.3: Arquitetura de Seeheim

Nesta arquitetura, as funções dos componentes lógicos de Suporte à Interface e Controle de Apresentação (figura 2.2) são realizadas pelo módulo de Apresentação. O componente lógico de Controle de Diálogo é mapeado para o módulo de mesmo nome e funções. O módulo de interface com a aplicação realiza, de maneira explícita e exclusiva, a comunicação entre interface e componente semântico:

- A *apresentação* é responsável por imagens exibidas na tela e trata eventos produzidos por ações dos usuários sobre dispositivos de entrada.
- O *controle de diálogo* examina a seqüência de informações (pedidos e dados fornecidos pelos usuários) providas pela apresentação e decide a ação a ser tomada (que envolve, em geral, chamada de rotinas da aplicação).
- A *interface com a aplicação* define a semântica da aplicação, incluindo a especificação de dados manipulados e as rotinas de comunicação a serem utilizadas entre a interface e o componente semântico da aplicação.

As decisões do controle de diálogo são definidas de acordo com o *contexto da interação*. Por exemplo, em um editor de texto, a mesma ação de entrada de caracteres é interpretada diferentemente de acordo com o *modo* (ou *estado*) corrente: sobrescrever ou inserir.

Arquitetura Slinky

Slinky é um metamodelo que visa a criação de arquiteturas de software voltadas para o atendimento de diferentes objetivos, dentro de um conjunto de requisitos e funcionalidades de interface previamente fixados [BFL+92].

O metamodelo estende a arquitetura de Seeheim, introduzindo um componente *adaptador de apresentação* que fornece *objetos de interação abstratos* independentes de *toolkits* específicos, para uso do controle de diálogo. Por exemplo, um "objeto seletor" pode ser implementado através de menus ou botões em diferentes *toolkits*. Além disto, Slinky refina o modelo de Seeheim, subsidiando a tomada de decisões de projeto que influem na implementação (mecanismos de comunicação e fatores de desempenho, por exemplo), e definindo o tipo de informação que flui entre os módulos (objetos do domínio da aplicação, objetos de interação abstratos, e *widgets*).

As funções especificamente realizadas em cada módulo da arquitetura Slinky são definidas de acordo com os objetivos prioritários para o projetista de interface. Assim, o projetista pode incluir, alterar e excluir funcionalidades, além de migrar funções entre os módulos. Por exemplo, se um *toolkit* oferece *widgets* suficientes para representar os objetos de interação abstratos escolhidos pelo projetista de interface, o módulo adaptador de apresentação pode ser minimizado ou até eliminado.

2.4.2 Arquiteturas Baseadas em Agentes

Um *agente* é um sistema de processamento de informação completo, que inclui receptores/transmissores de eventos, uma memória para manter um estado e um processador que ciclicamente processa eventos de entrada, atualiza o seu estado e, pode produzir eventos

ou mudar seu interesse em classes de eventos de entrada. Agentes comunicam-se com outros agentes, sendo o usuário considerado um agente do sistema.

Em arquiteturas baseadas em agentes, o sistema interativo é organizado em um conjunto de agentes especializados que descentralizam as responsabilidades pelas execuções de funções e cooperam entre si para realizá-las. Estas arquiteturas visam produzir comportamento complexo através de unidades computacionais simples e independentes.

Arquitetura MVC

Uma arquitetura bastante referenciada no desenvolvimento de sistemas interativos é MVC [KP88]. Ela divide uma interface em um conjunto de *tríades* de objetos pertencentes, respectivamente, às classes *model*, *view* e *controller*.

O componente semântico da aplicação (objeto da classe *model*) é isolado da interface (objetos das classes *view* e *controller*) e existe um protocolo de comunicação bem definido que permite a troca de mensagens entre objetos das diferentes classes da arquitetura.

Um objeto da classe *model* representa uma parte do modelo de dados utilizado na aplicação. Este objeto está associado a um ou mais objetos da classe *view*, que tratam da sua apresentação para o usuário. Os objetos da classe *controller* também são associados a um objeto da classe *model* e respondem pela interpretação das ações dos usuários sobre objetos da classe *view* correspondentes.

Arquitetura PAC

A arquitetura PAC [BC91] também se baseia na decomposição de objetos interativos em três tipos de componentes (apresentação, abstração e controle), mas permite a decomposição recursiva de componentes, organizando o sistema interativo como uma hierarquia de agentes (figura 2.4).

Um agente PAC possui três facetas similares às classes definidas pelo MVC – apresentação (*presentation*), abstração (*abstraction*) e controle (*control*) – mas com uma semântica diferente. Qualquer que seja o nível hierárquico, a apresentação representa o comportamento perceptível do agente, a abstração implementa funções do domínio da aplicação de forma independente da apresentação, e o controle une a abstração à apresentação. O controle ainda é responsável pela troca de informações entre a abstração e a apresentação e entre agentes PAC, suprindo uma deficiência da arquitetura MVC, que não possui um elemento explícito para comunicação entre agentes.

A hierarquia de agentes permite que o controle de um agente PAC gerencie, por exemplo, visões de uma mesma informação fornecidas por agentes "descendentes". A mudança em uma visão (fornecida por um agente) é sinalizada para o controle do agente ancestral que dá oportunidade para os demais agentes descendentes se atualizarem.

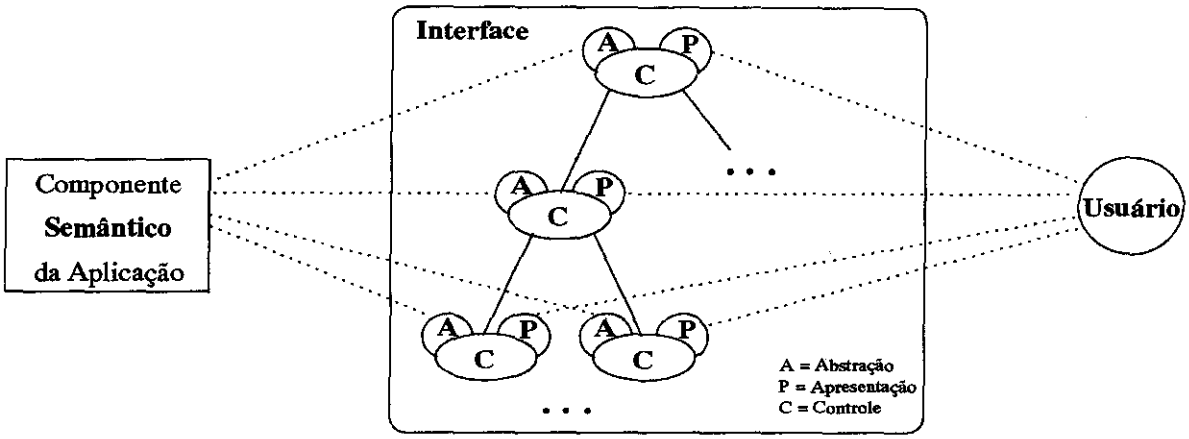

Figura 2.4: Arquitetura PAC

Aplicações geográficas, conforme discute o próximo capítulo, apresentam interfaces com características peculiares que demandam ferramentas de alto nível para viabilizar os processos de projeto e desenvolvimento. O objetivo desta tese é justamente definir ferramentas de alto nível que facilitem o trabalho do projetista de interfaces geográficas.

2.5 Resumo

Este capítulo apresentou os principais componentes da organização de um software de interface – componente interativo e software de suporte – e as principais arquiteturas de interface existentes na literatura. Uma aplicação é vista como um par < interface, componente semântico >, onde o componente semântico corresponde à lógica especificada pelo usuário para realização de uma tarefa, e a interface é o meio de acesso a esta lógica.

As propostas da tese utilizam esta organização, mas aproveitam ao máximo as facilidades existentes no SGBD subjacente para implementar a funcionalidade da interface. Antes de apresentar estas propostas, o próximo capítulo traz o estado da arte no que tange interfaces para SIG.

Capítulo 3

Pesquisas em Interfaces para SIG

3.1 Apresentação

O esforço de pesquisa que tem sido empregado na área de SIG se deve, em grande parte, ao vasto potencial de aplicações existentes para essa tecnologia. Dentre os módulos de SIG, o componente de interface vem despertando particular interesse, já que ele representa um dos fatores chave para a aceitação de um produto. Este capítulo apresenta um panorama das pesquisas feitas na interseção das áreas de interfaces e de sistemas geográficos, discutindo as principais abordagens e os problemas das interfaces atuais.

A próxima seção identifica as áreas de pesquisa em interfaces para SIG. A seção 3.3 descreve possíveis arquiteturas para construção de interfaces de SIG. A seção 3.4 analisa as linguagens de interação com SIG. A seção 3.5 aborda aspectos cognitivos e fatores humanos no contexto de interfaces para SIG. A seção 3.6 resume o capítulo.

3.2 Áreas de Pesquisa em Interfaces para SIG

Existem três grandes dificuldades para uso de SIG [Gou93]. Primeiro, existe uma dificuldade de *treinamento*, já que os SIG atuais são complexos. Em geral, a maior parte do treinamento é dedicada ao aprendizado da linguagem de comando e da estrutura do sistema. A segunda dificuldade está no *mapeamento* que o usuário precisa fazer entre as tarefas que precisam ser realizadas e os comandos disponíveis no sistema. Finalmente, existe a dificuldade de *personalização*. Os SIG atuais oferecem apenas possibilidades rudimentares de adaptação ao usuário.

As dificuldades mencionadas ocorrem, em muitos casos, devido à ausência de ferramentas adequadas para o desenvolvimento de interfaces para SIG. Os problemas relacionados a estas ferramentas são estudados em três áreas de pesquisa: *arquiteturas*, *linguagens de interação*, e *fatores humanos*. O relacionamento entre estas áreas é ilustrado na figura 3.1.

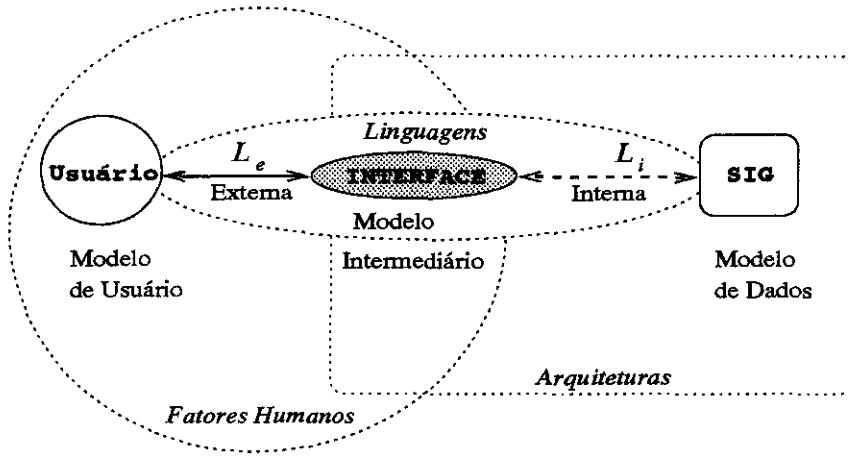

Figura 3.1: Áreas de pesquisa em interfaces para SIG

As *arquiteturas de interface* buscam otimizar a comunicação entre SIG e interface através de uma linguagem interna L_i. O principal objetivo é construir um modelo intermediário que possa ser eficientemente mapeado para os modelos de dados de SIG.

A *linguagem de interação* é usada para especificar a comunicação do usuário com o modelo intermediário da interface. O objetivo é prover mapeamentos eficientes entre este modelo e o modelo mental do usuário. Em analogia aos sistemas relacionais, as linguagens interna L_i e externa L_e correspondem, respectivamente, à álgebra relacional e à linguagem SQL.

Os trabalhos na área de *fatores humanos* usam uma abordagem abstrata e centrada no usuário para definir um modelo mental adequado e especificar mapeamentos para modelos intermediários. No entanto, os mapeamentos não são direcionados por uma linguagem específica, e sim pelo modelo mental, que estabelece os conceitos do modelo intermediário. O objetivo final é adequar a interface ao processo cognitivo do usuário.

O projeto de interface deve estabelecer uma solução de compromisso entre as necessidades conflitantes destas três áreas, discutidas nas seções seguintes.

3.3 Arquiteturas de Interface em SIG

Uma *arquitetura de interface* descreve um modelo para construção de interfaces para determinado tipo de aplicação. Em SIG, quatro aspectos são necessários para definir a interface a ser implementada. O primeiro aspecto é a *integração* entre o sistema geográfico e a interface. Em segundo lugar, a arquitetura precisa identificar os principais *módulos da interface*, especificando sua funcionalidade e interoperabilidade. O terceiro aspecto é seu *modelo de dados intermediário*, que deve ser capaz de expressar os conceitos do usuário, associado aos mecanismos de conversão destes conceitos para o modelo de dados

espacial do SIG. Finalmente, a arquitetura deve estabelecer uma *divisão de tarefas* a serem realizadas pelo SIG e pela interface.

Embora os quatro aspectos sejam importantes, grande parte das arquiteturas descritas na literatura omite um ou mais desses aspectos. As propostas existentes permitem identificar duas categorias de arquiteturas de interface. A primeira define arquiteturas voltadas para a implementação da interface do SIG propriamente dito ([Env92], [PMP93] e [Rig95], por exemplo). A segunda categoria engloba a primeira, contendo arquiteturas que estabelecem um substrato comum para implementar interfaces para aplicações geográficas sobre SIG. As propostas de [Voi91], [AYA+92] e [Voi94] são exemplos desta categoria. A análise apresentada a seguir compara diferentes abordagens para os quatro aspectos fundamentais de uma arquitetura de interface para SIG.

3.3.1 Ligação entre Interface e SIG

O grau de influência da arquitetura do SIG sobre a arquitetura do sistema de interface depende basicamente do tipo de conexão entre os dois sistemas. Existem duas abordagens opostas para realizar esta conexão: ligação *forte* e ligação *fraca* [Voi94]. A ligação forte considera a interface como parte do SIG, compartilhando o seu modelo de dados espacial. Na abordagem fraca o sistema de interface é totalmente independente do SIG.

Ligação Forte

Na *ligação forte*, a arquitetura da interface é diretamente influenciada pelos detalhes da arquitetura do sistema geográfico (seção 1.2.1). Em uma arquitetura de SIG com SGBD *proprietário*, a interface deve possuir módulos para interagir com cada componente do sistema geográfico (figura 3.2, parte da direita).

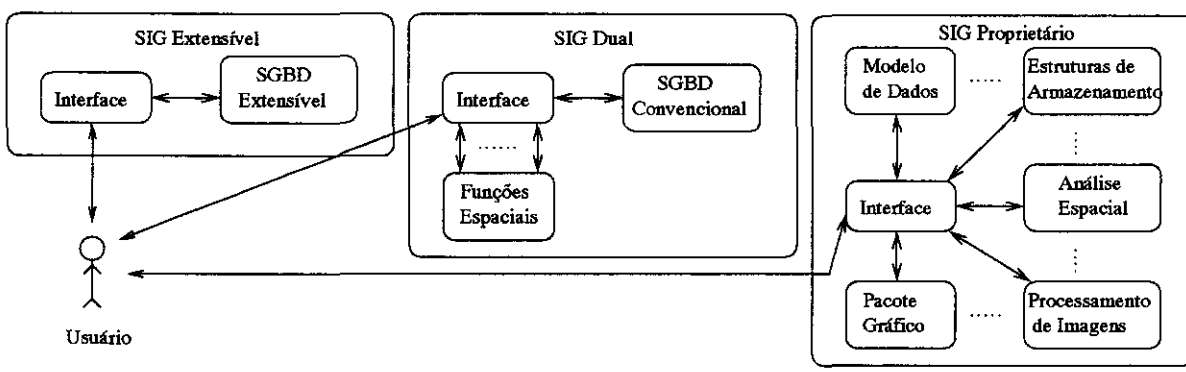

Figura 3.2: Ligação forte entre interface e SIG

Na arquitetura de SIG *dual*, a interface deve prover pelo menos dois tipos de módulo:

(a) para efetuar a comunicação com o SGBD convencional e (b) para interagir com os módulos que tratam das funções espaciais (figura 3.2, parte central).

No caso de arquitetura de SIG *extensível*, a interface precisa de um único módulo para gerenciar o conhecimento sobre o SGBD extensível subjacente (figura 3.2, parte da esquerda). Este módulo, no entanto, é bastante complexo, pois precisa representar todas as características específicas do SGBD utilizado.

Qualquer que seja a arquitetura do SIG (proprietária, dual, ou extensível), a interface fortemente ligada possui conhecimento sobre as estruturas de dados internas, e pode manipulá-las sem a intervenção de outros módulos do sistema. Além disto, não existe necessidade de mapeamentos entre modelos, tornando o sistema mais eficiente.

Por outro lado, existe maior dificuldade para manutenção e evolução tanto da interface quanto do SIG. Por exemplo, não é fácil determinar os módulos que precisam ser modificados para acrescentar novos tipos de dados ao sistema, pois não existe uma separação clara entre as funções da interface e do SIG propriamente ditas. Além disso, este tipo de ligação torna praticamente inviável a adaptação da interface a outros SIG.

Ligação Fraca

Na arquitetura proprietária a interface pode interagir com todos os módulos do SIG. Esta liberdade de comunicação diminui na arquitetura dual (onde a interface se comunica basicamente com dois módulos do SIG). Na arquitetura extensível a interface mantém contato apenas com o SGBD extensível. Desta forma, o grau de encapsulamento da interface com relação aos módulos do SIG cresce da arquitetura proprietária para a extensível. A abordagem de *ligação fraca* se baseia justamente na maximização deste encapsulamento.

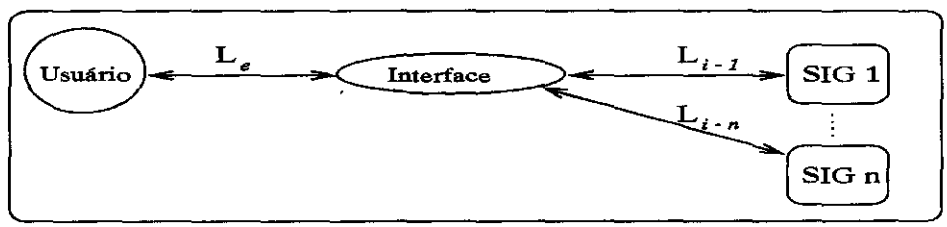

Figura 3.3: Ligação fraca entre interface e SIG

Nesta abordagem, a interface é um sistema independente e isolado do sistema geográfico, como descrito na figura 3.3. Para que os sistemas possam interagir, protocolos de comunicação e conversão de modelos devem ser especificados *a priori*. Esta necessidade não existe na abordagem forte, já que a interface incorpora conhecimentos sobre os mecanismos de acesso aos dados. Por outro lado, a integração fraca apresenta vantagens: (1) a independência entre os sistemas promove a especialização das tarefas e permite a reutilização da interface em diferentes SIG; (2) é preciso atender a necessidade

de desenvolvimento de sistemas abertos, que possam ser integrados com outros produtos [VvO92, GR93, VS94]; (3) pacotes e serviços especializados podem ser incorporados a um dos sistemas sem afetar o outro; (4) a comunicação entre os sistemas é padronizada (por meio de uma linguagem interna L_i, provida pelo SIG).

Convém notar que não existe uma divisão absoluta entre as abordagens de integração fraca e forte. De fato, a passagem da ligação forte para a fraca é gradual, sendo que a integração com SIG extensível pode ser considerada uma solução intermediária.

Exemplos de Abordagens de Ligação Existentes

A maior parte das propostas existentes na literatura descreve arquiteturas que adotam a ligação fraca com o SIG subjacente [Voi91, PMP93, Voi94, Rig95]. Por outro lado, os trabalhos que descrevem a arquitetura de interface de sistemas comerciais adotam, via de regra, ligação forte entre interface e SIG.

O sistema ARC/INFO, um dos SIG comerciais mais difundidos na atualidade, é um exemplo típico de ligação forte [Env92]. Apesar de ser um sucesso comercial, a arquitetura de interface do sistema apresenta alguns problemas conceituais. Em primeiro lugar, cada ferramenta possui seus próprios componentes de interface, fortemente ligados a ela, tornando difícil a tarefa de separar o código da interface daquele relativo às funções do sistema. Além disso, as ferramentas estão baseadas no modelo de dados específico do sistema ARC/INFO, o qual não é facilmente mapeado para outros modelos espaciais existentes.

Uma arquitetura para interface de aplicações em SIG aparece em [Voi91, Voi94]. A principal contribuição desta arquitetura é uma análise detalhada sobre o problema de integração entre interface e SIG. O resultado desta análise descreve os pontos fortes e as deficiências das abordagens de integração fraca e forte, indicando uma possível solução de compromisso, baseada em ferramentas para construção de interfaces dentro de um ambiente de bancos de dados geográficos predefinido.

A arquitetura de interface apresentada em [Rig95] visa sobretudo a independência entre interface e SIG. Para isso a arquitetura define três níveis de representação de informações. O nível mais baixo é o nível *conceitual* do SGBD, seguido de um nível de *representação abstrata* (modelo intermediário de interface) e finalmente do nível de *representação do usuário*. Os níveis da arquitetura oferecem um certo grau de independência entre interface e SIG, mas introduzem a necessidade de tradução entre os três modelos.

3.3.2 Componentes de uma Arquitetura de Interface

Os módulos de uma arquitetura de interface definem a organização e a funcionalidade do sistema de interação. Além disto, a estrutura e os relacionamentos entre os componentes determinam a maneira pela qual a interface é desenvolvida. Uma característica comum

em aplicações geográficas é a necessidade de realizar operações de apresentação, consulta e modificação de dados geo-referenciados. Em geral, estas operações utilizam o conceito de mapa, representando uma coleção de objetos geo-referenciados logicamente relacionados.

O conceito de mapa (no sentido em que as pessoas estão habituadas a raciocinar, isto é, em termos de representação cartográfica) é necessário, mas não suficiente para aplicações que fazem manipulação integrada de dados convencionais e geo-referenciados. Mesmo assim, grande parte das arquiteturas de interface para SIG limita-se à definição de módulos para realizar operações sobre mapas, dificultando a integração de informações no código da interface.

Componentes de Arquiteturas Recentes

A arquitetura proposta em [Voi94] define duas camadas: o *nível de suporte*, contendo sistemas que interagem com a interface, e o *nível de interface* propriamente dito (figura 3.4).

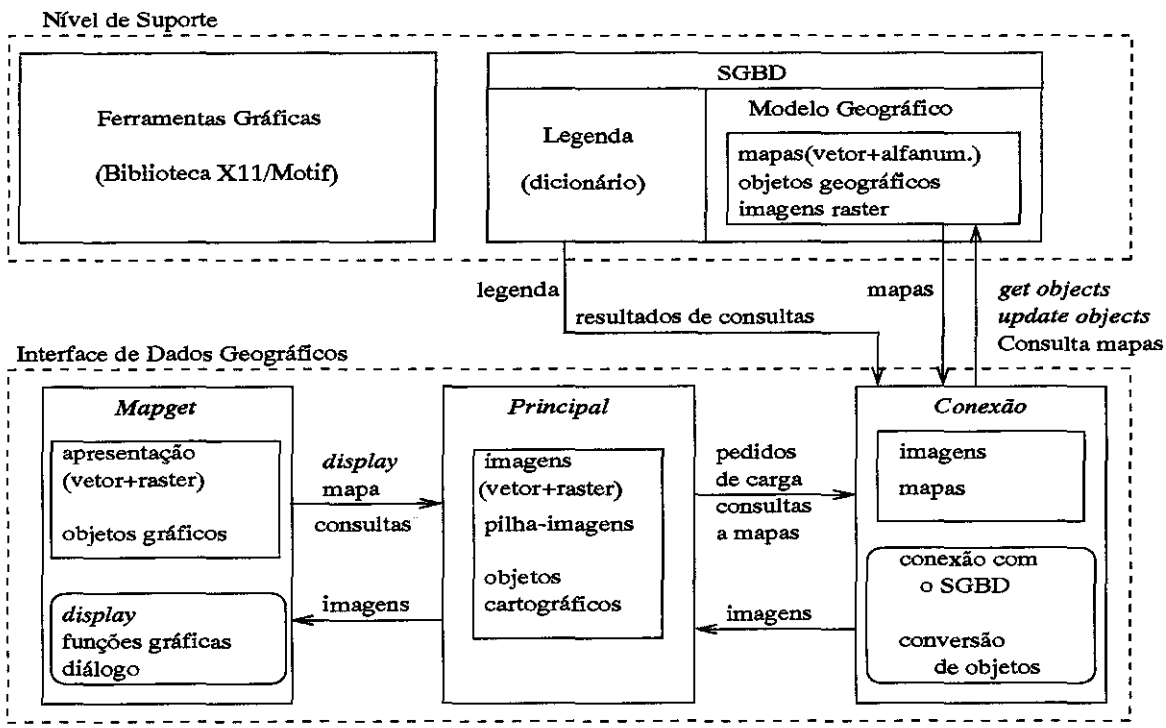

Figura 3.4: Arquitetura adaptada de [Voi94]

O nível de suporte contém dois módulos ortogonais: o *módulo de ferramentas gráficas*, usado para visualização de dados, e o *módulo de banco de dados*, que impõe uma série de restrições sobre o SGBD geográfico visando oferecer uma solução de compromisso para o

problema de ligação da interface ao SIG. A interface pode se conectar apenas a SIG que suportam as restrições impostas pelo módulo de banco de dados.

O nível de interface da arquitetura é formado por três módulos fortemente relacionados. O módulo *Mapget* é responsável pelo diálogo com o usuário. O módulo *Principal* lida com visões abstratas de mapas e controla a comunicação entre o módulo Mapget e o módulo *Conexão*. Este último é de fato o principal módulo da arquitetura, sendo responsável pelo mapeamento de dados entre interface e SGBD. Ele é menos complexo do que seria na abordagem de integração fraca, graças às restrições impostas sobre o SGBD.

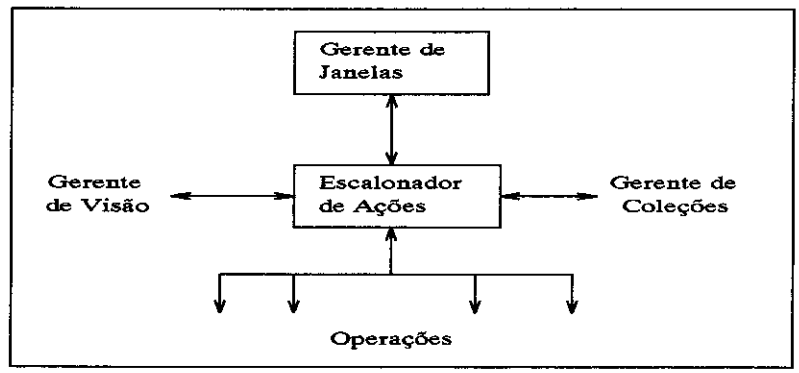

Figura 3.5: Arquitetura adaptada de [AYA+92]

A figura 3.5 descreve a arquitetura de interface proposta em [AYA+92]. O *gerente de janelas* trata de operações gráficas sobre o sistema de janelas. O *escalonador de ações* encaminha as operações solicitadas pelo usuário para execução. A maior parte das operações pode ser iniciada através de interações gráficas ou textuais. No entanto, a interação textual é mais poderosa, pois existem comandos textuais que não podem ser expressos graficamente. Esta escolha no nível de arquitetura direciona e limita o conjunto de soluções possíveis para o problema de linguagem de interação (seção 3.4).

Os componentes de interface da arquitetura proposta em [Env92] estão distribuídos pelo conjunto de ferramentas que compõem o sistema ARC/INFO (figura 3.6), implementando uma ligação forte entre interface e SIG. Apesar dos problemas de redundância decorrentes deste tipo de ligação, a interface apresenta certa coerência porque todas as ferramentas estão baseadas no mesmo modelo de dados espacial (geo-relacional híbrido).

Os componentes da arquitetura proposta em [Rig95] formam uma hierarquia de objetos compostos (agentes), organizados de acordo com o modelo PAC (seção 2.4). O principal agente da arquitetura tem uma faceta de abstração formada pelos modelos de dados intermediário (abstrato) e do SGBD (figura 3.7).

O modelo do usuário define mapas como objetos complexos formados por estilos gráficos e listas de camadas temáticas. Todos os mapas são gerenciados pelo componente de controle, que garante a coerência das apresentações gráficas. O usuário interage com

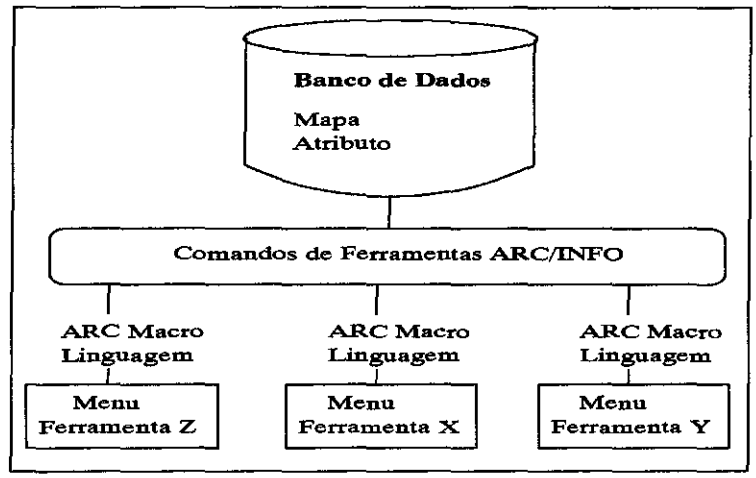

Figura 3.6: Arquitetura adaptada de [Env92]

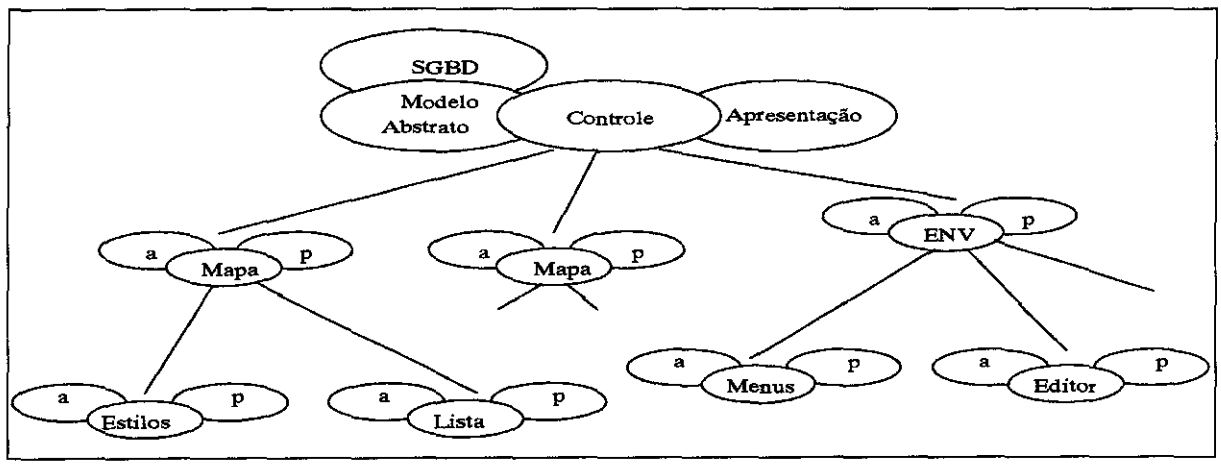

Figura 3.7: Arquitetura adaptada de [Rig95]

os componentes de apresentação. Os mapas são os principais objetos de apresentação. enquanto objetos secundários (como listas e estilos) controlam os parâmetros da apresentação. O objeto ENV representa o ambiente, ou seja, todos os objetos estáticos da interface (janelas e botões, por exemplo).

3.3.3 Modelo de Dados Intermediário de Interface

A interface é responsável por mapear o *modelo mental* do usuário para o *modelo de dados* do SIG. Este mapeamento exige vários tipos de tradução, em geral obtidos através de uma camada auxiliar denominada de *modelo intermediário* da interface.

Na área de SIG não existe padrão de desenvolvimento e tampouco um modelo de dados e metáfora universais. Por este motivo, usuários de SIG são obrigados a entender as

estruturas internas do sistema e, a cada novo sistema, modificarem sua forma de interação. Isto ocorre porque a interface, ao invés de representar o modelo mental do usuário, está mais próxima das diferentes implementações de modelos de dados em SIG.

Para corrigir este problema é preciso um modelo intermediário que mapeie *representações* espaciais descritas pelo modelo de dados do SIG para *conceitos* espaciais compreendidos pelo usuário. A interface deve tirar proveito do SGBD subjacente e, ao mesmo tempo, oferecer aos usuários um modelo que contemple os conceitos por eles utilizados.

Algumas Propostas de Modelos de Dados Intermediários

O modelo intermediário de [Voi94] usa a noção de mapas organizados em camadas, que podem ser opacas ou transparentes. Esta propriedade permite a visualização de vários mapas sem a necessidade de realizar uma operação de sobreposição. Um *gerente de camadas* provê operações que permitem controlar a ordem das camadas, e uma aplicação geográfica é formada basicamente por um conjunto de pilhas de camadas.

A arquitetura proposta em [Rig95] utiliza três tipos de modelos de dados (chamados de *níveis* da arquitetura) para tratar duas categorias básicas de operações de interface, que são as operações de banco de dados (navegação e consulta, por exemplo) e as operações gráficas (tais como sobreposição ou controle de parâmetros visuais de apresentação). O *nível abstrato* suporta as operações gráficas enquanto o *nível conceitual* trata das operações de bancos de dados. O *nível de usuário* contém objetos que são diretamente visualizados pelo usuário, como mapas, estilos de apresentação, e legendas.

O modelo de dados intermediário da interface no sistema ARC/INFO é derivado do próprio modelo geo-relacional híbrido do SIG [Env92]. Este modelo utiliza uma estrutura topológica para armazenar um conjunto de camadas temáticas (ou *coberturas*). Cada camada é uma unidade de armazenamento de uma realidade geográfica (isto é, um *tema*).

A arquitetura proposta nesta tese utiliza o modelo GMOD, apresentado em [Pir97], como modelo intermediário da interface, conforme discute a seção 4.4.1.

3.3.4 Divisão de Tarefas entre Interface e SIG

Há dois grupos de funções disponíveis na interface de um sistema geográfico: *funções de interface*, que envolvem apenas mecanismos de interação com o usuário (entrada de dados e apresentação de informações); e *funções de* SIG, que envolvem o processamento de dados geo-referenciados (isto é, análise de dados espaciais). A arquitetura da interface precisa especificar a divisão entre os dois grupos de funções, determinando o componente do sistema que será responsável pela execução de cada função.

Esta divisão de responsabilidades é desconsiderada em muitas propostas de arquiteturas de interfaces, sobretudo nas que especificam uma ligação forte entre SIG e interface, já

que neste caso não existe uma separação evidente entre o código da interface e o código que efetua recuperação e análise de dados espaciais.

A ausência da divisão de tarefas no nível da arquitetura da interface causa diversos problemas, entre os quais se destaca a redundância de código. Isto ocorre porque as funções de interface são repetidas em cada função de análise desenvolvida no SIG. Este tipo de redundância dificulta as tarefas de manutenção e evolução do sistema.

Exemplos de Abordagens para Divisão de Tarefas

A arquitetura de [Voi91] define um conjunto de ferramentas para implementação de interfaces específicas. Nesta abordagem a criação de uma interface envolve cinco etapas: leitura de dados e meta-dados do BD; conversão para o modelo intermediário; definição de objetos de interação; criação da apresentação de acordo com parâmetros do ambiente; e ligação da apresentação a um componente de interface capaz de apresentar mapas. O SIG funciona apenas como repositório de dados geo-referenciados, atuando na primeira etapa do processo; as demais etapas são efetuadas pela interface.

As ferramentas e aplicações de ARC/INFO constroem modelos de dados específicos com base no modelo espacial genérico do sistema [Env92]. Por exemplo, existem ferramentas separadas para entrada de dados, análise espacial, desenvolvimento de aplicações, e produção cartográfica. Cada ferramenta deriva seu próprio modelo, e tem seus componentes de interface associados, utilizando elementos de interação (*widgets*) padronizados.

A arquitetura proposta em [Voi94] efetua a divisão de tarefas com base em três categorias de funções: funções de *representação visual* (apresentação de dados espaciais); funções de *manipulação de mapas*; e funções de *interação com o banco de dados*. Um problema desta proposta é que a taxonomia de funções não define uma partição disjunta das operações identificadas. Não fica claro, por exemplo, a separação das facilidades de consulta oferecidas pelo SGBD e pela interface, já que operações de consulta aparecem tanto como funções de manipulação de mapa quanto de interação com o banco de dados.

3.4 Linguagens para Interação com SIG

Os sistemas de interface atuais se enquadram em dois paradigmas de interação: textual (tradicional) e visual (utilizando cores, gráficos e ícones). A presença de dados geo-referenciados torna as apresentações gráficas mais adequadas em SIG. Por exemplo, interfaces textuais apresentam uma dada área através de uma lista de coordenadas de seus pontos limites, enquanto uma interface visual apresenta o desenho do polígono que representa a área em questão. Por outro lado, algumas características são mais facilmente

descritas através de textos. No exemplo, as características da área, tais como os dados do seu proprietário, ou o seu valor venal, seriam melhor representadas textualmente.

Os paradigmas de interação textual e visual têm sido utilizados para definir diferentes tipos de linguagens de interação:

1. *Linguagens de comando* são aquelas em que o usuário escreve instruções que refletem o conjunto de operações implementadas no SIG. *Extensões de* SQL representam a abordagem mais utilizada de linguagem de comando em SIG.

2. *Linguagens semi-visuais* integram uma linguagem textual, geralmente baseada em SQL, a um ambiente gráfico que inclui janelas e menus. O usuário não precisa se lembrar do nome dos comandos, mas ainda precisa pensar em termos de construção de sentenças, pois o ambiente preserva a estrutura da linguagem de comandos.

3. As *linguagens de manipulação direta* descrevem restrições espaciais através de diagramas e da manipulação de objetos gráficos com auxílio do *mouse*. É uma maneira natural de expressar conceitos espaciais, mas apresenta problemas para descrever relacionamentos espaciais isolados, já que um diagrama contém informação sobre os tamanhos relativos dos objetos, suas formas e relacionamentos topológicos.

Além do paradigma de interação, dois fatores devem ser considerados em uma linguagem de interação com SIG. O primeiro se refere ao conjunto de operadores espaciais suportado, que determina o poder de expressão e a eficiência de implementação da linguagem. O segundo fator importante é a capacidade de manipulação de informações já obtidas para compor diretamente novas expressões e consultas. A interação com SIG possui este caráter incremental, onde os resultados obtidos servem de parâmetros para novas operações. Uma parte considerável das interfaces existentes apresenta dificuldades para prover esta capacidade ao usuário.

3.4.1 Linguagens Baseadas no Paradigma Textual

Interfaces textuais têm a vantagem de prover suporte para expressão de ações complexas, que não podem ser facilmente expressas por meios gráficos. Além disso é possível definir *macros* que encapsulam seqüências complexas de comandos. O principal problema está na distância semântica entre o modelo de dados utilizado pela linguagem e o modelo mental do usuário de SIG. Em geral, linguagens textuais não permitem a expressão natural de conceitos espaciais [Ege92]. Apesar disto, as extensões espaciais de SQL são o tipo de linguagem de interação textual mais utilizado nas propostas atuais.

GQL A linguagem de consulta GQL [Goh89] adota padrões reconhecidos (como SQL e GKS) para produzir uma interface genérica que possa ser utilizada em um ambiente de computação heterogêneo. O acesso aos bancos de dados é feito através de chamadas SQL embutidas em programas de aplicação. Um único operador, &, é acrescentado à sintaxe padrão do SQL, para permitir a apresentação de objetos gráficos como resultado de uma consulta. Para isto é preciso que o dado referenciado pelo operador & na consulta tenha um componente gráfico correspondente no BD. Componentes gráficos possuem um identificador único que permite sua associação a uma tupla de uma relação através da chave primária. Um dicionário de dados gráficos mantém uma lista das entidades que possuem estes componentes gráficos, permitindo ao usuário decidir sobre a possibilidade de uso do operador &.

A linguagem GQL trabalha com o nível mais baixo de funcionalidades gráficas de GKS. As primitivas de saída básicas da linguagem, como *polyline* ou *cell array*, podem ter sua aparência controlada interativamente pelo usuário, determinando a cor, o tipo de linha, e o padrão de preenchimento de objetos gráficos. O ambiente gráfico pode ser descrito através do operador *define*, que permite apenas a especificação do tamanho da janela gráfica e o controle de atributos gráficos primitivos. Edições mais elaboradas, tais como mudança de escala ou movimentação dos objetos gráficos, não são permitidas pela linguagem. Além disso, a linguagem é restrita a dados armazenados no formato vetorial.

GEOQL A linguagem GEOQL [Ooi90] é uma extensão espacial de SQL que provê operadores para determinar diversos tipos de relacionamentos espaciais, tais como: interseção, adjacência, junção, terminação, continência, localização, pertinência, e proximidade.

"Encontre todas as cidades com mais de 5000 habitantes em um raio de 200 km do monte Buffalo dentro da área delimitada na tela."

```
SELECT    CITY.Name
FROM      CITY, MOUNTAIN
WHERE     MOUNTAIN.Name = ''Buffalo'' and
           CITY.Population > 5000 and
           CITY within 200 km of MOUNTAIN and
           window contains CITY and
           window contains MOUNTAIN.
```

Figura 3.8: Exemplo de consulta em GEOQL adaptado de [Ooi90]

A linguagem delega ao projetista do BD a tarefa de definir em uma única relação todos os atributos não espaciais de uma classe de entidades geográficas. Além disto, cada relação deve conter informações relacionadas com apenas uma classe de entidades. Desta forma,

a relação pode ser usada para identificar o tipo do operando (ou seja, a classe de entidade geográfica) em uma expressão espacial. Isto é importante porque alguns operandos da linguagem não são comutativos (terminação e continência, por exemplo).

A região geográfica de interesse para uma consulta pode ser definida na tela através de uma janela (*window*). O predicado "*window contains* entidade" é adicionado aos predicados da consulta, para cada entidade geográfica referenciada na consulta (figura 3.8). A operação de *zoom* é implementada através deste mecanismo: apenas a área apontada pelo usuário é desenhada no mapa de saída. A linguagem suporta dados espaciais do tipo ponto, linha e região. Dados não espaciais são armazenados em relações, enquanto dados espaciais são mantidos em um arquivo externo.

A definição de um conjunto fechado de operadores espaciais é um ponto negativo de GEOQL. O problema é que esta rigidez limita o poder de expressão da linguagem, impedindo a utilização de outros operadores espaciais (métricos e direcionais).

Spatial SQL A extensão de SQL apresentada em [Ege94] expressa predicados sobre atributos geométricos através de operadores espaciais. A interação é feita por meio de duas linguagens: uma *linguagem de consulta*, para recuperação de dados, e uma *linguagem de visualização*, para definição de apresentações. O usuário estabelece inicialmente o ambiente de visualização e passa a realizar consultas que são apresentadas de acordo com este ambiente. A mesma abordagem é empregada na linguagem LEGAL [CCH+96].

A linguagem de consulta estende os domínios relacionais com objetos espaciais de até três dimensões, generalizados em um domínio chamado *spatial*. Operadores espaciais unários fazem acesso a uma propriedade espacial de uma tupla (operadores que permitem a recuperação da dimensão e dos limites de um objeto, por exemplo). Duas operações espaciais binárias (distância e ângulo de direção) calculam um valor com base em duas tuplas das relações espaciais. Funções agregadas, tais como mínimo e máximo, podem ser aplicadas sobre resultados de operadores binários. Relacionamentos topológicos binários (disjunção, sobreposição, continência, entre outros) resultam em valores booleanos, do mesmo modo que os comparadores binários tradicionais.

Atributos espaciais são utilizados da mesma forma que atributos convencionais na cláusula *select* (como uma projeção), ou na cláusula *where* (como um predicado). Diversas geometrias podem ser definidas para um único objeto (conceito de representação múltipla) e um qualificador (*pick*) é introduzido para formular consultas que referenciam objetos espaciais apresentados na tela. Ele é usado como um predicado e pode qualificar atributos espaciais da cláusula *where*. O usuário aponta para o objeto gráfico que deve corresponder ao tipo de objeto especificado na consulta.

A informação especificada na linguagem de apresentação é integrada com a consulta propriamente dita, e a apresentação dos resultados é feita de acordo com o ambiente

de visualização predefinido. Todas as consultas processadas seguem o mesmo estilo de visualização, enquanto o usuário não modificar o ambiente através da linguagem de apresentação. Os tipos de especificação de ambiente de visualização incluem o modo gráfico (cores, símbolos), a escala e a janela de apresentação (delimitando a área a ser apresentada), e a informação de contexto espacial (partes de relações espaciais a serem adicionadas aos resultados da consulta).

Além de SQL, outras linguagens de bancos de dados podem ser utilizadas como base para interação textual com SIG. A álgebra geo-relacional e a linguagem GEO-SAL, descritas a seguir, são exemplos desta abordagem.

Álgebra Geo-Relacional *Linguagens algébricas* permitem o mapeamento eficiente de operações para funções de um SIG. Um exemplo representativo deste tipo de linguagem é a *álgebra Geo-Relacional* [Gut88]. A linguagem estende a álgebra relacional incluindo tipos de dados e operadores geométricos e permitindo aninhamento de relações. A representação dos objetos geométricos não é visível ao usuário e só pode ser manipulada através dos operadores espaciais propostos. Uma tupla descreve um objeto espacial através de atributos convencionais e geométricos enquanto uma relação descreve uma coleção homogênea de objetos espaciais.

Os tipos de dados definidos na álgebra são: números, *strings*, booleanos, relações, e tipos geométricos (tais como pontos, linhas, regiões, áreas, e polígonos). Os operadores da álgebra geo-relacional são agrupados em cinco classes:

1. Operadores relacionais ($\cup$, $\cap$, $-$, $\times$, $\bowtie$, σ, π) e extensões (λ e *extract*). As extensões permitem que os operadores σ e π tomem como parâmetro uma expressão booleana da álgebra. O operador λ acrescenta dinamicamente um atributo a uma relação. O operador *extract* extrai um valor atômico de uma relação com base em uma condição de seleção de tupla e em um nome de atributo para o valor extraído.

2. Predicados geométricos ($=$, $\neq$, *inside*, *outside intersects*, *is_neighbor_of*) comparam dois objetos geométricos de diversas maneiras. A comparação pode ser parte da expressão que parametriza uma seleção ou uma junção. Por exemplo:
   (x *inside* y) := points(x) $\subseteq$ points(y)

3. Transformadores geométricos (*intersection*, *overlay*, *vertices*, *voronoi*, *closest*) recebem como entrada relações com atributos geométricos e produzem uma relação que inclui novos objetos geométricos.

4. Operadores que retornam objetos geométricos atômicos são usados para construir o polígono convexo de um conjunto de pontos (operador *convex_hull*), e para determinar o centro de um conjunto de pontos (operador *center*).

5. Operadores geométricos que retornam valores escalares efetuam operações métricas (*distance*, *diameter*, *perimeter*, *area*, entre outros).

É possível estabelecer uma correspondência direta entre operadores geométricos desta álgebra e algoritmos geométricos clássicos, garantindo uma implementação eficiente.

GEO-SAL A linguagem de consulta GEO-SAL [SH91] incorpora facilidades espaciais a uma linguagem (SAL) de análise de dados em um sistema de bancos de dados relacionais estatísticos. A principal modificação introduzida visa permitir o uso de tipos de dados espaciais nas colunas das relações. A representação interna de objetos espaciais é visível ao usuário, embora não seja necessário, em geral, conhecimento sobre ela para a formulação de consultas. A linguagem suporta dados espaciais definidos como pontos, linhas, polígonos, e partições, definindo cinco classes de operadores espaciais:

1. Operadores *unários* extraem dados geométricos de uma tupla, tais como as coordenadas de um ponto, ou a área de um polígono;

2. Operadores de *transformação* modificam algumas propriedades espaciais, incluindo a rotação de um polígono, ou a translação de uma partição;

3. Operadores binários que calculam *relacionamentos geométricos* têm como exemplo típico o operador de distância euclideana;

4. Operadores binários *booleanos* permitem o teste de relacionamentos topológicos, tais como os de continência, de sobreposição, e de disjunção;

5. Operadores de *construção* de objetos criam novos objetos a partir de outros já existentes. Um exemplo típico são os operadores de conjunto (união, diferença).

A composição de operadores permite o cálculo de outros tipos de dados geométricos. Por exemplo, a aplicação do operador de comprimento (*length*) sobre o resultado do operador limite (*boundary*) gera o perímetro de um polígono.

Tanto a álgebra geo-relacional quanto GEO-SAL podem expressar consultas complexas de modo conciso. No entanto, ambas as linguagens são inadequadas para usuários finais.

3.4.2 Linguagens Baseadas no Paradigma Visual

Linguagens visuais apresentam duas grandes qualidades: são mais naturais para o usuário final (ou seja, são mais próximas de seu modelo mental); e permitem a combinação e reutilização de resultados de consultas para formulação de novas consultas de maneira simples. No entanto as linguagens visuais não são padronizadas e impõem muita dificuldade para

formulação de consultas complexas, especialmente as que envolvem negação e disjunção [BM92]. Além disso, os operadores e parâmetros de consultas visuais são sobrecarregados semanticamente, podendo gerar ambigüidades na interpretação.

Linguagens Semi-Visuais

Egenhofer e Frank apresentam, em [EF88b, EF88a], a definição de uma linguagem de consulta semi-visual para SIG onde se destacam três aspectos: o leiaute da interface, o diálogo com o usuário, e o controle da apresentação de dados.

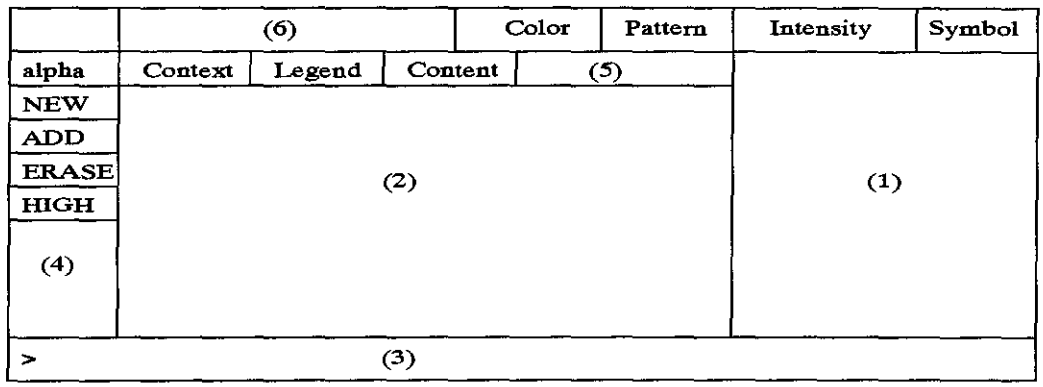

Figura 3.9: Leiaute de interface adaptado de [EF88b, EF88a]

O *leiaute* da interface provê facilidades para visualização tanto de resultados como de formulação de consultas (figura 3.9). O objetivo é facilitar a comparação visual, que é um dos principais instrumentos de análise utilizados pelo usuário. O leiaute proposto define seis áreas: (1) saída léxica; (2) saída gráfica; (3) entrada léxica; (4) painel de operações de controle; (5) menu de propriedades de mapa; e (6) menu de apresentações gráficas.

O *diálogo com o usuário* está voltado para a formulação de consultas. As informações podem ser apresentadas textual ou graficamente, inclusive através de referências a objetos dos mapas apresentados. A linguagem dispõe de quatro operações para gerenciamento das *apresentações gráficas*: *New*, para eliminar resultados anteriores da janela gráfica; *Overlay*, para acrescentar o resultado da consulta corrente ao mapa previamente apresentado; *Erase*, para eliminar do mapa apresentado apenas os resultados da consulta corrente; e *Highlight*, para destacar no mapa os resultados da consulta corrente. Estas operações se encontram no painel de controle da interface (figura 3.9, parte (4)).

A combinação de resultados de diferentes consultas é um ponto forte da linguagem. No entanto, o mapa pode perder a clareza devido ao excesso de conteúdo. Para resolver este problema, o botão *Content* apresenta uma descrição dos predicados aplicados a cada classe de objetos representada no mapa. A seleção das informações a serem apresentadas no mapa é complementada pela especificação de suas propriedades gráficas. O usuário pode

escolher cores, padrões e símbolos predefinidos na interface. A escolha destas propriedades define uma legenda para o mapa, a qual pode ser visualizada através do botão *Legend*. O botão *Context* permite a definição de escala e contexto espacial de um mapa.

Linguagens de Manipulação Direta

Cigales e *Grog* são linguagens de manipulação de dados (DML) voltadas para consultas temáticas e consultas orientadas a redes, respectivamente. [CM91] propõe a unificação destas linguagens para formar um ambiente gráfico de consulta a SIG.

A linguagem *Grog* modela dados espaciais através de grafos direcionados, suportando quatro classes de consultas: avaliação de caminhos (deslocamento entre dois locais considerando-se restrições existentes); interseção de caminhos (subcaminhos comuns); inclusão de caminhos (avaliação de caminho com subcaminhos específicos); e manipulações de nós (locais comuns entre dois caminhos). As manipulações envolvem operações sobre arcos (ligação direta ou transitiva; interseção de ligações; inclusão de arcos), sobre nós (inclusão e interseção de nós), e sobre arcos e nós (inclusão de nós em um caminho).

Uma consulta gráfica em *Grog* é formulada como um conjunto de grafos orientados e rotulados. Os rótulos podem ser variáveis ou constantes, e três tipos de arcos são previstos para modelar ligações, inclusão e interseção. Arcos são operadores binários orientados que podem representar os resultados de uma subconsulta (um conjunto de caminhos).

Cigales é uma linguagem orientada a consultas temáticas usando dois objetos gráficos básicos: linha e área. Os operadores espaciais incluem adjacência, inclusão, interseção, e distância euclideana.

Uma consulta gráfica em *Cigales* é formulada pela seleção de objetos básicos, aos quais se associa semântica através de rótulos que os identificam com entidades do banco de dados, e pela aplicação de operadores espaciais sobre os objetos selecionados. Existem dois espaços de consulta: o espaço de trabalho, onde a consulta é formulada, e o espaço de consultas, onde aparecem as consultas validadas pelo usuário.

A unificação de *Grog* e *Cigales* parece ser uma boa idéia, já que a primeira não contempla operações sobre "áreas" enquanto a segunda não suporta operadores de "caminho". No entanto, o processo de unificação das linguagens dá origem a uma série de ambigüidades, permitindo diferentes interpretações para seu significado: a semântica de cada operador da linguagem é bem definida, mas a composição de operadores pode levar a erros de interpretação do significado da consulta. O gerenciamento de consultas complexas também é um problema para a linguagem unificada. Consultas contendo um grande número de objetos básicos não podem ser expressas de maneira razoável. Mais ainda, a extensão necessária para acomodar tais consultas envolve problemas difíceis, como os relacionados com generalização cartográfica.

3.4.3 Comentários sobre Linguagens de Consulta em SIG

Diferenças fundamentais existem entre gerenciamento de consultas espaciais e convencionais. Conceitos tabulares são pouco indicados para representações de BD espaciais, pois os dados estão associados com geometrias e apresentações do tipo *mapa*. De fato, mesmo o uso mais simples de dados espaciais requer uma análise de limites geográficos para a recuperação dos dados. Como não é possível armazenar todos os relacionamentos espaciais, a linguagem de consulta dos SIG deve confiar em análises espaciais implícitas, o que não acontece em linguagens de consulta convencionais [EH93].

Grande parte das linguagens espaciais implementadas atualmente se baseia em *extensões de* SQL. [Ege92] demonstra a inadequação do uso desta linguagem de banco de dados como base para linguagens de consulta de alto nível em SIG.

Enquanto as extensões de SQL geram interfaces híbridas, onde a formulação de expressões é feita no formato convencional (textual) e a visualização de resultados é feita por meios gráficos e textuais, nas interfaces *visuais* tanto a formulação da consulta quanto a apresentação de seus resultados é feita através de recursos gráficos. Apesar desta abordagem parecer mais interessante, existem muitos problemas em aberto, como por exemplo a ambigüidade inerente ao uso de diagramas.

Dois princípios podem ser adotados para resolver os problemas de ambigüidade. Primeiro é possível recorrer ao usuário, aumentando a necessidade de diálogo. A segunda opção é adotar uma semântica *default* para cada caso de ambigüidade. Isto aumenta a complexidade do modelo de resolução de consultas, mas simplifica as ações do usuário.

SQL estendido *versus* Linguagens visuais		
	Vantagens	Desvantagens
SQL estendido	padronização da linguagem; tratamento e otimização de consultas alfanuméricas; facilidade de acoplamento com linguagens de programação	limitado pelo poder de expressão do modelo relacional; dificuldade na otimização de consultas envolvendo dados geométricos e alfanuméricos; consultas orientadas a redes são difíceis de se expressar
Linguagem visual	mais natural para o usuário final; facilita combinação de operações e reutilização de resultados para formulação de novas consultas	ausencia de padronização; complexidade para definição de consultas não triviais, especialmente as que envolvem negação

Figura 3.10: Os principais paradigmas de interação em SIG

A figura 3.10 resume as qualidades e pontos negativos das duas principais abordagens para linguagem de interação com SIG. Embora interfaces visuais estejam ganhando importância, as linguagens de consulta da maioria dos SIG usam uma *abordagem híbrida*

com formulação textual e apresentação visual de resultados de consultas. O poder de expressão das linguagens visuais terá que crescer muito antes que linguagens puramente visuais possam ser implementadas de modo a atender os requisitos dos sistemas geográficos. Ambientes cooperativos para consulta a BD geográficos representam um avanço recente no sentido de produzir linguagens que atendam a estes requisitos [CHF93].

3.5 Fatores Humanos e Cognição Espacial em SIG

O projeto de interfaces se baseia em suposições sobre os potenciais usuários de um *software*. O conjunto de hipóteses sobre o comportamento dos usuários e sobre a sua forma de raciocinar é representado por um *modelo mental de usuário* (ou simplesmente *modelo de usuário*). Pesquisas na área de fatores humanos em SIG se concentram na construção de modelos de usuário em aplicações geográficas. Com base no modelo de usuário são definidos o modelo de dados intermediário de interface e a linguagem externa que mapeia os conceitos do usuário em abstrações do modelo intermediário.

As pesquisas nesta área são inerentemente multidisciplinares, envolvendo ciências como a psicologia, a educação e a ergonometria [MSH93]. O restante desta seção apresenta alguns resultados relevantes que mostram a diversidade dos assuntos estudados nesta área. À primeira vista, os resultados apresentados parecem ter pouca coisa em comum. Alguns deles tratam diretamente dos problemas de projeto de interface; outros enfatizam o processo de cognição espacial. Apesar dos diferentes enfoques, os trabalhos apresentam um objetivo comum: o entendimento da percepção humana do espaço. Esta é a condição essencial para se construir um modelo mental de usuário de SIG.

Metáforas de Interface Geográfica

Segundo [Kuh91], uma metáfora de interface é um mapeamento (matemático) que parte de um domínio de origem e define uma estrutura para um domínio alvo. A escolha da metáfora em SIG especifica: (a) conceitos que o usuário pode manipular; (b) divisão de tarefas entre sistema e usuário; e (c) o tipo de comunicação que é adotada entre eles.

Grande parte das interfaces propostas para SIG se baseia na metáfora de mapa. De fato, mapas têm sido utilizados para representar informações espaciais desde os primórdios da civilização, e o sucesso desta metáfora em SIG se deve, em grande parte, à popularidade dos mapas de papel. Apesar da sua ubiqüidade, existem diversos problemas relacionados à utilização da metáfora de mapas em SIG: as únicas operações adequadamente suportadas são as de apresentação; mapas apresentam uma tendência a esconder a incerteza e a imprecisão inerente aos dados geo-referenciados; mapas são uma aproximação bidimensional de uma realidade tri ou tetradimensional (se a dimensão temporal for considerada).

Existem, no entanto, algumas características desta metáfora que precisam ser preservadas. Um exemplo é a qualidade de apresentação de resultados, herdada da cartografia.

[Kuh91] propõe o uso de uma nova metáfora em SIG: *apresentações como visões*. Esta metáfora explora propriedades do mecanismo de visualização humana nas questões que envolvem resolução espacial e mudança de escala. A ciência cognitiva afirma que o ser humano percebe o mundo em diferentes níveis de detalhe, e é justamente esta a abordagem proposta para a nova metáfora: analogia com o sistema de percepção visual humano. Este sistema envolve categorização do que é visto, e é dependente do ponto de vista espacial do observador. A vantagem desta metáfora é que o usuário passa a dispor de um processo dinâmico de visualização, ao invés de mapas estáticos.

Já [Gou93] propõe duas metáforas para o projeto de interface: a visão de manipulação de dados e a visão de mundo abstrato. Estas visões não são mutuamente exclusivas: dependendo da aplicação o foco em uma das visões pode ser mais indicado.

A *visão de manipulação de dados* está baseada em ferramentas de manipulação de dados. Estas devem explorar facilidades como múltiplas janelas, cores e gráficos. A *metáfora de toolkit* envolvida nesta visão é útil para usuários-programadores, capazes de combinar comandos e funções disponíveis em bibliotecas para realizar as tarefas desejadas.

A *visão de mundo abstrato* é uma metáfora mais adequada para análises científicas. A ênfase aqui está na modelagem cognitiva, com o objetivo de associar conceitos da interface aos conceitos usados naturalmente pelas pessoas. A principal diferença entre as modelagens de dados e cognitiva é que esta última não tenta predefinir o modelo mental (ou seja, a visão do mundo) mas apenas tenta descobrí-la e explorá-la. A *metáfora de desktop* é um exemplo de visão de mundo abstrato na área de escritórios. Uma metáfora equivalente para SIG poderia ser a bancada de trabalho de um geógrafo. Uma vantagem desta metáfora é que ela estabelece um contexto, restringindo as operações àquelas possíveis dentro de uma dada escala.

Entendimento do Mundo Geográfico

A realidade geográfica pode ser estudada através de duas visões fundamentais do espaço: a visão de *objeto* e a visão de *campo* [Cou92].

A visão de objeto se baseia em pontos, linhas e polígonos que representam entidades geográficas. Estes objetos geométricos possuem as propriedades dos objetos do mundo real: são discretos e possuem existência independente; possuem identidades permanentes; possuem atributos e formas; e podem ser manipulados (por exemplo, contados, movidos, coloridos). A maior parte dos pontos, linhas e polígonos que existem no mundo real se origina de artefatos humanos (trabalhos de engenharia e limites administrativos, por exemplo). O foco da visão de objeto se concentra nos relacionamentos espaciais (topológicos, projetivos, métricos) que podem existir entre as entidades geográficas.

A visão de campo adota uma abordagem oposta, baseada na abstração da natureza das entidades do mundo real. Qualquer fenômeno geográfico é representado por funções matemáticas, visualizadas graficamente como uma seqüência de *pixels*. Um pixel define o valor de cor a ser apresentado em um ponto da tela.

Dentre os fatores que determinam a cognição geográfica, dois podem ser destacados: escala e propósito (ou intenção humana). Cada combinação destes fatores tende para uma diferente perspectiva (objeto ou campo) da realidade geográfica. Nenhuma das visões pode ser considerada melhor, já que a cognição humana utiliza ambas em diferentes escalas e para diferentes propósitos.

O espaço também pode ser compreendido através da divisão em regiões. Ao contrário da maioria dos trabalhos, que estuda regiões naturais, [Gut92] analisa os fenômenos relacionados à ação humana, que podem dar origem a quatro tipos de regiões: (i) regiões formadas por variações naturais e culturais do espaço (exemplo: áreas francófonas); (ii) regiões originadas por visões ou aspirações sociais (exemplo: divisões políticas); (iii) regiões caracterizadas pela qualidade de vida (exemplo: áreas de poluição); e (iv) regiões para controle de problemas sociais ou ambientais (exemplo: parques ecológicos).

Esta taxonomia mostra que regiões são construções mentais que aparecem e desaparecem de acordo com o tipo de raciocínio utilizado. A ambigüidade é inerente ao termo *região*: ele se refere ao espaço absoluto e também a um sistema (social, econômico, cultural, geográfico) espacialmente referenciado. Este tipo de ambigüidade de termos e de conceitos precisa ser considerado no projeto de uma interface para SIG.

Linguagens e Comunicação em SIG

Uma interface de SIG atua como uma linguagem intermediária entre a linguagem formal e abstrata do SGBD subjacente e a linguagem empírica e informal do usuário [LS92]. O vocabulário da linguagem do BD é o conjunto de dados armazenado, e a gramática é o seu esquema conceitual. O *sistema de comunicação* entre usuário e SIG pode ser entendido como conjuntos de informações codificadas em alguma destas linguagens, e o processo de tradução entre elas (figura 3.11).

Cada processo do sistema de comunicação é estudado por uma ciência diferente, refletindo o caráter multi-disciplinar desta área de pesquisa. O estudo da percepção e ação humanas está centrado principalmente em aspectos de ergonometria e psicologia, enquanto o projeto do processo de consulta a BD está diretamente relacionado à ciência da computação. Já o mapeamento de informações do BD para a linguagem de saída do SIG tem sido tratado sob o ponto de vista da cartografia.

Na comunicação cartográfica, três grandes fontes de erros podem ser detectadas: decodificação incorreta (o leitor não entende a simbologia); codificação incorreta (uso errôneo de símbolos no mapa); e impedância de linguagem (a linguagem cartográfica não é capaz

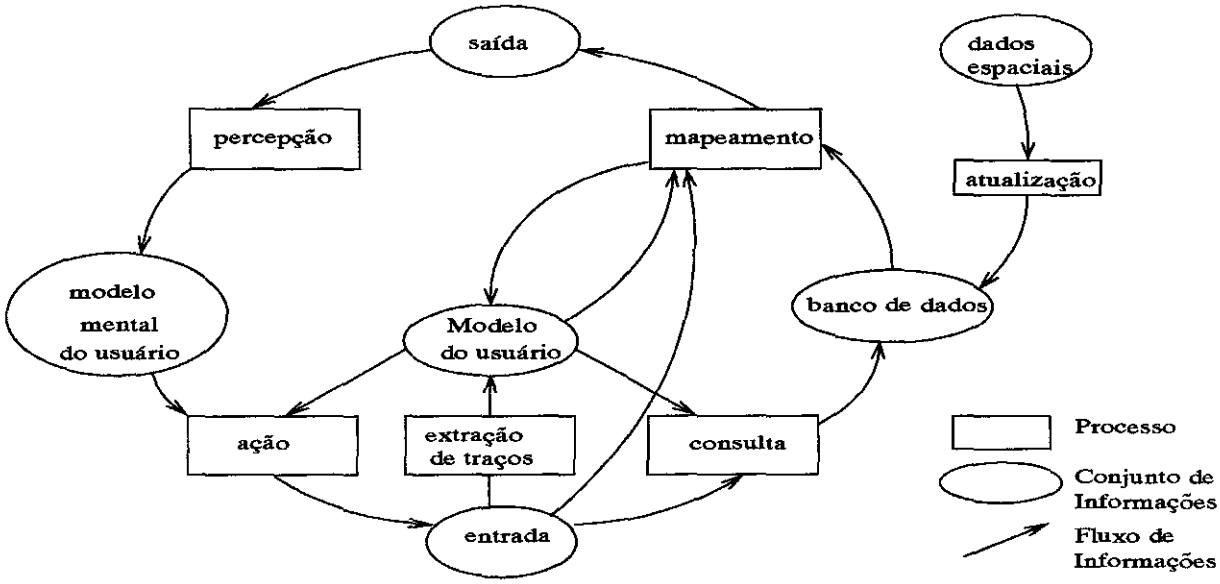

Figura 3.11: Sistema de comunicação em SIG, adaptado de [LS92]

de expressar o fenômeno desejado). Além disto, a comunicação cartográfica é unidirecional. Em SIG, o usuário pode modificar o conteúdo da apresentação, de modo que a comunicação se torna circular (figura 3.11).

Para lidar com significados de palavras do mundo real e semântica de aplicações é preciso que a linguagem de consulta espacial suporte *conceitos espaciais*, isto é, usuários devem ter acesso às idéias que eles geralmente usam quando pensam sobre o espaço. Estes conceitos espaciais podem ser simples (como "adjacência") ou complexos e dependentes da aplicação (tais como arcos em uma rede ligando pares de nós no espaço). Relacionamentos espaciais baseados em topologia, métrica, e ordem são essenciais em linguagens espaciais, embora a semântica destes relacionamentos varie com o sistema [EH93].

Modelo Mental de Usuário de SIG

Um modelo mental de usuário é uma representação explícita de propriedades de um certo usuário, sendo objeto de estudo em diversas ciências: computação, educação, psicologia, lingüística, ergonometria, entre outras. As linhas de pesquisa nesta área incluem: construção de modelo mental (estudo de conceitos, objetivos, e comportamento típico de um usuário); emprego de modelo mental (recuperação e apresentação adaptativa de informação; sistemas de ajuda); técnicas de inferência de modelo mental (métodos psicométricos; hierarquias de estereótipos; acompanhamento de sessões de trabalho); e aspectos pragmáticos (privacidade e consistência do modelo; avaliação empírica).

De modo geral, diversos fatores influenciam a definição de um modelo mental de

usuário: opções utilizadas; número e tipo de erros cometidos; tempo para tomada de decisões; eficiência na resolução de problemas; entendimento de conceitos e metáforas; e padrão de memorização. Do ponto de vista específico de interfaces, os aspectos importantes incluem: uso de cores e formatos; posicionamento de ícones; uso de gráficos e diagramas; paradigma de interação escolhido; e experiência no uso do sistema.

Uma função importante do modelo do usuário é prever suas ações e preferências, mudando a funcionalidade do sistema de maneira adequada. Entretanto, a adaptação automática pode ter seus inconvenientes: muitas pessoas não gostam do controle imposto pelo computador; um modelo de usuário simplista pode descartar informações de interesse; e o modelo mental pode ser um risco à privacidade do usuário.

Uma maneira de construir o modelo mental é através da observação do comportamento do usuário durante uma sessão de trabalho, para associá-lo a estereótipos (pares $<atributo, valor>$ descrevendo as características do usuário – idade e experiência, por exemplo) [LS92]. Alguns atributos podem disparar outros estereótipos, que podem ser organizados em hierarquias, com o usuário "canônico" na raiz. O estereótipo de um usuário pode ser construído a partir da combinação e modificação de estereótipos da hierarquia.

Três fatores pragmáticos precisam ser considerados na definição de um modelo mental em SIG: a informação que o usuário deseja, para restringir o universo de interação; a informação que o usuário já possui, para deduzir o contexto de trabalho; e a habilidade do usuário de inferir informações. Estes fatores determinam os dados que devem ser apresentados ao usuário, e ajudam na definição dos estereótipos.

Utilização e Construção de Mapas

A interação com mapas em papel é diferente da interação com mapas em SIG. Um mapa em papel representa uma realidade permanente e tangível enquanto mapas em SIG são subdivididos em mapas visíveis mas não permanentes (mapas na tela); e mapas não diretamente visíveis mas permanentes (mapas nos BD). O processo de construção de mapas é uma atividade cartográfica que tem sido objeto de pesquisa por muito tempo. O processo de uso de mapas, por outro lado, é uma área de estudo recente [Woo93].

O processamento de mapas tenta controlar os estágios de percepção durante a leitura de mapas, identificando como os elementos da imagem são detectados, progressivamente reconhecidos através de ligações com a memória de longa duração, e combinados em grupos de símbolos com maior significado. O modelo do sistema de processamento de informação visual humano é usado para representar diferentes modos de processamento.

Outro conceito interessante, que une noções cognitivas e de bancos de dados, é o de *mapa dinâmico*, introduzido em [AKK94]. Mapas dinâmicos têm três propriedades: (a) reflexão imediata de mudanças nos BD; (b) seleção de dados a partir de consultas genéricas; e (c) adaptação da apresentação a mudanças contextuais. As duas primeiras

propriedades estão presentes em muitas interfaces de SIG. A terceira propriedade está apoiada no conceito de *visão* aplicado a bancos de dados e a apresentações gráficas. Em bancos de dados, visões são definidas como consultas, enquanto no contexto de gráficos elas são definidas por procedimentos de visualização.

Linguagens de consulta são ferramentas adequadas para definição de visões de BD, mas não permitem o controle das visões gráficas representadas na tela pelos objetos visuais. Para resolver este problema, [AKK94] separa objetos *conceituais* de objetos de *apresentação*. Diversos objetos de apresentação podem se referir ao mesmo objeto conceitual, em diferentes métodos de visualização. Mudanças em objetos conceituais são propagadas para os objetos de apresentação correspondentes. Um par <*visão do* BD, *método de visualização*> define cada camada visual de um mapa dinâmico.

3.6 Resumo

Este capítulo mostrou que a construção de interfaces em SIG pode ser particionada em três áreas de estudo: arquiteturas, linguagens e fatores humanos. A primeira área tenta minimizar a distância semântica entre o modelo de dados dos SIG e o modelo intermediário da interface. As outras duas áreas usam diferentes abordagens para minimizar a distância entre o modelo mental do usuário e o modelo intermediário da interface.

O projeto da interface deve estabelecer uma solução de compromisso entre estas necessidades conflitantes. Não há uma solução genérica para todos os problemas envolvidos; nem mesmo as soluções *ad hoc* atendem os diversos requisitos de usuários. As principais soluções propostas na literatura em cada área de pesquisa foram aqui analisadas, levando às seguintes conclusões:

- Arquiteturas existentes para interfaces de SIG falham na definição de pelo menos um dos seguintes aspectos: integração entre interface e SIG, funcionalidade e interoperabilidade dos módulos; modelo intermediário de representação e mapeamentos entre modelos de dados; divisão das tarefas entre interface e SIG.

- Embora interfaces visuais estejam ganhando importância, as linguagens de consulta existentes em SIG usam uma abordagem híbrida envolvendo formulação textual e apresentação visual de resultados. O poder de expressão das linguagens visuais puras precisa ser melhorado antes que estas linguagens sejam adotadas em SIG.

- Muitas propostas de interface não levam em consideração o seu aspecto multidisciplinar. Em um extremo estão as soluções computacionalmente eficientes, mas que ignoram a importância da adaptação ao perfil do usuário da interface. No outro extremo estão os trabalhos que propõem modelos de usuário muito próximos do

modelo cognitivo humano, mas que são inviáveis sob o ponto de vista de implementação em um computador. Projetos bem sucedidos precisam encontrar uma solução de compromisso entre estes dois extremos.

Estas observações sobre projeto e desenvolvimento de interfaces para SIG foram utilizadas como ponto de partida para as propostas desta tese. O próximo capítulo introduz o arcabouço teórico que suporta a construção de interfaces para aplicações geográficas, levando-se em consideração as idiossincrasias apresentadas neste capítulo.

Capítulo 4

Arquitetura para Interfaces Geográficas

4.1 Apresentação

Apesar da complexidade inerente à construção de interfaces geográficas, não existe uma infra-estrutura adequada para suportar este processo. De fato, grande parte dos desenvolvimentos é feita de maneira *ad hoc*, ou utilizando apenas ferramentas de interface genéricas. Para diminuir o custo de desenvolvimento de interfaces geográficas são necessários mecanismos que suportem uma *abordagem sistemática* para este processo.

Este capítulo descreve uma *arquitetura genérica para interfaces geográficas* que viabiliza tal abordagem. A implementação de interfaces de acordo com a organização proposta promove melhorias na modularização do *software* (reduzindo a complexidade e a interdependência entre código da interface e da aplicação), na manutenção corretiva e evolutiva deste *software*, e na reutilização de componentes de interface.

A próxima seção traz definições utilizadas no capítulo. A seção 4.3 dá a motivação para a nova arquitetura, cujos princípios básicos são apresentados na seção 4.4. A seção 4.5 traz diretivas de projeto que orientam a utilização da arquitetura. As camadas da arquitetura são analisadas nas seções 4.6 (Camada de Ligação), 4.7 (Camada de Modelos de Dados), e 4.8 (Camada de Aplicação). A seção 4.9 sintetiza o capítulo.

4.2 Definições

Conceitos de Orientação a Objetos

Não existe uma definição formal unanimemente aceita para um modelo de dados orientado a objetos [ABD+89, ZM90, BM91, BF94, Dit94]. Este texto se baseia no modelo orientado

a objetos de [Bee89]. Todo **objeto** é uma **instância** de alguma classe e é caracterizado por seu *estado* (conjunto de valores de atributos) e *comportamento* (conjunto de operações ou métodos que podem ser aplicados ao objeto). Todas as instâncias de uma classe possuem a mesma estrutura e comportamento.

Um objeto o pode ser composto por outros objetos $o_1,....,o_n$, caso em que o é chamado **complexo** e $o_1,....,o_n$ são os *componentes* de o. Este processo de composição é realizado através da aplicação de **construtores** - por exemplo, construtores de conjunto - que permitem a especificação progressiva de objetos cada vez mais complexos a partir de componentes previamente definidos. Objetos não complexos são denominados *simples*.

As **classes** são estruturadas em hierarquias de herança: os ancestrais de uma classe C na hierarquia são as *superclasses* de C e seus descendentes as *subclasses* de C. Os descendentes de uma classe **herdam** sua estrutura e comportamento. Um objeto é instância de sua classe, e é considerado **membro** de todas as suas superclasses. A diferença entre um *membro* e uma *instância* de uma classe é que aquele pode possuir outras propriedades além daquelas definidas na classe (diretamente ou por herança).

Aplicações e Interfaces Geográficas

Dois tipos de programas podem ser identificados em um sistema computacional: *programas de sistema* e *programas de aplicação*. Os primeiros realizam tarefas genéricas que oferecem subsídios para implementação dos últimos, que por sua vez realizam funções de interesse específico de um grupo de usuários. A classificação de um programa em uma das categorias depende do ponto de vista. Por exemplo, um programa que implementa uma função de análise espacial em um SIG pode ser considerado um programa de aplicação (ou simplesmente uma *aplicação*) quando comparado a um programa do sistema operacional que o suporta, mas é um programa de sistema sob a visão de um programa que utiliza tal função do SIG para realizar uma tarefa definida pelo usuário.

Utilizando os conceitos do capítulo 2, este texto considera uma **aplicação geográfica** como um programa de aplicação interativo que utiliza os recursos do SIG para manipular dados convencionais e geo-referenciados. O termo **interface geográfica**, ou simplesmente *interface*, é utilizado para designar o Componente Interativo de uma aplicação geográfica, enquanto o seu Componente Semântico é chamado **Núcleo Semântico** (seção 2.3). A figura 4.1 ilustra estes conceitos, situando uma aplicação geográfica no contexto do *software* subjacente.

As funções disponíveis ao usuário podem ser classificadas em dois grupos: as **funções de interface**, que podem ser processadas independentemente pelo Componente Interativo; e as **funções semânticas**, que envolvem processamentos complexos ou transformações semânticas dos dados. Funções de interface incluem, por exemplo, as funções que mudam a forma de apresentação dos dados para o usuário. Funções semânticas são

implementadas no Núcleo Semântico; em uma aplicação voltada para redes de transportes, por exemplo, uma função que calcula o caminho mínimo entre dois pontos (utilizando as funções de análise espacial do SIG) é uma função semântica. O Componente Interativo tem dois papéis: prover funções de interface, e servir de intermediário entre o usuário e o Núcleo Semântico para permitir a execução de funções semânticas.

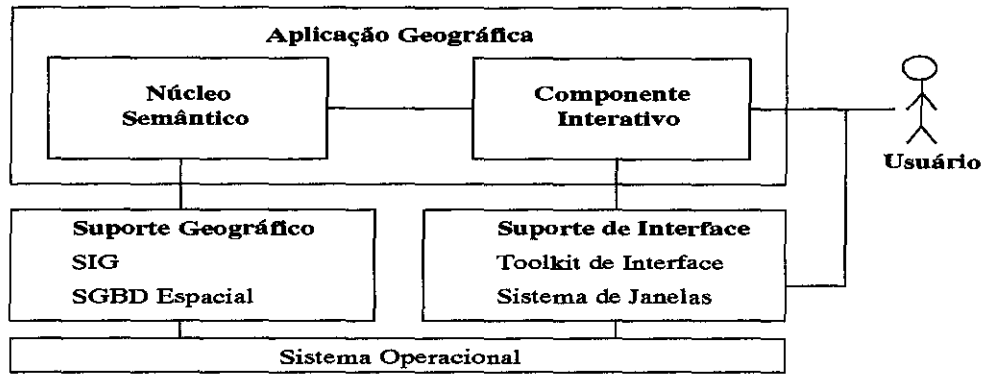

Figura 4.1: Contexto de uma aplicação geográfica

Aplicações geográficas utilizam as funcionalidades dos SIG da mesma forma como aplicações convencionais se beneficiam das facilidades oferecidas pelos SGBD. Em geral, uma aplicação se comunica com o SIG (ou SGBD) subjacente através de uma API (*Application Program Interface*, isto é, um protocolo de comunicação que serve de interface para um programa de aplicação). Ainda, lembra-se ao leitor que a proposta da tese está voltada a aspectos de interface para SAG (Sistemas de Aplicações Geográficas) conforme definido na seção 1.4.

4.3 Limitações das Arquiteturas Atuais

Existe um consenso na comunidade científica com relação às vantagens de se considerar separadamente os componentes interativo e semântico de uma aplicação [HS89, BC91, NMK91, Edm92, Mye95]. Entre outros benefícios, a divisão de uma aplicação em Componente Interativo e Núcleo Semântico promove a especialização e otimização de cada componente, oferece a possibilidade de integração de interfaces personalizadas, permite uma organização lógica da aplicação e o encapsulamento de dados e funções, facilita a manutenção corretiva e evolutiva, e possibilita a reutilização de componentes de *software*.

As principais arquiteturas de *software* de interface apresentadas na literatura, discutidas no capítulo 2, estabelecem como princípio fundamental a separação entre os componentes interativo e semântico da aplicação. No entanto, estas arquiteturas geralmente apresentam propostas de separação em um nível muito abstrato, de modo que a imple-

mentação da funcionalidade especificada é uma tarefa complexa. Arquiteturas modulares, por exemplo, são imprecisas no que diz respeito aos protocolos de comunicação entre módulos, enquanto arquiteturas baseadas em agentes não definem de modo claro quais componentes do sistema devem ser agrupados em um agente.

No caso de aplicações geográficas, além das dificuldades inerentes à manipulação de dados geo-referenciados, é preciso integrar diversos tipos de sistemas de suporte (SIG e *toolkits* especializados, por exemplo) para compor o ambiente de *software* das aplicações. Apesar das arquiteturas genéricas (tais como PAC, Slinky ou Seeheim) ajudarem na organização geral das aplicações, elas são inadequadas para interfaces geográficas porque:

- não levam em consideração as idiossincrasias das aplicações geográficas (seção 1.3.1). Por exemplo, as arquiteturas gerais trabalham com múltiplas visões de um objeto exclusivamente no nível de apresentação, enquanto dados geo-referenciados podem ter múltiplas representações também no banco de dados;
- não atendem aos requisitos particulares de organização do *software* geográfico. Em particular, as arquiteturas não consideram a necessidade de integração com diferentes tipos de *software* de suporte (somente *toolkits* de interface).
- não facilitam a transição da fase de projeto para a fase de implementação.

As arquiteturas específicas para interfaces geográficas, discutidas na seção 3.3, tentam resolver estes problemas, mas apresentam soluções limitadas em diversos aspectos. Algumas destas arquiteturas são direcionadas para um determinado tipo de SIG e propõem soluções *ad hoc* ([Env92] e [AYA+92], por exemplo). As arquiteturas de interface geográfica que visam independência de SIG, por outro lado, estabelecem restrições funcionais sobre o SIG ou sobre o Componente Interativo:

1. Restrições sobre o SIG: exigem que certos tipos de serviço sejam oferecidos pelo SIG para que ele possa ser utilizado na arquitetura de interface. Exemplos destes serviços incluem funções para suporte à manipulação de dados geo-referenciados (como o dicionário de *legendas de mapas* de [Voi91, Voi94]), e funções para integração da interface à arquitetura do SIG (como na proposta de [PMP93], em que a interface é uma camada integrada em uma arquitetura de SIG).

2. Restrições sobre o Componente Interativo: ao invés de restringirem o SIG, certas arquiteturas de interface impõem limites sobre os tipos de funções disponíveis ao usuário. A idéia é minimizar a interação entre aplicação geográfica e SIG através da redução do conjunto de funções intercambiadas, e também reduzir a complexidade desta interação eliminando, por exemplo, funções com efeitos colaterais nos

dados armazenados. Em geral, apenas funções de consulta podem ser definidas nas interfaces construídas com estas arquiteturas [Rig95, OM95].

A próxima seção apresenta uma nova arquitetura de *software* para desenvolvimento de interfaces geográficas. Como as arquiteturas anteriores, ela divide o *software* em um conjunto de componentes abstratos inter-relacionados. No entanto, ao contrário de suas predecessoras, a arquitetura torna explícito o processo de refinamento destes componentes, até atingir o nível de programas que efetuam a funcionalidade prevista.

4.4 Visão Geral da Arquitetura

A arquitetura de *software* proposta nesta tese sintetiza os pontos positivos das principais arquiteturas de interface (genéricas e geográficas) existentes e introduz novos conceitos para resolver as dificuldades discutidas na seção anterior. O objetivo da arquitetura é atender às necessidades específicas de SAG, definindo, entre outros aspectos:

- diretivas de projeto que orientam o projetista no uso da arquitetura em cada etapa do desenvolvimento de uma interface geográfica;
- um mecanismo de ligação entre os componentes semântico e interativo da aplicação geográfica que garante independência mútua;
- o modo de interação entre todos os componentes da arquitetura, através de protocolos de comunicação uniformes;
- módulos adicionais de suporte à construção da interface, com suas respectivas responsabilidades e funcionalidades, produzindo serviços reutilizáveis pelas diversas aplicações do SAG;
- módulos adicionais para integração com o *software* de suporte, visando obter independência de SIG e portabilidade de interface com relação ao *toolkit*.

Uma premissa fundamental desta arquitetura é o suporte à construção do *software* de interface das aplicações de um SAG ao longo de todas as fases do seu ciclo de vida (análise, projeto, implementação e manutenção). Para isto, a arquitetura estabelece uma solução de compromisso entre *especificações abstratas*, ideais para o projeto, e *especificações detalhadas*, necessárias para implementação, provendo suporte para especificação dos aspectos de engenharia de *software* necessários para implementar o projeto.

4.4.1 O Modelo de Dados Intermediário

Uma arquitetura de interface geográfica deve definir um modelo de dados intermediário que possa representar os conceitos espaciais usados pelo usuário, mapeando-os para aqueles efetivamente implementados no SIG. O modelo de dados adotado nesta arquitetura está baseado no GMOD, um modelo orientado a objetos voltado para as necessidades de aplicações geográficas [OPM97], que estende o modelo de dados geográfico de quatro níveis introduzido em [CFS+94].

Os quatro níveis de abstração propostos no GMOD são: o mundo real (os fenômenos geográficos); o nível conceitual (uma visão abstrata dos fenômenos, onde as operações são independentes da representação dos dados); o nível de representação (onde as operações são especializadas para cada representação de uma entidade geográfica); e o nível físico (que trata do armazenamento e recuperação eficientes dos dados geo-referenciados). Além de permitir a modelagem de dados propriamente dita, o GMOD também engloba a modelagem dinâmica e de processos, o que permite refletir aspectos evolutivos do mundo real. O modelo intermediário da arquitetura de interface utiliza apenas os níveis conceitual e de representação do GMOD, os quais são descritos a seguir. Mais detalhes sobre o GMOD podem ser encontrados em [Pir97].

Nível Conceitual

No nível conceitual, entidades geo-referenciadas são classificadas em geo-objetos (visão de objetos) e geo-campos (visão de campos). Cada classe tem suas próprias operações de alto nível (*operações conceituais*) e suas representações específicas.

Os três aspectos fundamentais do nível conceitual são (1) as classes (no sentido do paradigma de orientação a objetos); (2) os relacionamentos (permitindo diversos tipos de ligação entre classes); e (3) restrições estáticas e dinâmicas (impostas sobre classes, relacionamentos e objetos).

A figura 4.2 mostra os principais componentes do nível conceitual do modelo de dados GMOD. As classes representadas como retângulos mais escuros auxiliam na organização do banco de dados espacial para cada aplicação/projeto de usuário. As classes dentro do pontilhado formam o núcleo do modelo geográfico, com destaque para a Geo-Classe, cujas instâncias possuem componentes espaciais, e para a Classe Convencional, cujos objetos não são geo-referenciados. Estas classes podem ser especializadas de acordo com os conceitos utilizados em cada tipo de aplicação. Por exemplo, em uma aplicação de telefonia, a classe Poste especializa a Geo-Classe, enquanto a classe Assinante é uma especialização de Classe Convencional. Os possíveis relacionamentos entre as classes do modelo são indicados na caixa situada no canto superior direito da figura 4.2.

O modelo propõe duas especializações da Geo-Classe: Geo-Objeto e Geo-Campo, para

descrever, respectivamente, as visões de objetos e de campos do mundo real. Ambas as subclasses possuem obrigatoriamente um *atributo de localização* que descreve a região a que cada instância se refere. O domínio deste atributo é definido por outra classe básica do modelo: a Geo-Região. Esta classe modela a dimensão espacial da Geo-Classe, descrevendo uma área da superfície terrestre, de acordo com uma determinada escala e projeção. A Geo-Região encapsula métodos espaciais abstratos, isto é, independentes da representação utilizada para armazenar a descrição do conceito espacial.

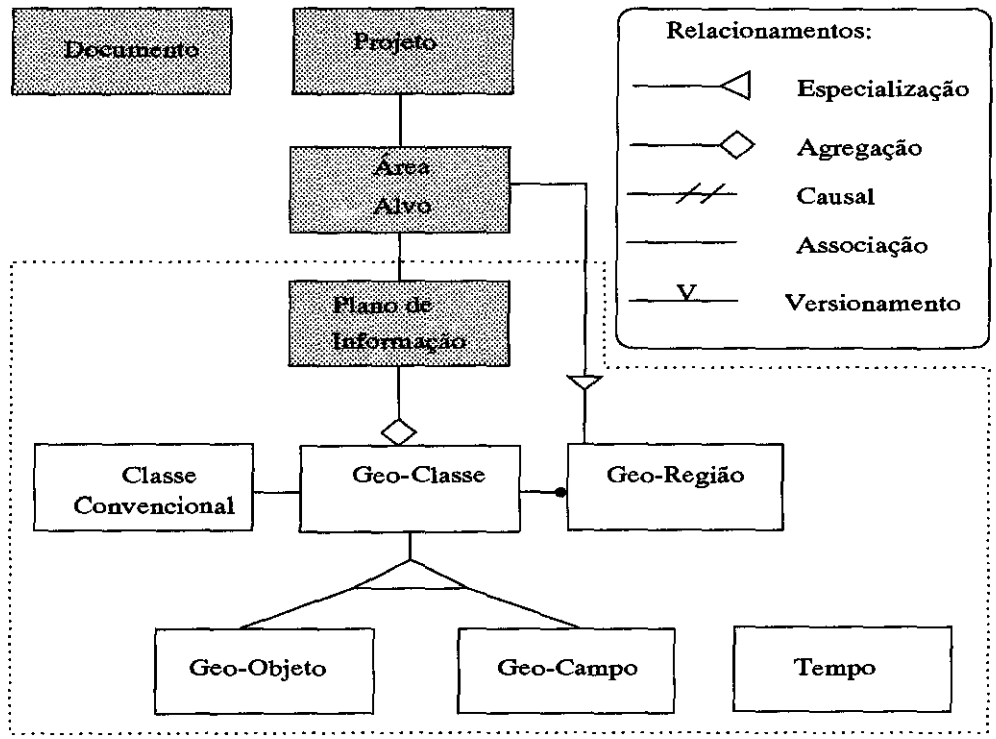

Figura 4.2: O nível conceitual do modelo GMOD

A modelagem geográfica envolve dois tipos de relacionamentos espaciais: explícitos (que precisam ser especificados pelo usuário) e implícitos (que podem ser derivados das representações geográficas). O GMOD está voltado para a representação de relacionamentos explícitos. Estes relacionamentos podem ser de diferentes tipos. Além das associações simples, agregações, e especializações, comuns ao paradigma de orientação a objetos, existem ainda dois relacionamentos especificamente direcionados para aplicações geográficas: versionamento e causalidade.

Relacionamentos de versionamento permitem determinar conexões entre versões de um mesmo conceito. Um importante tipo de versionamento é o imposto pela evolução temporal. Por exemplo, dois objetos de classes distintas podem podem ser ligados porque eles representam a evolução de um determinado fenômeno ao longo do tempo. Versionamento

também pode ser usado para ligar diferentes representações de uma mesma entidade, e para construção de diferentes cenários envolvendo uma realidade geográfica.

Relacionamentos de causalidade estabelecem ligações de causa e efeito na modelagem de um determinado fenômeno. Por exemplo, o tipo de solo de uma dada área afeta diretamente as características de vegetação desta área. Relacionamentos de causalidade são comuns em atividades de planejamento ambiental, que estudam a influência e o impacto de atividades humanas sobre a natureza.

Relacionamentos de causalidade e de versionamento são inerentemente temporais. Por exemplo, a curva de maturação de uma dada plantação pode variar ao longo do tempo de acordo com propriedades do solo ou do uso de fertilizantes. Da mesma forma, um dado fenômeno pode ocorrer apenas após uma certa atividade humana (tal como a poluição da água causada por lixo industrial). Desta forma, o GMOD permite a temporalização de relacionamentos de causalidade e versionamento. Tanto estes relacionamentos quanto as classes do modelo podem ser temporalizados através de referências à classe *Tempo*. Esta, por sua vez, pode ser especializada para diferentes características temporais (tais como intervalos discretos ou contínuos).

Nível de Representação

O nível de representação leva em consideração detalhes relacionados à representação geométrica e topológica das propriedades espaciais de entidades geográficas definidas no nível conceitual. Uma determinada propriedade espacial pode estar associada a diferentes representações, correspondendo, por exemplo, a diferentes escalas cartográficas.

Cada modelo de representação é fortemente influenciado pelas técnicas e formatos utilizados para a captura dos dados geo-referenciados (como por exemplo as imagens de satélite ou o levantamento de amostras em pesquisas de campo).

As classes do nível de representação são um refinamento das classes do nível conceitual. As classes Geo-Objeto e Geo-Campo possuem classes correspondentes no nível de representação (Rep-Objeto e Rep-Campo, respectivamente), conforme mostra a figura 4.3.

A classe Rep-Objeto utiliza uma hierarquia de classes geométricas que permitem definir pontos, linhas e polígonos, e seus inter-relacionamentos espaciais (topologia, por exemplo). Este tipo de representação é geralmente denominado de *representação vetorial*.

A classe Rep-Campo provê subclasses para definir as diferentes representações de campo utilizadas pelas aplicações. Entre estas representações estão incluídas as grades regulares e irregulares, os mapas de contorno e de pontos amostrais, e as subdivisões planares. As representações utilizadas na classe Rep-Campo são conhecidas como *representações matriciais*.

Assim, o GMOD suporta o paradigma de representação múltipla, em que um dado

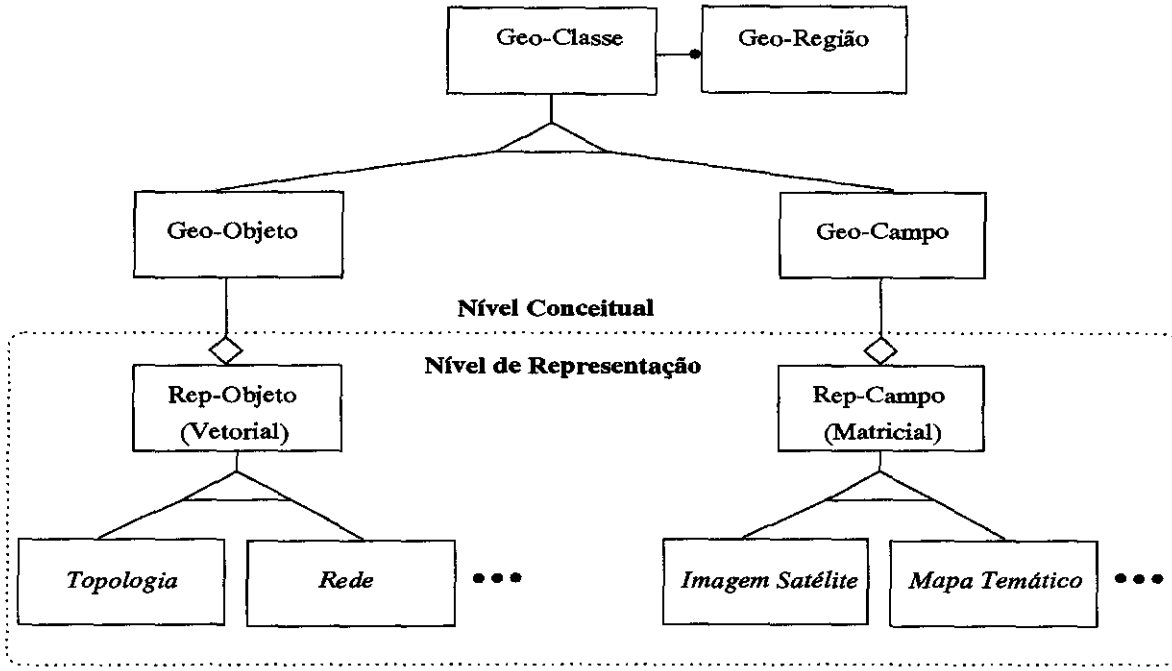

Figura 4.3: Relacionamento entre os níveis do modelo GMOD

fenômeno geográfico pode ser percebido de diferentes maneiras, de acordo com as necessidades de cada aplicação. Esta característica, no nível de interface, leva em consideração dois importantes requisitos dos usuários: a manipulação transparente de representações múltiplas; e a definição de diferentes apresentações para cada representação. A utilização do GMOD na arquitetura de interface proposta é discutida a seguir.

O GMOD tem outros dois componentes (modelo dinâmico e modelo de processos), mas esta tese se atém aos detalhes necessários para a construção de interfaces. Os capítulos 6 e 7 apresentam exemplos de interação do usuário, tais como a manipulação de postes telefônicos em uma região. Do ponto de vista do GMOD, isto significa criar uma classe de Geo-Objetos Poste, associada a uma Geo-Região onde as instâncias de Poste estão posicionadas. No nível de representação, uma instância de Poste será representada vetorialmente por objetos do tipo Ponto. Estes, por sua vez, podem ser armazenados como coordenadas (X, Y). O GMOD permite, desta forma, refinar progressivamente a definição de entidades geográficas.

4.4.2 Componentes da Arquitetura

A arquitetura de *software* apresentada em [OM95] estabelece uma infra-estrutura para o desenvolvimento de interfaces de navegação e consulta em diferentes SIG. A arquitetura de *software* de interface introduzida nesta tese é uma generalização daquela proposta, permitindo o desenvolvimento de interfaces geográficas de propósito geral (e não apenas de ferramentas de consulta). A arquitetura define três componentes principais, organizados

em camadas sobrepostas, conforme mostra a figura 4.4.

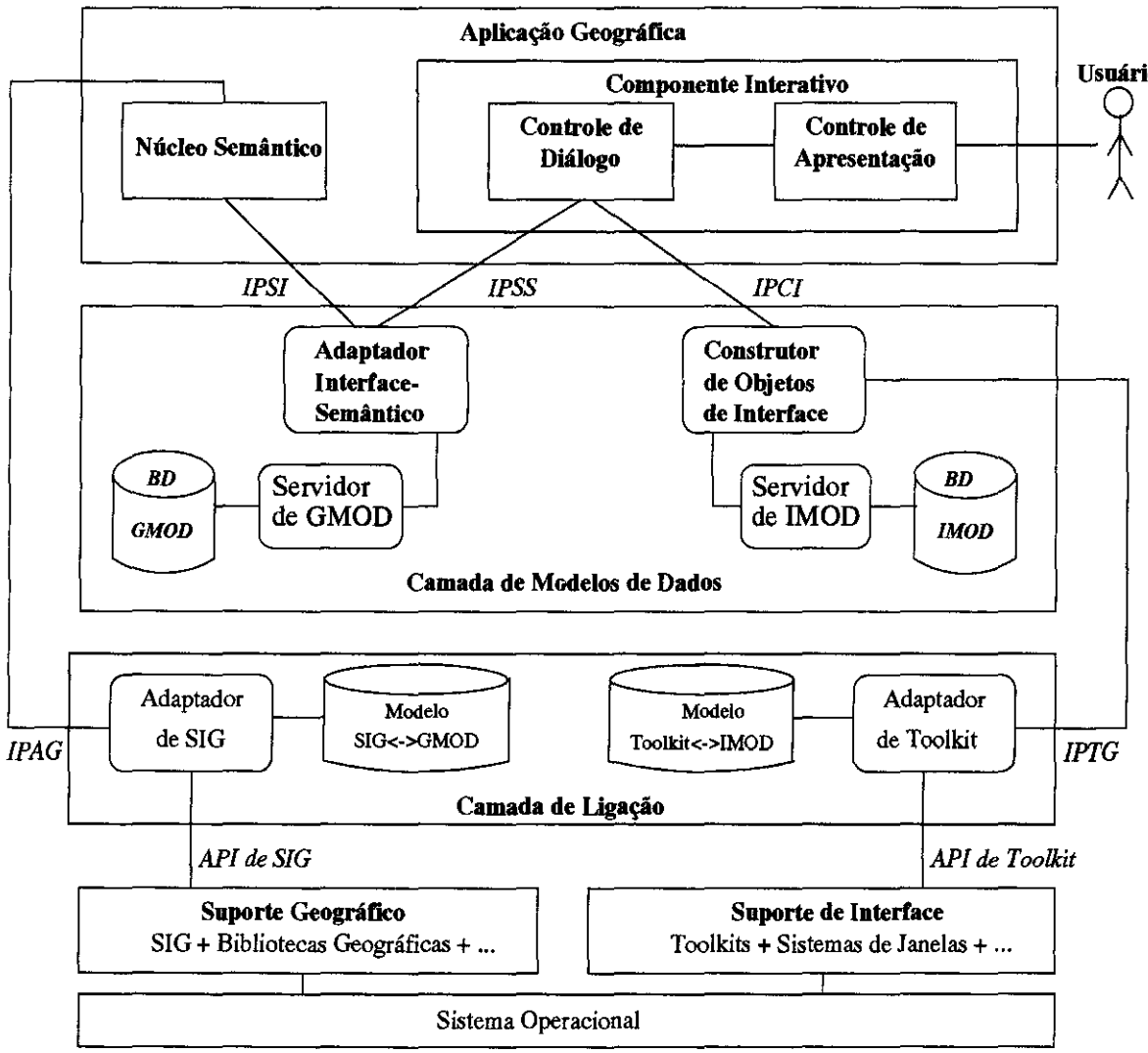

Figura 4.4: Arquitetura de *software* de interface geográfica

Em interfaces desenvolvidas de maneira *ad hoc*, os diversos componentes funcionais estão embutidos em um único módulo de *software*. Este bloco monolítico torna-se extremamente complexo, reunindo em um único local todos os aspectos do problema (comunicação com o SIG, modelo de dados intermediário e de interação, conversão entre modelos, ligação entre Componente Interativo e Núcleo Semântico, visualização de dados geo-referenciados, entre outros).

Por outro lado, a estruturação do *software* de interface em camadas, proposta pela arquitetura, promove a modularização e a especialização dos componentes, simplificando o desenvolvimento, e estabelecendo uma divisão clara de tarefas e responsabilidades.

Cada camada é responsável pelo gerenciamento de um conjunto bem definido de tarefas: comunicação com os sistemas de suporte subjacentes (Camada de Ligação); gerenciamento de modelos de dados utilizados na interface (Camada de Modelos de Dados); e implementação da aplicação geográfica propriamente dita, ou seja, das funções semânticas e de interação com o usuário (Camada de Aplicação).

Camada de Ligação

A camada inferior da arquitetura é responsável pela comunicação com os sistemas de suporte do SAG. Entre estes sistemas existem dois que se destacam para efeito da tese: o SIG subjacente e o *toolkit* de interface adotado.

Para cada sistema de suporte utilizado pelo SAG, a Camada de Ligação provê um **módulo adaptador** responsável por oferecer às camadas superiores da arquitetura um acesso normalizado e independente do sistema de suporte adotado. Um módulo adaptador tem três grandes objetivos:

- Garantir portabilidade: a portabilidade é obtida pela capacidade de utilizar qualquer sistema de suporte que possa ser conectado com o adaptador. Todas as dependências da aplicação geográfica com relação ao sistema subjacente estão restritas ao módulo adaptador, que funciona como um *device driver* para este sistema.

- Promover a reutilização de *software*: isto acontece no contexto global da arquitetura, pois todas as aplicações de diversos SAG podem compartilhar adaptadores.

- Apoiar as tarefas de manutenção: a facilidade de manutenção do sistema decorre do encapsulamento do sistema de suporte, de modo que qualquer modificação neste sistema afeta apenas o (modelo de mapeamento do) adaptador, e não tem efeitos colaterais nos demais módulos da arquitetura.

Cada módulo adaptador define, basicamente, uma máquina abstrata com uma interface de programação uniforme. Desta forma, as camadas superiores utilizam a mesma interface de comunicação para qualquer sistema subjacente. Adaptadores para os sistemas de suporte destacados anteriormente são exemplificados a seguir.

Adaptador de SIG O adaptador de SIG pode utilizar os mecanismos de integração de interfaces a SGBD propostos em [OA93b]. A idéia principal é estabelecer um protocolo de comunicação entre o adaptador e o SIG subjacente com base em um conjunto de *operações primitivas* definidas sobre o modelo de dados intermediário da interface (GMOD).

Os módulos deste adaptador fazem a tradução de esquemas, dados e operações do modelo intermediário para o modelo de dados do SGBD do SIG subjacente, e vice-versa.

Para isto o adaptador mantém um *modelo de mapeamento* que define a correspondência entre conceitos do modelo do SIG e os do GMOD.

A comunicação com o SIG é feita através de sua API, se o SIG define uma, ou diretamente com os módulos de gerência de dados do SIG, caso contrário. Portanto, a arquitetura utiliza duas abordagens distintas de integração com o SIG: a integração é *fraca* com SIG *abertos* (com API) e *forte* com SIG *fechados* (sem API). É importante notar que apenas este adaptador é dependente do SIG subjacente. Ele oferece para o restante da arquitetura uma API chamada IPAG (Interface de Programação para Aplicação Geográfica) e se encarrega de traduzir este protocolo de comunicação para o mecanismo de comunicação oferecido pelo SIG.

Adaptador de *Toolkit* O Adaptador de *Toolkit* é um módulo que permite definir objetos de interface (e objetos de interação associados) de acordo com o modelo de objetos de interface proposto para a arquitetura (modelo IMOD, definido no próximo capítulo).

Embora cada *toolkit* possua representações diferentes para objetos de interação, de acordo com suas regras de *look-and-feel*, existe um conjunto básico de tipos de objetos de interação que é oferecido pela maioria dos *toolkits*. O Adaptador de *Toolkit* define este conjunto de tipos fundamentais de *widgets* e implementa regras de mapeamento para *widgets* do *toolkit* adotado. Este mapeamento estabelece a correspondência entre a API IPTG (Interface de Programação de *Toolkit* Geográfico) e a API do *toolkit* utilizado.

A vantagem do uso do adaptador é a independência com relação ao *toolkit* permitindo, por exemplo, utilizar um *toolkit* diferente para portar a interface para outra plataforma computacional. Esta mesma idéia aparece na arquitetura de *software* de interface genérica proposta em [BFL+92], e também na implementação de *toolkits* multi-plataforma [Val89, MMM+97].

Camada de Modelos de Dados

Esta camada é responsável por manter dois bancos de dados: um para os dados intercambiados entre Componente Interativo e Núcleo Semântico, visando permitir o processamento da aplicação geográfica (BD GMOD); outro para os dados intercambiados entre Componente Interativo e usuário, para suporte à interface geográfica (BD IMOD). Cada BD é organizado de acordo com um modelo de dados diferente, e está associado a um servidor de BD que gerencia o acesso aos dados respectivos.

Embora mostrados logicamente na Camada de Modelos de Dados, ambos bancos de dados podem ser implementados diretamente no SGBD geográfico do SIG subjacente.

Suporte ao GMOD O primeiro BD utiliza o modelo intermediário da interface (GMOD) e contém os dados da aplicação que são manipulados no Componente Interativo. Cada

aplicação define seu próprio esquema para este banco de dados. Tanto o Núcleo Semântico quanto o Componente Interativo pode ter acesso aos dados gerenciados pelo servidor de GMOD através de um **Adaptador Interface–Semântico**. Este adaptador define os principais aspectos da interação entre o Componente Interativo e o Núcleo Semântico (nível de abstração dos dados transferidos, componente proprietário de cada dado, mecanismo de descrição de dados intercambiados, e tipo de acesso permitido).

O Núcleo Semântico e o Componente Interativo se comunicam com o adaptador através de duas API específicas: IPSI (Interface de Programação de Serviços de Interface) e IPSS (Interface de Programação de Serviços Semânticos), respectivamente.

Suporte ao IMOD O segundo modelo de dados gerenciado pela Camada de Modelos de Dados é o modelo de objetos de interface IMOD, que oferece facilidades para criar diferentes apresentações dos dados contidos no BD GMOD. Este modelo, descrito no próximo capítulo, representa os objetos que são usados no diálogo com o usuário, os quais são definidos, de maneira abstrata, através da API IPCI (Interface de Programação do Construtor de Interface).

O módulo **Construtor de Objetos de Interface** permite compor objetos de interface complexos a partir de objetos mais simples recuperados do BD IMOD com o auxílio do servidor de IMOD. O construtor utiliza a API IPTG para solicitar do Adaptador de *Toolkit* os *widgets* correspondentes a um determinado objeto de interface complexo.

Camada de Aplicação

Enquanto as camadas inferiores estabelecem mecanismos de suporte, esta camada define o *software* da aplicação geográfica propriamente dita. Como ocorre em todas as arquiteturas de *software* de interface, a aplicação é dividida em Núcleo Semântico e Componente Interativo.

Componente Interativo Este componente é responsável pela interação (ou diálogo) com o usuário, envolvendo dois módulos principais: Controle de Diálogo e Controle de Apresentação.

O módulo de **Controle de Apresentação** realiza duas tarefas fundamentais: (1) a transformação de ações do usuário (isto é, eventos originados pelo usuário) em operações sobre os objetos de interface do BD IMOD; e (2) a gerência de apresentações gráficas e textuais dos dados contidos neste BD.

O módulo de **Controle de Diálogo** é o *cérebro* do Componente Interativo, sendo responsável pela definição do comportamento dinâmico da interface, incluindo o controle do próprio módulo de Controle de Apresentação. Por exemplo, um objeto de interação pode estar associado a um ou mais objetos da aplicação definidos segundo o GMOD. O

módulo de Controle de Diálogo gerencia estes relacionamentos entre objetos de interação e objetos de aplicação.

O Componente Interativo de uma aplicação geográfica forma a parte da arquitetura de *software* de interface que deve ser tratada especificamente sob a ótica das teorias computacionais de interface homem–computador. O capítulo 5 apresenta a proposta desta tese para construção deste componente.

Componente Semântico da Aplicação

Como já foi mencionado, os componentes interativo e semântico da aplicação precisam se comunicar, e por esta razão o Componente Semântico precisa ser definido na arquitetura de *software* de interface. No entanto, o Núcleo Semântico não é, obviamente, um módulo da interface geográfica, assim como não o é o SIG subjacente.

A única restrição definida na arquitetura de interface a respeito do Núcleo Semântico visa regulamentar a comunicação entre este e o Componente Interativo especificado pela arquitetura. Esta comunicação deve, idealmente, garantir autonomia aos componentes. Na literatura, esta autonomia é denominada de *independência de diálogo* [HH89].

4.4.3 Interoperabilidade

A figura 4.4 mostra linhas rotuladas (IPAG, IPSI, IPSS, IPTG) ligando as camadas da arquitetura. Cada linha representa um canal de comunicação entre módulos de camadas distintas da arquitetura, e o conjunto de canais define os protocolos de comunicação e a interoperabilidade entre os componentes da arquitetura.

Um canal implementa uma API entre um módulo *servidor*, que oferece um conjunto de funções, e um módulo *cliente*, que consome as funções oferecidas. Cada módulo define os serviços oferecidos para clientes e as funções utilizadas de outros servidores.

As API são os únicos canais de comunicação entre diferentes camadas da arquitetura. Isto contribui para o encapsulamento dos módulos, facilitando o desenvolvimento, a manutenção, a evolução, e a reutilização do *software*. O princípio é análogo ao conceito de *encapsulamento* do paradigma de orientação a objetos.

Na prática, cada componente especificado pela arquitetura é composto de outros módulos mais simples que cooperam entre si para implementar a funcionalidade definida para o componente complexo. A arquitetura não estabelece regras para criação de sub-módulos, nem para comunicação entre eles. Os servidores de modelos de dados, por exemplo, oferecem serviços a clientes situados na mesma camada, mas o projetista é livre para determinar a melhor solução de implementação, de acordo com as prioridades estabelecidas para cada projeto.

Os requisitos de um projeto determinam os sub-módulos e os protocolos de comunicação entre eles. Este tipo de flexibilidade é fundamental para atender às necessidades de diferentes SAG; no entanto, é preciso observar que a arquitetura impõe um contexto para que o projetista possa exercer sua criatividade. Ao contrário das arquiteturas genéricas que ficam apenas no nível mais abstrato, esta arquitetura define os pontos em que pode haver decisões de projeto e aqueles em que regras precisam ser obedecidas. Desta forma, a arquitetura estabelece uma solução de compromisso entre a liberdade do projetista e a garantia de atendimento às bases de organização propostas.

Interface de Programação para Aplicação Geográfica (IPAG)

Este canal de comunicação oferece aos clientes na Camada de Aplicação uma API para um SIG capaz de suportar os conceitos do GMOD. O responsável pelo oferecimento destes serviços é o Adaptador de SIG da Camada de Ligação. As informações que fluem através da API incluem solicitações e respostas aos seguintes serviços:

- Estabelecer e encerrar conexão com um SIG;
- Abrir e fechar determinado esquema de BD;
- Executar uma expressão das linguagens de definição/manipulação de dados;
- Obter metadados sobre os BD (descrição de esquema, hierarquia de classes, composição, métodos, atributos);
- Realizar serviços de transação (*commit* e *rollback*, por exemplo).

O Adaptador de SIG converte estes serviços para o formato de comunicação disponível no SIG subjacente.

Interface de Programação de *Toolkit* Geográfico (IPTG)

Assim como o Adaptador de SIG oferece uma API independente do SIG subjacente, o Adaptador de *Toolkit* oferece uma API para um *toolkit* genérico, que oferece os seguintes serviços:

- Inicialização e finalização do sistema de execução (*run-time*) do *toolkit*;
- Criação e destruição de *widgets*;
- Consulta e atualização de valores de atributos (também conhecidos como recursos) de *widgets*;

O Adaptador de *Toolkit* traduz estes serviços genéricos para funções específicas da API do *toolkit* subjacente.

Interface de Programação do Construtor de Interface (IPCI)

O Construtor de Interface oferece facilidades para gerenciamento de objetos complexos de interface através da API IPCI. Os serviços oferecidos incluem a criação, instanciação, consulta, modificação e remoção de modelos de objetos de interface definidos segundo o IMOD.

O Construtor de Objetos de Interface mapeia os serviços da API IPCI que envolvem manipulação de objetos de interação (*widgets*) para a API IPTG. A seção 5.5 descreve o funcionamento deste Construtor.

Interface de Programação de Serviços de Interface (IPSI) e Interface de Programação de Serviços Semânticos (IPSS)

Os problemas envolvidos na especificação do Núcleo Semântico de uma aplicação geográfica incluem aspectos de modelagem conceitual e de engenharia de *software* que não fazem parte do escopo da arquitetura de interface. O leitor interessado nas dificuldades envolvidas na modelagem conceitual de aplicações geográficas encontrará subsídios em [OPM97, Pir97]. A arquitetura de interface geográfica precisa definir apenas o *mecanismo de comunicação* entre o Componente Interativo e o Núcleo Semântico, visando garantir a troca de informações sem comprometer a independência de diálogo.

O Núcleo Semântico se comunica com o Componente Interativo exclusivamente através de operações definidas nas API do Adaptador Interface–Semântico. A API IPSI define todas as funções de interface que podem ser invocadas pela aplicação. Analogamente, a API IPSS define todas as funções semânticas que são visíveis para o Componente Interativo. Para tanto, estas API definem um serviço de exportação de funções cuja semântica é informar ao Adaptador Interface–Semântico sobre as funções de um componente que podem ser invocadas pelo outro. Outro serviço predefinido nas duas API é o de manipulação do modelo de dados intermediário.

Três pontos precisam ser ressaltados com relação ao mecanismo de comunicação entre Componente Interativo e Núcleo Semântico. Primeiro, o Adaptador Interface–Semântico encapsula todos os pontos de interação entre o Componente Interativo e Núcleo Semântico, garantindo a propriedade de *independência de diálogo*. Além disto, o Adaptador pode ser utilizado para estabelecer um mecanismo de propagação de mudanças efetuadas sobre os dados compartilhados pelo Componente Interativo e Núcleo Semântico, conforme discute a seção 5.5.2.

Segundo, a separação entre funções semânticas e de interface permite a implementação de diálogo *concorrente* (ou *multi-tarefa*). Neste tipo de diálogo, etapas de diferentes tarefas podem ser realizadas intercaladamente. Isto é possível porque o Controle de Diálogo pode chamar funções semânticas (através do Adaptador Interface–Semântico) de maneira

assíncrona, de forma que a interface continue disponível para que o usuário realize outras tarefas.

O terceiro ponto diz respeito ao *controle de execução* da interface, isto é, o seqüenciamento lógico de eventos no Componente Interativo. Existem quatro tipos básicos de controle [HH89]: *por aplicação* (no qual o controle é estabelecido pelo Componente Semântico); *baseado em diálogo* (no qual a seqüência de ações depende dos eventos gerados pelo usuário); *misto* (no qual tanto o usuário quanto o Núcleo Semântico podem tomar a iniciativa das ações); e *balanceado* (no qual um componente de controle global coordena as ações invocadas pelo usuário e pela aplicação). A arquitetura deixa para o projetista da interface a decisão do tipo de controle a ser adotado. O mecanismo de comunicação baseado em funções exportadas e em um modelo de dados comum (o modelo intermediário da interface) permite a implementação de todos estes tipos de controle.

4.5 Diretivas de Projeto das Camadas

A organização em camadas definida na arquitetura determina os principais componentes da interface geográfica, estabelecendo uma relação do tipo cliente/servidor entre uma camada e as camadas inferiores. Esta organização estrutural serve de base para o projeto de novas interfaces geográficas segundo diretivas que determinam (a) os componentes que precisam ser definidos nos novos projetos, e (b) os componentes reutilizáveis que já foram desenvolvidos para projetos anteriores. As diretivas definem três etapas de desenvolvimento da interface geográfica:

1. Construção da Camada de Ligação.

2. Construção da Camada de Modelos de Dados.

3. Construção da Camada de Aplicação.

As diretivas definem a maneira de instanciar cada módulo das diferentes camadas da arquitetura, determinando um modo sistemático de desenvolvimento.

4.5.1 Exemplo de Interação

Antes de discutir a construção de cada camada, é preciso ter uma visão clara das interações envolvidas entre elas. O exemplo apresentado a seguir ilustra, de uma forma esquemática, uma interação típica entre um usuário e uma aplicação geográfica construída segundo a arquitetura proposta. O objetivo deste exemplo é mostrar os diversos tipos de trocas de informação e as interdependências entre os componentes da arquitetura.

Considere uma aplicação geográfica em que o usuário pode atualizar informações geo-referenciadas através da indicação de objetos de interesse em um mapa (apresentado em uma janela da interface). Ressalta-se que todos os objetos apresentados são resultado da aplicação de uma apresentação a objetos previamente selecionados e que já se encontram no BD GMOD.

- O usuário solicita uma operação interagindo com objetos de interface (a janela que contém o mapa e os símbolos gráficos que representam objetos geográficos). O módulo de Controle de Apresentação gerencia estes objetos e comunica ao módulo de Controle de Diálogo a ocorrência de uma ação do usuário. Por exemplo, aquele módulo pode converter um evento de "*double click*" em uma operação de seleção sobre um objeto (neste caso, a seleção se aplica sobre a representação gráfica de um geo-objeto).

- O módulo de Controle de Diálogo gerencia a correspondência entre objetos de interface (BD IMOD) e objetos de aplicação (BD GMOD). O módulo mantém ainda um *contexto de interação* (o "*estado*" da interface) que permite a identificação da operação a ser realizada em resposta à ação do usuário. Neste exemplo, o Controle de Diálogo detém o conhecimento que indica que a seleção do geo-objeto no contexto de interação corrente ativa as seguintes operações:

  1. Uma solicitação de consulta sobre o estado corrente do objeto ao Adaptador Interface–Semântico (via IPSS).

     O Adaptador Interface–Semântico pode executar diretamente esta operação, pois os dados do geo-objeto já se encontram no BD GMOD (uma vez que o objeto já estava apresentado na janela de mapa).

     Se esta suposição não fosse verdadeira, o Adaptador deveria repassar a consulta ao Núcleo Semântico (via IPSI), e este buscaria o dado no Adaptador de SIG (via IPAG). Este último módulo é que traduziria a consulta para o comando correspondente na API do SIG subjacente. O resultado da consulta seria, então, enviado no caminho inverso (SIG → Adaptador de SIG→ Núcleo Semântico → Adaptador Interface–Semântico → BD GMOD).

  2. Uma solicitação de apresentação do estado do objeto ao Construtor de Objetos de Interface (via IPCI).

     O Construtor recebe do Controle de Diálogo uma descrição do tipo de apresentação (neste caso, uma janela de edição) e os valores do estado do objeto a ser apresentado. Usando estas informações o Construtor busca no BD IMOD um modelo de criação deste tipo de janela, e constrói uma janela complexa, que é enviada ao Adaptador de *Toolkit* (via IPTG).

O Adaptador de *Toolkit* traduz a definição da janela complexa de apresentação para a API do *toolkit* subjacente, criando efetivamente os *widgets* da janela. Estes são repassados no caminho inverso (*toolkit* → Adaptador de *Toolkit* → Construtor de Interface → Controle de Diálogo → Controle de Apresentação) e constituem o núcleo do módulo de Controle de Apresentação.

3. Uma mudança no contexto de interação indicando a presença desta nova apresentação, e as possíveis ações sobre ela.

   O Controle de Diálogo é responsável por manter o "*estado*" da interface. Em cada estado, diferentes tipos de funções de interface podem estar disponíveis, e diferentes conjuntos de objetos de interação podem estar ativos. O Controle de Diálogo define as modificações necessárias para representar o estado resultante da criação da nova janela de apresentação.

- O usuário realiza as alterações desejadas sobre a apresentação do geo-objeto e confirma a operação. Novamente, o evento de usuário (acionamento de um botão) é traduzido pelo Controle de Apresentação em uma chamada ao Controle de Diálogo, o qual se encarrega de traduzir a operação para uma solicitação de alteração do BD GMOD junto ao Adaptador Interface–Semântico.

- O Adaptador Interface–Semântico faz a validação semântica da alteração solicitada, de acordo com as restrições expressas no esquema GMOD. O Adaptador pode ainda solicitar validação semântica adicional ao Núcleo Semântico, no caso de aplicações que possuem restrições que não podem ser expressas segundo o GMOD, tais como aquelas que dependem da semântica da aplicação. Se algum erro foi detectado, o Controle de Diálogo é informado; caso contrário, o Adaptador processa a solicitação da seguinte maneira:

  1. Atualiza o estado do geo-objeto no BD GMOD.
  2. Sinaliza para o Núcleo Semântico (via IPSI) a mudança ocorrida sobre o geo-objeto.

- O Núcleo Semântico toma as devidas providências para garantir que a atualização seja refletida em suas estruturas internas e no SIG (via IPAG).

As diretivas de projeto, discutidas a seguir, levam em consideração estes fluxos de informação ao longo dos componentes da arquitetura, visando obter módulos que sejam genéricos (para garantir reutilização), e ao mesmo tempo eficientes no cumprimento de suas tarefas específicas.

4.5.2 Construção da Camada de Ligação

Os dois módulos da Camada de Ligação são totalmente independentes entre si. Cada módulo precisa ser definido uma única vez *para cada sistema de suporte subjacente*, especificando o mecanismo de conversão entre a API que o Adaptador oferece para as demais camadas da interface e a API real do sistema de suporte adotado.

No caso do Adaptador de SIG é preciso capturar o modelo de dados do sistema geográfico subjacente através de um modelo de conversão entre o GMOD e o modelo de dados utilizado no SIG. Esta é uma tarefa complexa devido à variedade de modelos de dados existentes em SIG (seção 1.2.3).

O Adaptador de *Toolkit*, por outro lado, trabalha com diferentes tipos de sintaxe para expressar, em geral, os mesmos conceitos (tais como *widgets*, eventos e *callbacks*). No entanto, a implementação deste módulo também enfrenta dificuldades como, por exemplo, a manutenção do mesmo *look-and-feel* da interface em diferentes plataformas.

A Camada de Ligação é a parte da arquitetura de interface que depende das características dos sistemas subjacentes. Estes sistemas oferecem suporte de *run-time* à interface. Em contrapartida, a camada pode ser reutilizada por todas as aplicações que utilizam os mesmos sistemas de suporte. A seção 4.6 mostra os detalhes necessários para o acoplamento da arquitetura de interface aos sistemas de suporte através da camada de ligação.

4.5.3 Construção da Camada de Modelos de Dados

A Camada de Modelos de Dados, discutida na seção 4.7, também é composta de dois módulos ortogonais: o Adaptador Interface–Semântico, responsável pela comunicação entre Componente Interativo e Núcleo Semântico, e o Construtor de Objetos de Interface, que gerencia objetos de interface utilizados no diálogo com o usuário.

A comunicação entre Componente Interativo e Núcleo Semântico é intermediada pelo módulo Adaptador Interface–Semântico, com base em um BD organizado de acordo com o modelo de dados GMOD. O Adaptador permite especificar um esquema de BD comum ao Núcleo Semântico e ao Componente Interativo. Os dados efetivamente utilizados pela aplicação geográfica são armazenados neste BD, sobre o qual são executadas as funções exportadas pelos dois componentes. Todas as aplicações geográficas compartilham o módulo Adaptador Interface–Semântico, embora cada SAG defina esquemas GMOD próprios para suas aplicações.

O módulo Construtor de Objetos de Interface também é compartilhado por todas as aplicações geográficas. A sua tarefa é gerenciar modelos de objetos de interface complexos. Para isto o Construtor mantém uma biblioteca de definições de objetos, organizada de acordo com o modelo IMOD. Cada aplicação geográfica utiliza diferentes modelos de

objetos desta biblioteca, conforme discute o próximo capítulo.

A Camada de Modelos de Dados oferece módulos de suporte ao projeto e à execução da interface geográfica. Estes módulos são utilizados pelos projetistas, no desenvolvimento de novas interfaces, e pelos componentes da aplicação, para efetuar serviços em tempo de execução do SAG. Os serviços disponibilizados pela Camada de Modelos de Dados permitem o gerenciamento dos dois modelos de dados fundamentais para a arquitetura de interface: o modelo intermediário da interface (BD GMOD) e o modelo de objetos de interface (BD IMOD).

4.5.4 Construção da Camada de Aplicação

O Componente Interativo de uma aplicação geográfica é implementado na Camada de Aplicação. Os módulos de Controle de Diálogo e de Controle de Apresentação deste componente refletem o modelo mental de usuário concebido pelo projetista de interface. Um exemplo típico do tipo de conceito envolvido neste modelo é a definição da reação a ser produzida para um determinado evento de interface.

O Componente Interativo precisa ser instanciado pelo menos uma vez *para cada aplicação geográfica*. Vale notar que mais de uma instância deste componente pode ser criada para um mesmo Núcleo Semântico. Isto seria útil, por exemplo, para prover interfaces diferenciadas para grupos heterogêneos de usuários. Técnicas de reutilização e de personalização de interfaces podem ser utilizadas para diminuir o custo da construção desta camada, conforme mostram os capítulos 5 e 6.

4.5.5 Análise do Processo de Construção da Interface

As três diretivas que guiam o desenvolvimento dos módulos da interface mostram que, embora as camadas de Modelos de Dados e de Ligação sejam bastante complexas, o seu custo é diluído em função da reutilização em um número potencialmente grande de aplicações geográficas.

A Camada de Aplicação, por sua vez, representa um custo de desenvolvimento maior, por duas razões. Primeiro, esta camada trata dos problemas de interação com o usuário propriamente ditos, que no caso de aplicações geográficas são particularmente difíceis. Segundo, a camada deve ser instanciada para cada aplicação individual do SAG. Grande parte do esforço desta tese está voltado para reduzir o custo desta camada.

A construção de interfaces de acordo com as diretivas propostas representa ganhos consideráveis com relação ao desenvolvimento *ad hoc*. Além de promover a modularização da aplicação, a arquitetura torna possível uma reutilização substancial dos componentes que tratam da organização dos dados e da comunicação com os sistemas de suporte. Em

projetos tradicionais, todos estes componentes precisariam ser redefinidos (ou copiados) para cada nova aplicação desenvolvida.

4.6 A Camada de Ligação

Cada instância de um módulo Adaptador da Camada de Ligação implementa um *driver* externo que efetua a comunicação com um tipo específico de sistema de suporte. A dificuldade de implementação do Adaptador varia de acordo com o grau de equivalência existente entre conceitos utilizados na arquitetura e aqueles existentes no sistema subjacente. Dois tipos de sistema de suporte são importantes no contexto de aplicações geográficas: SIG e *toolkits* de interface.

4.6.1 O Adaptador de *Toolkit*

A ligação entre Componente Interativo e *toolkit* já foi bastante pesquisada na área de interfaces homem-computador. Praticamente todas as arquiteturas de interface levam em consideração este tipo de sistema de suporte. O mecanismo de ligação proposto pela arquitetura Slinky [BFL+92] se mostra bastante adequado aos objetivos da arquitetura proposta nesta tese. Assim, o módulo Adaptador de *Toolkit* da arquitetura implementa, basicamente, o *Componente de Apresentação* da arquitetura Slinky.

A descrição sucinta deste componente, apresentada em [BFL+92], é a seguinte:

> *"A mediation, or buffer, component between the Dialogue and the Interaction Toolkit Components that provides a set of toolkit-independent objects for use by the Dialogue Component (e.g., a "selector" object that can be implemented in the toolkit using either a menu or radio buttons)".*

Os objetos de interface independentes de *toolkit* são descritos através do modelo IMOD, e o Adaptador de *Toolkit* transforma estes objetos em *widgets* (objetos de interação) suportados pelo *toolkit* subjacente. Esta abordagem é semelhante àquela utilizada nos *toolkits virtuais* [Mye95]. A idéia é que o código relativo aos objetos de interação seja feito de acordo com o *toolkit* virtual, e este código seja mapeado, automaticamente, para o código do *toolkit* utilizado para implementação da interface em uma plataforma específica.

Existem dois tipos de *toolkits* virtuais, de acordo com o mecanismo utilizado para implementar *widgets* em diferentes plataformas. Os *toolkits* virtuais baseados em *mapeamento* utilizam algum tipo de ligação com as funcionalidades de *toolkits* convencionais disponíveis na plataforma adotada. Já os *toolkits* virtuais baseados em *replicação* reimplementam os *widgets* destes *toolkits* em cada plataforma. XVT [Val89] e Amulet

[MMM+97] são exemplos, respectivamente, de *toolkits* virtuais baseados em mapeamento e em replicação.

Ambas as abordagens possuem deficiências e pontos fortes [Mye95]. Na abordagem de mapeamento existe uma dificuldade em oferecer funcionalidades que não estão presentes em todos os *toolkits* subjacentes. Um problema na abordagem de replicação é o crescimento das bibliotecas de *run-time*, com possível diminuição de desempenho. A arquitetura de interface proposta nesta tese deixa a cargo do projetista a escolha da solução mais adequada para os objetivos e requisitos de cada tipo de SAG.

4.6.2 O Adaptador de SIG

Ao contrário do que acontece com os *toolkits*, existe pouco trabalho desenvolvido sobre a integração entre SIG e interface. As abordagens de integração podem ser classificadas de acordo com o isolamento entre funções dos dois componentes (seção 3.3.1). A abordagem proposta nesta tese utiliza um módulo da arquitetura – o Adaptador de SIG – exclusivamente para efetuar esta integração (figura 4.5).

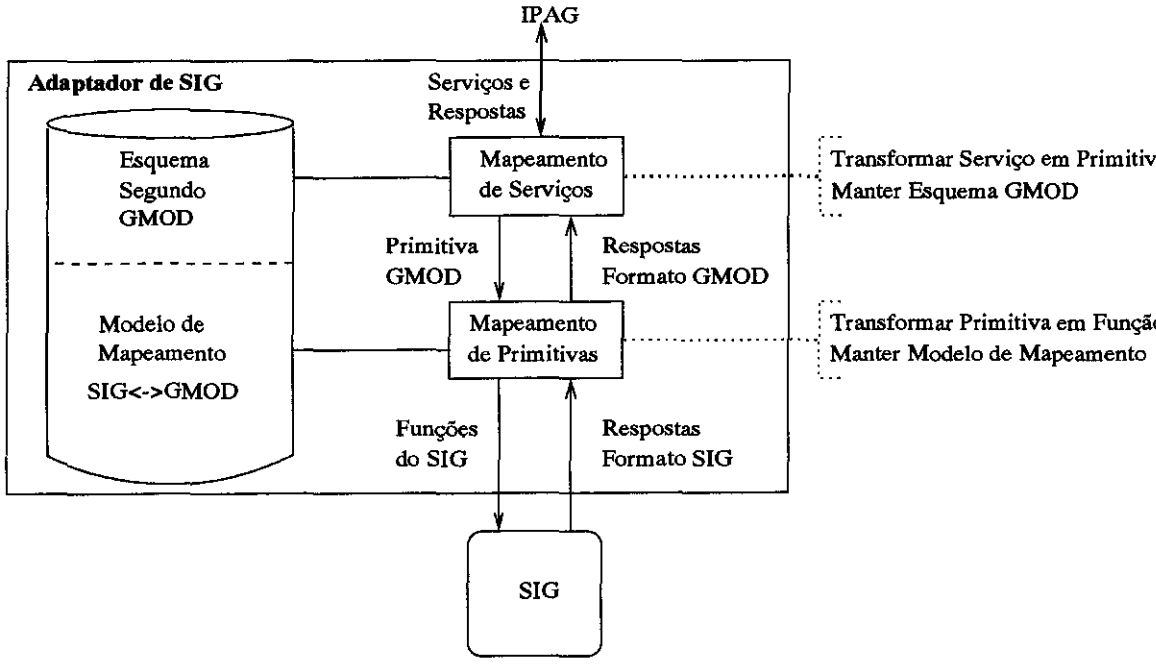

Figura 4.5: Adaptador de SIG da arquitetura

O Adaptador de SIG oferece uma API para um SIG virtual que suporta o modelo GMOD. Isto é feito através do mapeamento entre o modelo de dados do SIG e o GMOD, sendo a complexidade do Adaptador inversamente proporcional ao grau de similaridade entre estes modelos de dados. O Adaptador mantém um BD próprio (BD SIG↔GMOD) que permite mapear conceitos e arquivos do SIG em conceitos e classes do GMOD. Este BD também

armazena informações sobre os esquemas de BD do SIG, necessárias para construir os esquemas GMOD correspondentes, sob a forma de tabela de correspondências.

O Adaptador solicita funções do SIG usando um protocolo baseado em operações primitivas, originalmente utilizado para o acoplamento de um sistema de interface genérico a diferentes SGBD orientados a objetos [OA93b]. As operações primitivas são expressas quer de acordo com a sintaxe da API do SIG (integração fraca), quer utilizando os módulos gerenciadores de BD do SIG (integração forte para SIG sem API).

Operações Primitivas

As operações primitivas propostas para comunicação com o SIG são *Get-Schema*, *Get-Class-Extension*, *Get-Value*, e *Exec-Function*. Elas foram projetadas como um conjunto minimal de funções que devem ser supridas por um SIG baseado em SGBD orientado a objetos para que um usuário possa ter acesso e controle sobre os dados armazenados. Estas operações são definidas como operações sobre esquemas (metadados) e sobre objetos dos bancos de dados geográficos. A semântica das operações primitivas é definida pela arquitetura da interface, enquanto o modo de implementá-las é próprio de cada SIG. Cada operação manipula dados (esquemas ou extensões) que lhe são passados pelo Adaptador a partir do mapeamento realizado no BD SIG↔GMOD.

A operação *Get-Schema* obtém do SIG a definição de um esquema de BD geográfico. Ela usa como parâmetro de entrada o nome de um esquema de BD e recebe a definição deste esquema sob a forma de (1) uma lista de classes (com seus respectivos tipos, assinaturas de métodos, e superclasses); (2) uma lista de operações/funções definidas sobre estas classes. *Get-Schema* permite que o Adaptador mantenha os grafos de herança e de composição do esquema no seu BD de mapeamento.

A operação *Get-Class-Extension* retorna uma lista contendo todos os identificadores de objetos (*oids*) que correspondem a instâncias de uma determinada classe de um BD. O nome da classe e o nome do esquema são recebidos como parâmetros da operação.

A operação *Get-Value* representa uma consulta ao conteúdo (estado) de um objeto. A operação deve receber como parâmetro apenas o identificador de objeto que se deseja consultar. Como resposta, o SIG envia o conteúdo do objeto solicitado, que é repassado para o cliente que solicitou o serviço de consulta usando informação de esquema para interpretar os valores recebidos. Os valores são armazenados no BD GMOD e os metadados para mapeamento são armazenados no BD SIG↔GMOD.

A operação *Exec-Function* possibilita invocar as funções do SIG subjacente para análises e transformações nos dados armazenados. Dois tipos de funções podem ser solicitadas do SIG: funções definidas como métodos de uma classe; e funções independentes, definidas no esquema do BD. O Adaptador transforma estes dois tipos de funções em operações do SIG. Esta tarefa abrange a conversão e passagem de parâmetros, a detecção do término

da operação junto ao SIG, e o recebimento de mensagens e resultados.

É claro que tanto o comando a ser submetido quanto o formato dos resultados produzidos são dependentes do SIG utilizado. A adaptação da arquitetura de interface a diferentes SIG consiste em implementar um Adaptador de SIG capaz de: (a) mapear operações primitivas sobre o modelo intermediário (GMOD) para a sintaxe dos comandos em cada SIG; e (b) monitorizar a execução de funções do SIG, convertendo as respostas de/para o formato utilizado pelo modelo de dados intermediário. A figura 4.5 esquematiza este processo.

Uso e Funcionalidade do Adaptador de SIG

Embora a figura 4.5 mostre um único SIG subjacente, a arquitetura é perfeitamente capaz de lidar com diversos SIG. Isto depende apenas da implementação, no Adaptador de SIG, de *Módulos de Mapeamento de Primitivas* apropriados. Existem duas maneiras para utilização de diferentes SIG como suporte a uma aplicação geográfica:

1. Utilização isolada: apesar de haver diferentes SIG disponíveis, a aplicação escolhe um determinado SIG para sua execução. Neste caso, um controle adicional (originado pela aplicação, através do serviço de Conexão da API IPAG, por exemplo) deve definir a instância do Módulo de Mapeamento de Primitivas a ser ativado. A partir desta ativação, o Adaptador de SIG funciona como se houvesse apenas um Módulo de Mapeamento de Primitivas.

2. Utilização concorrente: a aplicação opta por utilizar um SIG diferente para realizar tarefas distintas. Desta forma, informações adicionais são necessárias para seleção do SIG utilizado em cada tarefa. Dois tipos de controle podem ser empregados para definir o SIG a ser utilizado. No *controle por aplicação*, a informação sobre o SIG é passada como parâmetro em cada operação solicitada através da API IPAG. No *controle automático* o próprio Módulo de Mapeamento de Serviços do Adaptador de SIG decide o SIG apropriado para realizar cada operação (com base, por exemplo, no tipo de representação do dado utilizado na operação).

A figura 4.6 mostra esta situação: cada SIG está acoplado a um Módulo de Mapeamento de Primitivas, e o Módulo de Mapeamento de Serviços precisa realizar a tarefa adicional de selecionar o (ou permitir a seleção do) SIG a ser utilizado.

A utilização concorrente dos Módulos de Mapeamento de Primitivas permite o suporte a aplicações que exigem o uso de dados gerenciados por mais de um SIG: cada módulo mapeia operações primitivas para um determinado SIG. Em outras palavras, uma operação primitiva op_i pode ser mapeada para um SIG Si enquanto outra operação primitiva op_j definida sobre o mesmo esquema de BD pode ser mapeada para um SIG Sj. Desta maneira, é possível explorar as características de diferentes SIG para implementar a aplicação

geográfica. Um exemplo típico seria uma aplicação que utiliza as visões de campos e de objetos do espaço geográfico. A arquitetura permite o uso de dois SIG especializados, respectivamente, em manipulação de representações vetoriais e matriciais.

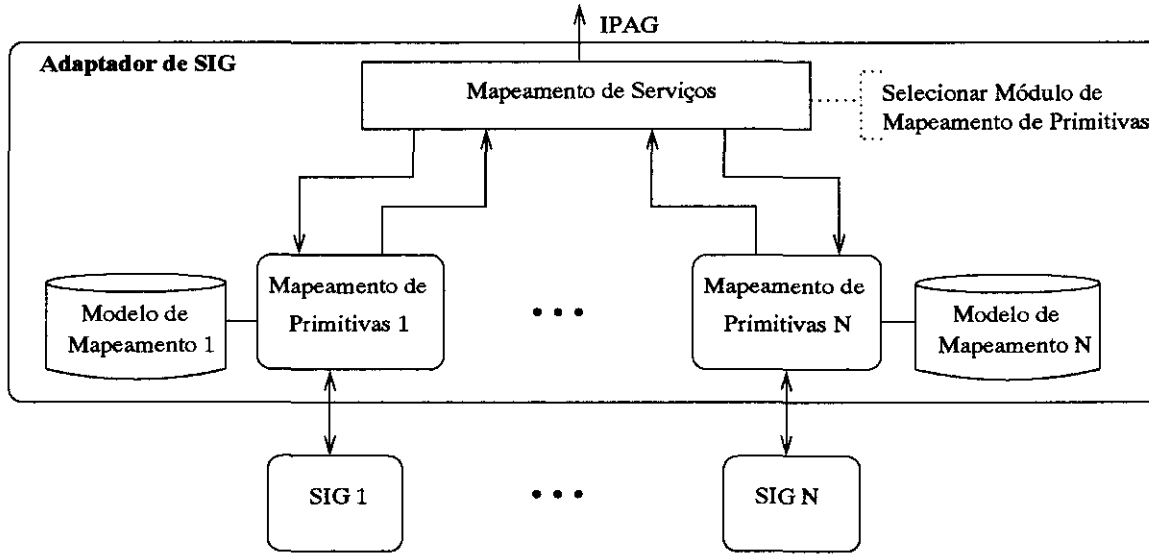

Figura 4.6: Um Módulo de Mapeamento de Primitivas para cada SIG

Na utilização isolada, a aplicação fica livre para escolher o SIG mais adequado para seus propósitos. De fato, a contribuição mais importante da Camada de Ligação para o restante da arquitetura de interface é a *independência de dados* com relação ao SIG. Esta característica garante que as aplicações não sejam afetadas por modificações efetuadas tanto na forma de armazenamento (independência física de dados) quanto no nível conceitual (independência lógica de dados) do SIG. As modificações ocorridas no sistema subjacente são propagadas apenas até o Módulo de Mapeamento de Primitivas (envolvendo também o modelo de mapeamento SIG↔GMOD). Tanto o Componente Interativo quanto o Núcleo Semântico tornam-se, assim, totalmente independentes do SIG utilizado.

É óbvio que os controles adicionais para seleção do SIG utilizado aumentam a complexidade de desenvolvimento da Camada de Ligação. No entanto, o custo de implementação pode ser diluído entre as várias aplicações que podem reutilizar esta camada. Além disto, a separação entre interface e SIG facilita a manutenção e a evolução de ambos os sistemas, e permite que o usuário utilize diferentes SIG de maneira transparente. Esta é uma solução para um problema comum nas interfaces atuais: o usuário que trabalha com diferentes SIG precisa aprender as particularidades de cada sistema utilizado.

Mapeamento entre Modelos de Dados

O mapeamento entre modelos que deve ser feito pelo Módulo de Mapeamento de Primitivas envolve um problema bastante estudado em sistemas de bancos de dados. Trata-se

da dificuldade de prover transparência de modelos de dados entre SGBD heterogêneos. O objetivo destes sistemas é permitir que um usuário/aplicação opere de maneira uniforme sobre diferentes modelos de dados.

A solução geralmente adotada para este problema introduz processadores que mapeiam um esquema E_1, definido de acordo com o modelo de dados M_1 de um SGBD S_1, para um esquema E_2, definido de acordo com o modelo de dados M_2 de um SGBD S_2 [SL90, Oli93, AM95]. O processo de mapeamento é feito em duas etapas. Na etapa *estrutural* as estruturas conceituais do modelo de dados M_1 existentes no esquema E_1 são traduzidas para estruturas equivalentes no modelo de dados M_2 que define o esquema E_2. Na etapa *operacional* o mapeamento estrutural é utilizado para permitir que operações expressas na linguagem de manipulação do modelo M_1 sejam transformadas em operações na linguagem definida pelo modelo M_2.

O Adaptador de SIG da Camada de Ligação deve prover exatamente este tipo de mapeamento. A automatização do processo exige a disponibilidade de metadados que definam precisamente o esquema do BD do SIG. Se o BD for projetado sem a preocupação em prover estes metadados, o mapeamento automático de esquemas e comandos torna-se praticamente impossível de se realizar.

Considere, por exemplo, um SIG que utiliza um SGBD relacional estendido com campos longos. No modelo relacional um único construtor – a relação – é usado para modelar tanto entidades quanto relacionamentos entre elas. Além disto, o modelo possui poucas restrições de integridade estruturais intrínsecas. Estas limitações semânticas tornam inviável o mapeamento para o modelo GMOD sem a intervenção de um especialista humano (o administrador de bancos de dados do SIG) que irá prover as estruturas que permitem tal mapeamento.

O ambiente UAPÉ, proposto em [OPM97], pode ser utilizado para modelagem de BD geográficos independente do SIG utilizado, contribuindo para a solução deste problema. O ambiente incorpora, de maneira automática, os metadados necessários para o mapeamento de/para o GMOD na definição do esquema do BD geográfico enviado para o SIG adotado.

4.7 A Camada de Modelos de Dados

Assim como a Camada de Ligação, a Camada de Modelos de Dados oferece serviços de suporte à Camada de Aplicação. Porém, enquanto os serviços da Camada de Ligação envolvem a comunicação com sistemas de suporte externos, os serviços da Camada de Modelos de Dados envolvem (1) a comunicação entre componentes da aplicação geográfica, e (2) a comunicação entre a aplicação e o usuário (através da interface).

Dois tipos de modelos de dados são utilizados para realizar estas comunicações: o GMOD, que representa os dados compartilhados entre o Núcleo Semântico e o Componente

Interativo; e o IMOD, que contém os objetos de interface com o usuário. O primeiro modelo serve de base para o Adaptador Interface–Semântico, discutido a seguir. O segundo modelo é utilizado pelo Construtor de Objetos de Interface, que será analisado no próximo capítulo. Tanto BD GMOD quanto BD IMOD são gerenciados pelo SGBD subjacente.

4.7.1 O Adaptador Interface–Semântico

O Módulo Adaptador Interface–Semântico oferece à camada superior da arquitetura um serviço de bancos de dados compatível com o modelo de dados intermediário da interface (GMOD). Para compor o esquema de um BD, o Adaptador utiliza informações fornecidas pelo Componente Interativo e pelo Núcleo Semântico, já que o propósito do BD é descrever os dados comuns aos dois componentes da aplicação geográfica.

Vale notar que ambos os componentes da aplicação podem utilizar os serviços da Camada de Ligação para recuperar dados e metadados no formato GMOD, e para executar funções sobre estes dados. Apenas o subconjunto destes dados que cruza a fronteira entre os componentes precisa ser definido no Adaptador. A construção de um BD comum deve ser baseada em um contrato em que os dois componentes acordam sobre (1) o fluxo de informações entre os componentes, incluindo a especificação de tipos, o nível de abstração de dados, e o componente responsável pelos dados; e (2) o conjunto de funções que cada componente pode executar sobre os diferentes tipos de dados.

Com isto, o Adaptador pode implementar uma API entre os dois componentes da aplicação sem comprometer a independência de diálogo. Mais ainda, as informações do modelo intermediário simplificam a execução de dois tipos de tarefas em um sistema interativo: a construção de apresentações e a validação semântica de dados e operações.

A construção de apresentações pode ser feita a partir da descrição do banco de dados. O sistema de interface pode utilizar o esquema e a extensão do BD GMOD para gerar, de maneira automática, apresentações dos dados manipulados pela aplicação. Este é o princípio fundamental das ferramentas de interface que seguem a abordagem de *construção de interface baseada em modelos* [SLN93, Pue97].

A interface pode utilizar a informação semântica contida no modelo intermediário para realizar certos tipos de verificações semânticas, como por exemplo as que se referem a restrições estruturais do BD GMOD (cardinalidade de relacionamentos, tipos de parâmetros, intervalos de valores, entre outras). A validação semântica feita na interface alivia a carga do Núcleo Semântico e diminui o volume de comunicação necessário entre os componentes.

4.7.2 Bancos de Dados GMOD e IMOD

Os BD da Camada de Modelos de Dados são gerenciados pelo SGBD do SIG subjacente. Cada aplicação define uma visão conceitual do espaço geográfico, à qual corresponde um

esquema de BD GMOD.

Tanto o Componente Interativo quanto o Núcleo Semântico fazem acesso ao BD exclusivamente através do Adaptador Interface–Semântico. Este módulo manipula o BD que descreve as entidades geográficas utilizadas na aplicação, e mantém a correspondência entre uma entidade conceitual e suas diferentes representações. Outra tarefa importante do Adaptador é converter operações conceituais em operações sobre representações específicas, as quais são enviadas para o SIG através da IPAG.

O BD IMOD e o Construtor de Interface são descritos no próximo capítulo. O BD GMOD é o único ponto de contato entre os componentes interativo e semântico da aplicação geográfica. O esquema deste BD descreve (1) os dados efetivamente utilizados na interface; e (2) as funções de aplicação e de interface que podem ser aplicadas sobre estes dados. A interação entre os dois bancos de dados também é descrita no próximo capítulo.

4.8 A Camada de Aplicação

A Camada de Aplicação é composta pelo Núcleo Semântico e pelo Componente Interativo. Apenas este último componente é especificado pela arquitetura, sendo responsável pelo gerenciamento do diálogo com o usuário, que consiste em duas tarefas principais: a criação e controle de apresentações de informações; e a interpretação de eventos de interface (ações do usuário). O Componente Interativo é detalhado no capítulo 5.

A definição de apresentações envolve operações gráficas e construção de *widgets* em tempo de execução. A interpretação de ações do usuário requer um mecanismo de monitorização de eventos de interface. Ambas as tarefas estão diretamente relacionadas aos objetivos específicos da aplicação geográfica. Por exemplo, a definição de visualizações para objetos geográficos depende do modelo mental do usuário final da aplicação.

Apesar de ser um componente definido especificamente para cada aplicação geográfica, o custo de uma interface desenvolvida de acordo com a arquitetura aqui apresentada pode ser reduzido através de técnicas de reutilização de *software* e de adaptação ao usuário. Os capítulos 5 e 6 discutem as propostas desta tese para facilitar o emprego destas técnicas no desenvolvimento do Componente Interativo da Camada de Aplicação.

4.9 Resumo

Este capítulo introduziu uma arquitetura de *software* de interface que define uma estrutura básica para projeto e desenvolvimento de interfaces geográficas. De acordo com a taxonomia de ferramentas apresentada na seção 2.3, a arquitetura pode ser classificada como uma variação da abordagem de *estrutura de aplicação predefinida* para construção

de interfaces geográficas. Foram apresentados os fundamentos da arquitetura em termos de componentes, mecanismos de comunicação inter-módulos, e diretivas de utilização.

O principal objetivo da arquitetura proposta é definir e organizar o Componente Interativo de um *sistema de aplicações geográficas* (SAG), viabilizando a reutilização de componentes entre suas aplicações. A idéia é diluir o custo de desenvolvimento através de módulos de interface que são compartilhados pelas diversas aplicações geográficas de um SAG.

A arquitetura está organizada em três camadas sobrepostas; cada camada é responsável pelo gerenciamento de um conjunto de tarefas, oferecendo serviços para as camadas superiores. As principais tarefas desempenhadas pelas camadas da arquitetura incluem:

- a comunicação com o SIG subjacente (camada inferior);
- o mapeamento entre modelos de dados (camada intermediária); e
- a interação com o usuário (camada superior).

Esta estruturação em camadas promove a modularização e a especialização dos componentes da interface, simplificando o desenvolvimento, e estabelecendo uma divisão clara de tarefas e responsabilidades. Além disso, esta estrutura permite definir diretivas gerais de projeto, possibilitando a reutilização das camadas mais baixas da arquitetura e garantindo independência de dados para a aplicação geográfica.

É preciso observar que aplicações *ad hoc* (que não se enquadram em um SAG) podem não obter estes benefícios, pela ausência de outras aplicações que possam compartilhar os mesmos componentes de interface. Neste caso, o custo de desenvolvimento de certos componentes reutilizáveis da arquitetura não se justifica. No entanto, a estrutura geral das camadas da arquitetura pode ser útil mesmo para o desenvolvimento da interface de uma aplicação isolada. A estrutura evita a abordagem usual em que todos os componentes da interface são embutidos em um único módulo de *software*. Este módulo é, em geral, extremamente complexo, pois precisa tratar de todas as tarefas mencionadas acima.

Apesar do ganho advindo da reutilização através de Adaptadores e da simplificação de cada componente, o custo de desenvolvimento ainda é alto, devido ao esforço necessário para a construção da camada superior da arquitetura (mais especificamente, do Componente Interativo da aplicação geográfica). Embora esta camada seja inerentemente dependente da aplicação e do modelo mental do usuário, existe a possibilidade de reduzir o seu custo de desenvolvimento. Os próximos capítulos apresentam soluções neste sentido, em termos de mecanismos para reutilização e personalização do Componente Interativo.

Capítulo 5

Modelo para Construção do Componente Interativo

5.1 Apresentação

O Componente Interativo da aplicação geográfica gerencia a interação entre o usuário e os demais módulos da arquitetura, com o objetivo de oferecer acesso às funções semânticas e de interface da aplicação geográfica. Apesar de sua importância, este componente não tem sido adequadamente explorado nos SAG atuais porque, em geral, o custo de sua construção cresce exponencialmente em relação à sua capacidade de ajuda ao usuário.

Este capítulo propõe técnicas e ferramentas de engenharia de software para o desenvolvimento do Componente Interativo, visando garantir: (1) que este componente possa cumprir seu objetivo de facilitar o trabalho do usuário; e (2) que ele possa ser desenvolvido dentro de patamares de custo aceitáveis. A idéia básica da abordagem é automatizar o processo de geração de interfaces a partir de modelos abstratos reutilizáveis.

A próxima seção apresenta uma visão geral do processo de construção do Componente Interativo. A seção 5.3 introduz o IMOD, um modelo de objetos de interface genérico proposto como base para o projeto e implementação deste componente. A seção 5.4 mostra as classes do IMOD voltadas para as necessidades específicas de aplicações geográficas. A seção 5.5 apresenta um módulo que constrói a interface a partir dos objetos de interface especificados segundo o IMOD. A seção 5.6 discute a utilização desta proposta em um ambiente de desenvolvimento de aplicações geográficas. A seção 5.7 resume o capítulo.

5.2 Construção do Componente Interativo

O usuário final de uma aplicação interage exclusivamente com o Componente Interativo, e não tem acesso direto ao Núcleo Semântico da aplicação. Cada Componente Interativo

de uma aplicação realiza o diálogo com o usuário através de uma *linguagem de interação* que define (1) o conjunto de funções que pode ser acionado pelo usuário; (2) as ações do usuário necessárias para ativar cada função; e (3) as formas de apresentação dos resultados destas funções.

Grande parte das aplicações desenvolvidas sobre SIG se caracteriza pela utilização intensiva e interativa do software pelos seus usuários. Neste tipo de aplicação, o total de linhas de código destinado ao diálogo com o usuário atinge, com freqüência, mais de cinqüenta por cento do código total [MR92, Mye95]. Este capítulo propõe uma abordagem para redução do custo de desenvolvimento do Componente Interativo em aplicações geográficas baseadas na arquitetura descrita no capítulo 4. Os principais objetivos da abordagem proposta são:

- Prover a capacidade de desenvolvimento de *interfaces dinâmicas*, isto é, Componentes Interativos cuja definição de apresentação e comportamento pode ser modificada em tempo de execução;
- Reduzir custos através da automatização do processo de construção do Componente Interativo, com a possibilidade de reutilização e extensão de elementos de interface já desenvolvidos;
- Garantir a separação e independência entre os componentes semântico e interativo da aplicação geográfica;
- Oferecer facilidades para o projeto da interface, mantendo um mapeamento simples para a fase de implementação.

O restante desta seção introduz os mecanismos e modelos utilizados para atingir estes objetivos. Inicialmente apresenta-se a estrutura interna do Componente Interativo, e em seguida discute-se o processo de construção deste componente.

5.2.1 Estrutura Interna do Componente Interativo

A arquitetura apresentada no capítulo anterior segue a abordagem modular, definindo componentes funcionais e mecanismos de comunicação entre eles. Do ponto de vista lógico, a estrutura interna do Componente Interativo não foge a este paradigma, sendo composta pelos módulos de Controle de Diálogo e de Apresentação, como mostra a parte superior da figura 5.1. No entanto, a partir da fase de projeto, a estrutura interna do Componente Interativo é modificada para atender os objetivos apresentados acima. Nesta fase, os módulos de Controle de Diálogo e de Controle de Apresentação dão lugar a *Objetos de Interface* (compostos por *Objetos de Interação*), que são o meio de comunicação entre usuário e Núcleo Semântico.

De acordo com a discussão apresentada na seção 2.3, um objeto de interação corresponde a um *widget*; já um objeto de interface encapsula não apenas estrutura e comportamento padrão de uma classe de *widgets*, mas também conhecimento do contexto de interface no qual ele se encontra. Um conjunto de objetos de interface implementa a mesma funcionalidade lógica geralmente prevista para os módulos de Controle de Diálogo e de Controle de Apresentação (figura 5.1, parte inferior).

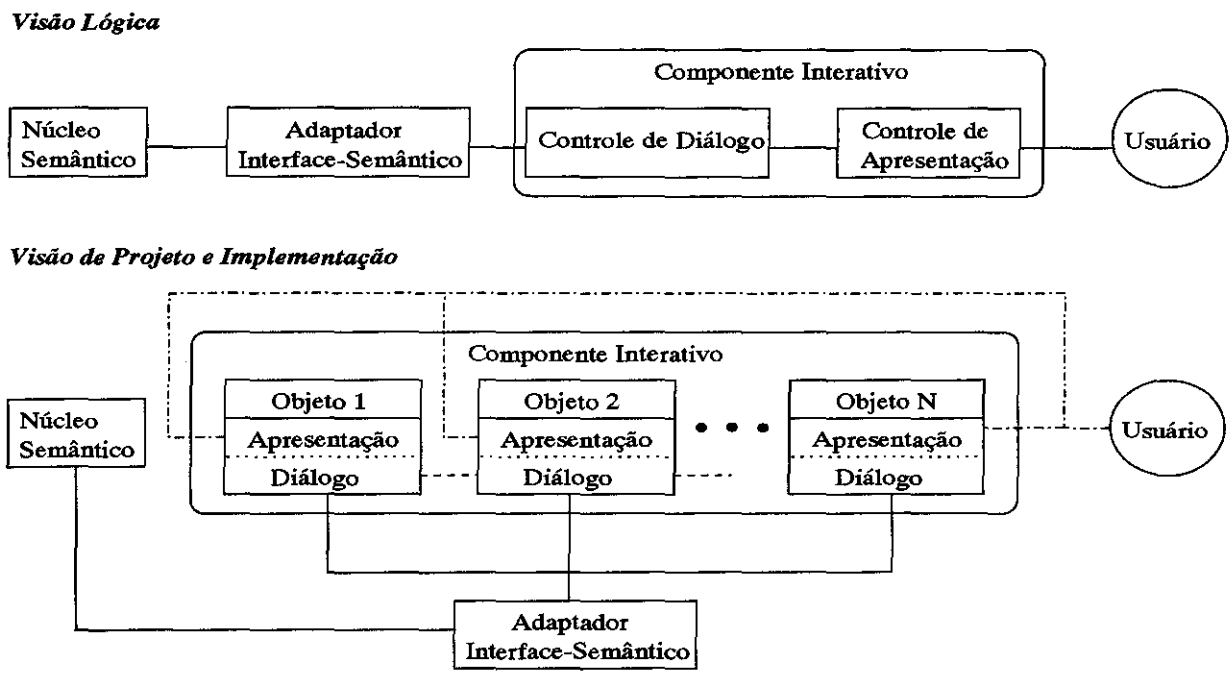

Figura 5.1: Perspectivas de construção do Componente Interativo

Este tipo de divisão da estrutura interna do Componente Interativo é próxima à filosofia proposta nas arquiteturas baseadas em agentes. Assim, a arquitetura apresentada nesta tese combina aspectos das arquiteturas modulares (como Seeheim e Slinky) e de agentes (como PAC e MVC). Uma abordagem similar aparece na arquitetura Pac-Amodeus (e suas variantes) [CCN97], que propõe uma combinação das arquiteturas PAC e Slinky.

Perspectiva de Projeto e Implementação

O principal fundamento utilizado no processo de construção do Componente Interativo é o modelo de objetos de interface IMOD, um modelo orientado a objetos voltado à construção de objetos de interface (seção 5.3). Este modelo de dados permite definir objetos de interface de maneira abstrata. Estas definições abstratas (chamadas de *modelos de objetos de interface*) são utilizadas pelo módulo Construtor de Objetos de Interface para definir os objetos de interação. O conceito de modelo de objetos de interface pode ser visto como

equivalente ao conceito de *template*, ou seja, uma estrutura que dirige a construção de cada objeto da interface. Em termos de bancos de dados, cada modelo de objetos é o esquema de uma classe.

O IMOD utiliza o conceito de *objeto* (seção 4.2) para realizar as tarefas de um *agente*. Deve-se ressaltar que existe uma correspondência direta entre os conceitos de *objeto* e de *agente*: ambos são unidades de processamento independentes que armazenam um estado e implementam um comportamento. *Grosso modo*, cada classe do modelo de objetos de interface IMOD define um agente na interface. Existem, no entanto, diferenças fundamentais entre este modelo e os modelos de agentes, como mostra a seção 5.3.

No paradigma de orientação a objetos, as instâncias de uma classe (objetos) são caracterizadas por seu estado e comportamento. No IMOD, as classes contêm objetos de interface que definem a apresentação (estado) e o controle (comportamento) de um aspecto da interface.

O estado de um objeto de interface abrange dois tipos de atributos: atributos de apresentação; e atributos de contexto. Os valores dos atributos de apresentação determinam, por exemplo, o aspecto gráfico do objeto de interface (cor, bordas, tipo de caractere, entre outros fatores) e a sua posição com relação ao leiaute definido pelo objeto que contém este objeto como componente. Os atributos de contexto definem o "modo" de interação; seus valores são utilizados pelos métodos que definem o comportamento do objeto para determinar as ações a serem tomadas em cada contexto de interface.

O comportamento de um objeto de interface define a sua reação a cada tipo de evento. Diversos modelos podem ser utilizados para especificar este tipo de comportamento, com destaque para aqueles baseados em Statecharts (diagramas de estado) [Har87]. Para esta notação, começam a surgir ferramentas capazes de sintetizar código ([HG97]) ou de executar diretamente as especificações abstratas ([LBL96]).

Vale lembrar que as arquiteturas baseadas em agentes (PAC, por exemplo) definem agentes como compostos pela tríade < abstração, apresentação, controle > (conforme discute o capítulo 2). No IMOD a apresentação e o controle são embutidos no objeto de interface, como mostra a figura 5.1, enquanto a abstração é provida pelo modelo intermediário da interface (GMOD).

5.2.2 Visão Geral do Processo de Construção

A construção de uma interface de acordo com a estrutura apresentada nesta tese envolve os módulos da arquitetura de interface em duas fases:

1. Fase de projeto: consiste basicamente em especificar os modelos de objetos de interface para os diversos elementos de diálogo da interface. Estes modelos são armazenados no BD IMOD, que pode ser considerado uma *Biblioteca de Objetos de*

Interface.

2. Fase de Implementação: abrange a instanciação de objetos destes modelos e a ligação das instâncias com o banco de dados GMOD.

Um modelo de objetos de interface define a apresentação e o comportamento de cada elemento interativo da interface. A especificação destes modelos é auxiliada pelo Construtor de Objetos de Interface, que gera, automaticamente, modelos de objetos de interface para as classes definidas no BD GMOD (modelo intermediário da interface). O esquema do BD GMOD é obtido junto ao Adaptador Interface–Semântico.

As especificações geradas são armazenadas na Biblioteca de Objetos de Interface (BD IMOD). Esta biblioteca é um repositório de diferentes modelos de objetos de interface que podem ser usados para construir uma interface geográfica. Os objetos da biblioteca são definidos segundo o IMOD, e podem ser reutilizados para diferentes aplicações, e modificados para aplicações específicas.

O projetista da interface pode posteriormente refinar os modelos de objetos gerados pelo Construtor de Objetos de Inteface e armazenar estes modelos refinados no BD IMOD. A fase de projeto da interface termina quando o projetista obtém um conjunto de classes no BD imod que atende aos requisitos da aplicação geográfica.

A fase de implementação da interface geográfica consiste em definir um programa para criar e gerenciar instâncias de objetos de interface. As instâncias são definidas sobre as classes (modelos) do BD IMOD e o Construtor é responsável por criar, junto ao Adaptador de *Toolkit*, os *widgets* envolvidos. O programa que implementa a interface deve relacionar as instâncias de objetos de interface às respectivas instâncias de geo-classes do BD GMOD.

Vale lembrar que tanto o BD GMOD quanto o BD IMOD são gerenciados pelo SGBD geográfico, como mencionado no capítulo anterior; já o Construtor de Objetos de Interface e o Adaptador Interface–Semântico são aplicações construídas sobre este SGBD.

5.3 O Modelo Dinâmico de Objetos de Interface

O IMOD – *Interface Data Model* – é o modelo proposto nesta tese para descrever objetos de interface. Este modelo foi desenvolvido com base no modelo de objetos de interface apresentado em [OCM95]. O IMOD utiliza o paradigma de orientação a objetos para descrever objetos de interface arbitrariamente complexos.

O emprego deste paradigma permite uma transição suave entre as fases de projeto e de implementação do Componente Interativo, e reforça as chances de reutilização de uma arquitetura de software [CWC94]. É preciso lembrar, todavia, que o modelo se restringe ao domínio específico de interfaces desenvolvidas de acordo com a arquitetura proposta

nesta tese. Esta restrição de domínio é indicada na literatura como um dos pontos chave para o sucesso de tecnologias de reutilização [BD94, Sta94].

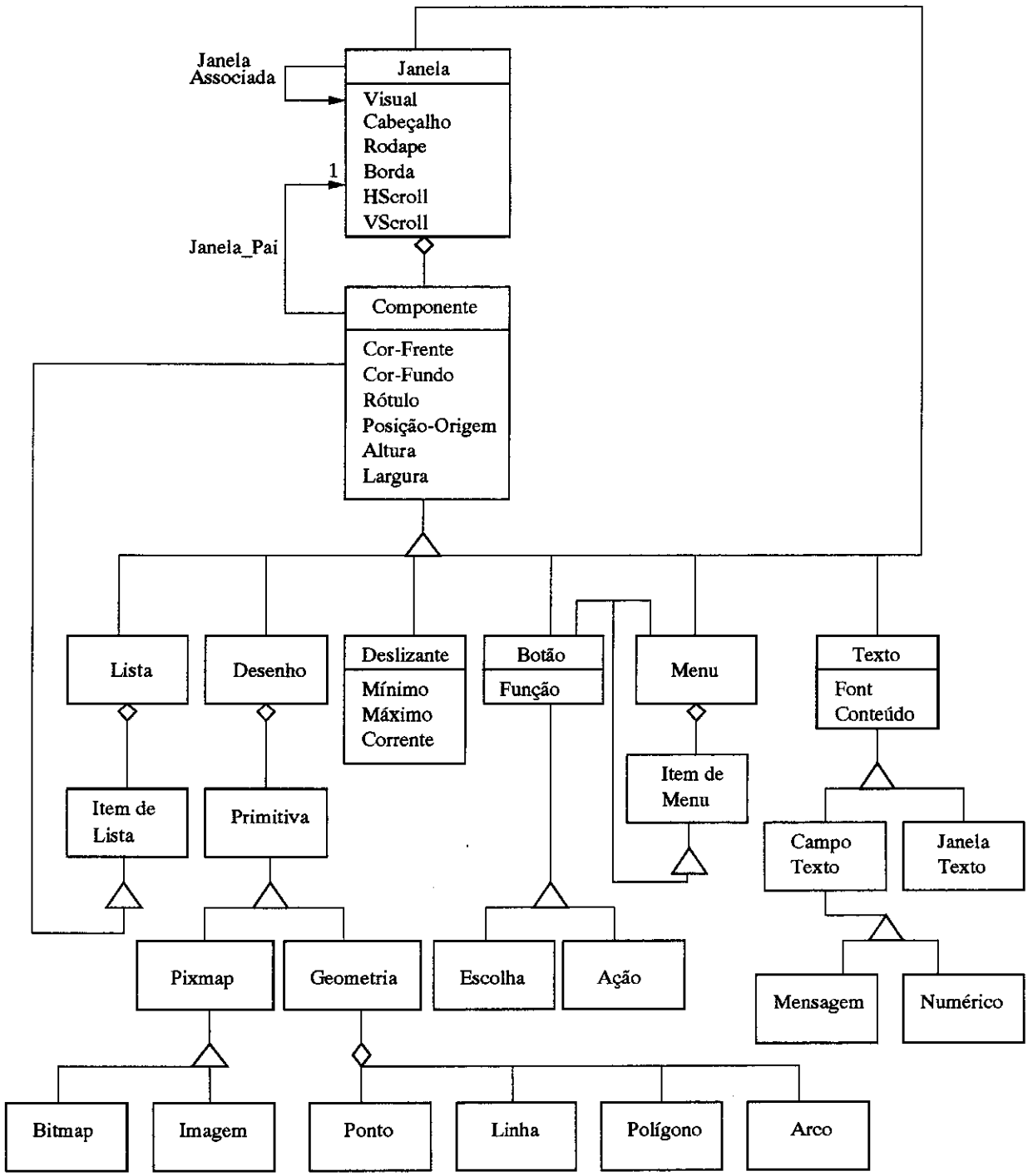

Figura 5.2: Núcleo do modelo de objetos de interface IMOD

A figura 5.2 mostra as classes e relacionamentos que formam o núcleo do IMOD, uti-

lizando a notação gráfica para o modelo orientado a objetos proposta pela metodologia OMT [RBP+91]: classes são representadas por caixas retangulares; associações são expressas por linhas ligando as classes; agregações são denotadas por um losango colocado sobre uma linha de associação; especializações são representadas por um triângulo colocado sobre a linha de associação com a classe genérica. Escolheu-se como base o conjunto de classes comuns a todos os *toolkits* de interface, junto com primitivas para construção de mapas. Estas classes básicas são agrupadas através de um único elemento de composição (a classe Janela), sendo o agrupamento direcionado a interfaces geográficas.

O objetivo deste modelo é facilitar o projeto de *interfaces visuais*, isto é, aquelas que suportam e permitem a coexistência de diversos estilos básicos de interação (linguagens, menus, telas formatadas, e manipulação direta). O IMOD descreve objetos de interface através de um conjunto de classes inter-relacionadas, onde cada classe descreve, em alto nível, um componente típico de um sistema de interface visual. Uma interface definida segundo este *modelo de objetos de interface* é formada por um conjunto de janelas que agregam objetos de interface e que se comunicam com uma aplicação específica.

A definição recursiva de componentes adiciona uma grande capacidade de reutilização ao modelo: a sua característica fundamental, no entanto, é a *extensibilidade*. É possível estender o modelo de duas formas:

- Adição de novas classes, que representa a incorporação de novos tipos de elementos de interface ao modelo. A seção 5.4 mostra a incorporação de classes específicas para tratamento de mapas ao núcleo do IMOD.

- Especialização e generalização de classes existentes, possibilitando redefinir as propriedades de elementos de interface, adaptando-os às particularidades de cada tipo de aplicação.

É importante ressaltar que este modelo é efetivamente orientado a objetos, de modo que cada classe define não apenas a estrutura das instâncias, mas também o seu comportamento. Outro ponto de destaque é a semelhança entre classes do IMOD e classes normalmente oferecidas por *toolkits* de interface. Esta semelhança se restringe à nomenclatura das classes, já que o modelo IMOD está situado em um nível de abstração maior que os dos modelos de *toolkits*.

5.3.1 A Classe Janela

O elemento básico de composição de interfaces no IMOD é a classe Janela. Qualquer interface visual baseia sua interação com o usuário em algum tipo de janela, que pode ser ou não gráfica, e que contém os elementos de diálogo usados na interação. Estes elementos são agrupados, recursivamente, em janelas subordinadas, geralmente de acordo

com a funcionalidade dos componentes. Em outras palavras, uma janela é formada por um conjunto de janelas subordinadas, onde cada janela agrega, por sua vez, elementos de interface relacionados de acordo com a concepção do projetista de interface.

Uma janela pode ser representada visualmente na interface, ou pode servir apenas para organizar seus componentes (atributo `visual`). Os principais atributos de uma janela visual são o formato das bordas (atributo `borda`), o padrão de apresentação de cabeçalho e rodapé (atributos `cabecalho` e `rodape`), e a indicação da presença de barras de rolamento (*scrollbars*) horizontal e vertical (atributos `hscroll` e `vscroll`, respectivamente). Adotando a notação do SGBD orientado a objetos O2 [BDK92], a classe Janela pode ser definida como:

```
Class Janela  inherits  Componente
  type tuple(janelas_associadas: list(Janela),
             componentes       : list(Componente),
             visual            : boolean,
             cabecalho         : string,
             rodape            : string,
             borda             : string,
             hscroll           : boolean,
             vscroll           : boolean)
  method
       abre_janela(),
       fecha_janela()
end;
```

O método `fecha_janela` é um exemplo de redefinição de comportamento de um objeto de interface. Em geral, este método é acionado por um evento do usuário ou da aplicação, e executa duas ações: torna a janela invisível para o usuário, e apresenta o ícone correspondente à janela. Este comportamento *default* é implementado pelo software de suporte (o *window manager*), mas pode ser redefinido pelo método `fecha_janela`. O método `abre_janela` tem o efeito oposto. Uma janela utiliza, ainda, propriedades (atributos e métodos) herdadas da superclasse `Componente`.

A estrutura de composição de uma janela de interface está refletida no modelo pela agregação que associa a classe Janela à classe abstrata Componente (representada pelo atributo `componentes`). A classe Componente é uma superclasse das diversas classes que representam elementos básicos de interface. Um destes elementos é a própria classe Janela, como será discutido a seguir.

Além de conter componentes de interface, uma janela pode estar associada a outras janelas. Este tipo de associação pode representar diferentes tipos de relacionamentos, como por exemplo a relação de hierarquia entre janela básica e janelas secundárias, em

um ambiente de janelas múltiplas, ou como a relação de ordem de chamada de janelas, em um sistema de telas formatadas. O *auto-relacionamento* definido pelo atributo `janelas_associadas` permite a modelagem deste conceito.

Um exemplo pode tornar mais clara a reutilização de componentes neste modelo: considere uma janela que permite escolher um mapa. Esta janela é um elemento de interface complexo, que pode conter listas para visualização e escolha, campos para entrada de identificadores de mapas, botões para seleção, filtros, entre outros elementos. Uma vez definido, este componente pode ser utilizado como parte de outras janelas. Dessa forma, novas janelas podem reutilizar diretamente o projeto e a implementação do componente de seleção de mapas.

5.3.2 Elementos Básicos de Interação

A classe abstrata Componente é a base para especificação dos diversos tipos de elementos de diálogo utilizados em interfaces visuais.

```
Class Componente
  type tuple(janela_pai     : Janela,
             widgets        : set(Widget), /* widgets do toolkit */
             posicao_origem : Ponto,
             altura         : integer,
             largura        : integer,
             rotulo         : string,
             cor_frente     : string,
             cor_fundo      : string)
  method
    constroi(),                          /* construtor de componente */
    destroi(),                           /* destrutor de componente */
    muda_local(nova_origem: Ponto),      /* move o componente */
    muda_forma(nova_origem: Ponto,       /* resize pode mudar origem */
               nova_forma: Retangulo)
end;
```

Um membro da classe Componente pertence sempre a uma única instância de Janela[1] indicada pelo atributo `janela_pai`. A classe Componente define ainda propriedades de apresentação que são herdadas pelos componentes especializados. As propriedades incluem as cores de *background* e *foreground* (`cor_frente` e `cor_fundo`), o valor do rótulo a ser apresentado junto ao componente (`rotulo`), a posição do componente com relação à ja-

[1] A seção 5.5.2 discute a única exceção a esta regra.

nela que o contém (`posicao_origem`), e as dimensões do componente (`altura` e `largura`).

Toda classe do modelo possui métodos específicos para construção e destruição de suas instâncias. Estes métodos, definidos de maneira abstrata em `constroi` e `destroi`, possuem a mesma semântica em todas as classes, embora tenham implementações diferentes.

Um componente é apresentado na sua `janela_pai`, ocupando um retângulo de dimensões `altura` e `largura`, cujo canto superior esquerdo fica na posição `posicao_origem` com relação ao retângulo da `janela_pai`. O método `muda_local` modifica a posição do canto superior esquerdo do retângulo do componente, enquanto `muda_forma` redefine o retângulo e a nova posição do componente. Estes métodos podem ser invocados em resposta a eventos do usuário (*move* e *resize* de uma janela, por exemplo) ou diretamente pela `janela_pai` (para gerenciar o leiaute de seus componentes).

As subclasses de Componente possuem conceitos correspondentes em praticamente todos os *toolkits* de interface atuais [NO90, Hel90, Ope91, Vis93, Pro96, MMM+97]: campos textuais, primitivas de desenho, controles deslizantes (*sliders*), botões, menus, e listas. De fato, estas subclasses utilizam as funções oferecidas pelos *widgets* correspondentes (atributo `widgets`). A discussão a seguir não trata de todas estas funções, mas se limita aos aspectos que são diretamente utilizados no modelo.

Campo Textual

Objetos da subclasse Texto possuem valores alfanuméricos (atributo `conteudo`) apresentados no formato de caracteres definido pelo atributo `font`. Instâncias de Texto podem ser usadas para entrada e saída de dados na interface.

```
Class Texto  inherits  Componente
  type tuple(font    : string,
             conteudo: string)
end;
```

O comportamento de um Texto é definido, basicamente, pelas funções oferecidas pelos *widgets* textuais. Existem diversas subclasses destes *widgets*, definindo comportamentos alternativos. Uma subclasse que aparece com freqüência é a classe Mensagem, que permite apenas a apresentação de informações alfanuméricas, mas não para entrada de dados.

Menu

Componentes da subclasse Menu permitem escolher entre várias opções disponíveis. A classe é composta por uma agregação de itens de menu, sendo que cada item pode ser um objeto da classe Botão, ou pode ser um outro menu (atributo `itens`). A escolha de um item de menu é feita através do método `ativa`, e um dos itens do menu é designado

como escolha *default* (atributo `default`). Assim como ocorre com a subclasse Texto, o comportamento da classe Menu utiliza funções disponíveis nos *widgets* de menu.

```
Class Menu  inherits  Componente
  type tuple(itens   : list(Componente),  /* Componente pode ser */
             default : Componente)        /*    Botao ou Menu */
  method
       ativa(Componente)  /* ativa o componente selecionado */
end;
```

Um menu cujos itens são também menus forma um "Menu em Cascata". Outro tipo de menu comumente utilizado é o menu de "Escolha Exclusiva", no qual cada item pertence à subclasse Botão de Escolha, e cuja ativação causa a desativação de todos os outros.

Controle Deslizante

Um controle deslizante (ou *slider*) é um componente analógico que permite a entrada e saída gráfica de um valor numérico situado dentro de um intervalo de valores. É comum o uso desta subclasse para fornecer *feedback* ao usuário (expressando, por exemplo, o percentual de uma tarefa que já foi realizada).

```
Class Deslizante  inherits  Componente
  type tuple(valor_corrente: integer,
             valor_maximo  : integer,
             valor_minimo  : integer)
  method
       reajusta_valores(corrente:integer, minimo:integer,
                        maximo:integer)
end;
```

O componente é tipicamente representado por uma barra que indica os valores mínimo e máximo (atributos `valor_minimo` e `valor_maximo`). Um elemento gráfico definido sobre a barra indica uma posição, que é convertida em um valor numérico (atributo `valor_corrente`), utilizando uma escala de conversão. O método `reajusta_valores` redefine os valores de atributos do controle deslizante.

Botão

Os objetos desta classe permitem, em geral, a ativação de uma função específica da aplicação (descrita pelo atributo `funcao` e ativada através do método `ativa_funcao`). Esta função pode ser restrita à camada de interface ou pode causar a chamada de funções do componente computacional (através de métodos definidos no BD GMOD).

```
Class Botao  inherits  Componente
  type tuple(funcao : Function)   /* ponteiro para a */
  method                          /*    funcao a ser ativada */
       ativa_funcao(funcao:Function)
end;
```

Um tipo de especialização muito utilizado define a subclasse Botão de Escolha, a qual estabelece um conceito de chaveamento (ligado ou desligado) para uma variável. Nesta subclasse a função ativada atualiza o valor de uma variável booleana da aplicação.

Lista

Em *toolkits* de interface, uma lista é tipicamente representada por uma área retangular dividida em linhas, onde cada linha define um elemento que pode ser representado por um campo textual, por um ícone ou por uma composição de texto e ícone. A definição da classe Lista do IMOD é mais abstrata.

```
Class Lista  inherits  Componente
  type tuple(itens     : list(Componente),
             separador: Desenho)
  method
       seleciona(itens:list(Componentes)),
       deseleciona(itens:list(Componentes))
end;
```

Uma instância de Lista representa elementos que podem ser selecionados ou deselecionados individual ou coletivamente (métodos `seleciona` e `deseleciona`). No entanto, a classe separa o *conceito* de lista de sua *apresentação*. Uma lista é formada por componentes de interface arbitrários (atributo `itens`), os quais possuem, como foi visto anteriormente, métodos que permitem o ajuste de seu leiaute. O projetista pode organizar os itens de uma instância de Lista de diferentes maneiras, com o auxílio do atributo `separador`, que define uma forma gráfica para separar os itens que compõem uma lista.

Esta definição abstrata de lista possui diversas subclasses, contemplando listas ordenadas, conjuntos matemáticos, vetores, tabelas multi-colunadas, e planilhas. Em particular, ela pode ser utilizada para representar construtores de atributos no modelo orientado a objetos, conforme discute a seção 5.4.1.

Desenho

Na classe Desenho estão as primitivas que permitem manipular valores gráficos nas janelas da interface. Estas primitivas correspondem às técnicas de desenho disponíveis na

biblioteca Xlib [Nye90], que são adotadas nos principais *toolkits* de interface disponíveis.

```
Class Desenho  inherits  Componente
  type tuple(primitivas_geometricas : set(Geometria),
             primitivas_pixmap      : set(Pixmap))
  method
       ponto_no_desenho(ponto:Ponto) : boolean,
       retangulo_corta_desenho(retangulo:Retangulo): boolean,
       desenhe()
end;
```

Uma instância de Desenho é definida por um conjunto de primitivas geométricas e de pixmaps (atributos `primitivas_geometricas` e `primitivas_pixmap`, respectivamente). A classe define um método que realiza o desenho definido pelas primitivas na sua janela_pai (método `desenhe`). Além disto, a classe especifica métodos para pesquisa espacial. Os métodos `ponto_no_desenho` e `retangulo_corta_desenho` identificam, respectivamente, se um determinado ponto pertence a, e se um determinado retângulo tem interseção com, alguma das primitivas da instância de Desenho.

Pixmap A classe Pixmap define primitivas para desenhos baseados em *mapas de pixel*, como por exemplo um bitmap, um ícone, ou uma imagem de satélite.

```
Class Pixmap  inherits  Componente
  type pixmap : array(pixel)
  method
       ponto_no_pixmap(ponto:Ponto) : boolean,
       retangulo_corta_pixmap(retangulo:Retangulo): boolean,
       desenhe()
end;
```

Neste tipo de primitiva, o próprio conteúdo, isto é, o conjunto de pixels (atributo `pixmap`) determina a aparência do desenho. O método `ponto_no_pixmap` determina se um determinado ponto pertence ao pixmap e o método `retangulo_corta_pixmap` identifica se um determinado retângulo tem interseção com o pixmap.

Geometria Primitivas geométricas permitem a criação de desenhos a partir de fórmulas matemáticas definidas em um sistema de coordenadas. As primitivas fundamentais são as que permitem desenhos de pontos, linhas, polígonos, e também as primitivas que permitem desenho de objetos com curvas (arcos).

Ao contrário do que ocorre com mapas de pixel, a descrição geométrica de uma entidade não define inteiramente sua apresentação. A *linguagem cartográfica* utiliza *variáveis*

visuais (tamanho, cor, valor, textura, e orientação, entre outras) para complementar a descrição da aparência (variável visual *forma*) de uma entidade geográfica [CCH+96].

```
type Propriedade_Visual
     tuple(cor       : string,
           textura   : string,
           tamanho   : integer,
           padrao    : string,  /* padrao de preenchimento */
           valor     : string,  /* valor textual do rotulo */
           orientacao: string)  /* orientacao do rotulo */
end;

type Apresentacao_Geometria
     tuple(pontos   : tuple(geometria:set(Ponto),
                            aspecto_grafico:Propriedade_Visual),
           linhas   : tuple(set(Linha),
                            aspecto_grafico:Propriedade_Visual),
           poligonos: tuple(set(Poligono),
                            aspecto_grafico:Propriedade_Visual),
           arcos    : tuple(set(Arco),
                            aspecto_grafico:Propriedade_Visual))
end;

Class Geometria  inherits  Componente
  type apresentacao_geometria : Apresentacao_Geometria
  method
       ponto_na_geometria(ponto:Ponto) : boolean,
       retangulo_corta_geometria(retangulo:Retangulo): boolean,
       desenhe()
end;
```

No IMOD, o tipo `Propriedade_Visual` define as variáveis visuais cartográficas que determinam a aparência de cada primitiva geométrica: ponto, linha, polígono e arco. Estas primitivas utilizam a implementação feita na biblioteca Xlib [Nye90], não sendo redefinidas aqui. O tipo `Apresentacao_Geometria` é uma combinação de primitivas geométricas básicas, associadas às variáveis visuais que definem sua apresentação.

Uma instância da classe Geometria define um valor para este último tipo, através do atributo `apresentacao_geometria`. O método `ponto_na_geometria` identifica se um ponto pertence à geometria da instância e o método `retangulo_corta_geometria` testa

se um retângulo tem interseção com esta geometria.

Componentes da classe Desenho são fundamentais para aplicações gráficas, tais como as definidas em SAG. Uma instância de Desenho pode combinar primitivas de geometria e de pixmap, oferecendo um mecanismo fundamental para desenhar instâncias de GeoClasse no contexto de um mapa. A seção 5.4.2 mostra a utilização deste componente para apresentação de mapas em SIG.

5.4 Uso do IMOD em Aplicações Geográficas

O núcleo do IMOD oferece classes básicas para especificação de interfaces visuais de propósito geral. Aplicações geográficas, no entanto, possuem idiossincrasias que não são facilmente tratadas pelas classes genéricas. Para atender a estas particularidades são propostas duas extensões ao núcleo do IMOD. A primeira cria componentes complexos que facilitam a gerência de dados da classe Convencional do GMOD (descrita na seção 4.4.1). A segunda extensão visa suportar a interação com dados geo-referenciados.

5.4.1 Atributos Convencionais

Apesar das dificuldades inerentes à construção de mapas, este não é o único problema em interfaces geográficas. Uma característica de aplicações geográficas freqüentemente desconsiderada em projetos de interface é a manipulação de dados convencionais. O componente espacial (ou geo-referenciado) da informação é de fato importante mas, em muitas aplicações geográficas, ele serve apenas de contexto para a localização e manipulação de dados convencionais.

Uma possível justificativa para a pouca importância dada aos atributos convencionais é que as interfaces de SGBD tradicionais já provêem mecanismos adequados para tratar este tipo de informação. Porém, esta afirmativa é verdadeira apenas no domínio do modelo de dados relacional; em SGBD orientados a objetos, as questões relacionadas à apresentação de e interação com classes e instâncias continuam em aberto. Como SIG modernos utilizam SGBD com modelos mais ricos em semântica (notadamente o modelo OO) é preciso definir ferramentas para projetar a interação com estes modelos. A taxonomia apresentada a seguir define as características de um modelo orientado a objetos que precisam ser consideradas na construção de apresentações de dados na interface.

Taxonomia de Objetos de Interface

O modelo GMOD permite a construção de atributos complexos através do uso de construtores de tipo do modelo de dados orientado a objetos. Uma taxonomia destes atributos com relação à sua forma de apresentação em interfaces visuais deve incluir:

1. Atributos *simples*: são aqueles apresentados através de um único objeto de interface. Podem ser subdivididos de acordo com o tipo deste objeto em:

   (a) Atributos *triviais*: o objeto de interface é um componente básico de interação do IMOD. Atributos atômicos do modelo orientado a objetos são, em geral, atributos triviais de interface: inteiros, reais, cadeias de caracteres, entre outros, podem ser adequadamente representados através de um botão ou de um campo textual elementar.

   (b) Textos: são apresentados através de janelas específicas que provêm facilidades padronizadas para visualização e edição de textos. Existe uma correspondência direta entre um atributo textual e alguma subclasse simples da classe Texto que o apresenta.

   (c) Imagens: assim como os textos, imagens são apresentadas em janelas dedicadas, e geralmente suportadas diretamente pelos *toolkits*. No entanto, textos possuem uma representação padronizada (ASCII), enquanto que imagens podem ser armazenadas em diferentes formatos (raster, bitmap, GIF, entre outros).

   (d) Atributos multimídia: o modelo orientado a objetos suporta a definição de novos tipos de dados, como por exemplo, sons e animações. Para representar estes tipos de dados é preciso definir objetos de interação *ad hoc*. Um atributo do tipo *som*, por exemplo, exige recursos adicionais de hardware e software normalmente não disponíveis em interfaces WIMP (baseadas em *Windows, Icons, Mice*, e *Pointing*).

   Apesar da falta de suporte em *toolkits* atuais, atributos multimídia podem ser considerados *simples* (de acordo com a presente taxonomia), porque são apresentados por um único componente de interface (uma janela de animação ou um elemento reprodutor de som) que não é composto por outros objetos.

2. Atributos *complexos*: envolvem mais de um objeto de interface para sua apresentação. Tomando-se o GMOD como base, pode haver três tipos de atributos complexos:

   (a) Atributos espaciais: são apresentados através de uma *Janela de Mapa*, um objeto de interface composto descrito na próxima seção.

   (b) Atributos retratados por construtores: podem ser homogêneos, quando permitem agrupar elementos de mesmo tipo (construtores *list* e *set*, por exemplo), ou heterogêneos, se permitem agrupar elementos de diferentes tipos (como o construtor *tuple*). De qualquer forma, um construtor é apresentado na interface por meio de um objeto de interface que representa o tipo do construtor, e que

é composto pelos objetos de interface que representam os diversos atributos agrupados pelo construtor.

A classe Lista do IMOD facilita a apresentação deste tipo de atributo complexo. Uma instância de Lista representa o construtor e os itens da lista representam os atributos agrupados pelo construtor. Itens de Lista são instâncias arbitrárias de Componente, de forma que é possível representar tanto construtores homogêneos quanto heterogêneos, sendo o separador da Lista usado para indicar o tipo de construtor representado. Mais ainda, um item de lista pode ser composto, permitindo representar eventuais atributos complexos agrupados por um construtor. Em particular, é possível definir construtores aninhados, o que é representado no IMOD por Listas aninhadas, isto é, instâncias de Lista cujos itens são outras listas.

(c) Sub-objetos: são atributos usados para representar o conceito de *objeto complexo*, segundo o qual um objeto pode ser formado por um conjunto arbitrário de outros objetos. Um sub-objeto é apresentado em uma janela composta de acordo com a definição de tipo de sua classe. A construção desta janela segue exatamente o mesmo modelo de construção que a janela do objeto composto que contém o sub-objeto.

O Construtor de Interface, descrito na seção 5.5, é uma ferramenta que constrói projetos de interface a partir das definições do GMOD, utilizando esta taxonomia para apresentação de atributos.

5.4.2 Atributos Espaciais

Praticamente todos os trabalhos em interfaces para SIG consideram alguma variação do conceito de *mapa* como elemento de comunicação entre o usuário e o SGBD espacial. A utilização deste conceito envolve não apenas uma elaborada produção cartográfica, mas também uma programação de interface complexa.

Mapas em Interfaces Geográficas

Uma interface geográfica deve apresentar dados e operações oferecidas pela aplicação através da combinação de diversos tipos de objetos de interface. Portanto, uma interface geográfica inclui elementos *cartográficos* (envolvendo dados geo-referenciados organizados, via de regra, em mapas) e também elementos de *controle* (envolvendo dados convencionais e operações da aplicação representados por objetos do IMOD).

Os relacionamentos entre os dois tipos de elementos dependem dos objetivos de cada aplicação. Esta dificuldade para definição de relacionamentos não é exclusiva de SIG.

ocorrendo, em geral, nos sistemas que manipulam objetos gráficos (CAD/CAM, por exemplo). A ausência de mecanismos genéricos para descrever os relacionamentos entre objetos gráficos e de controle se reflete nas ferramentas de desenvolvimento (*toolkits* de interface) que, em geral, não oferecem suporte para a implementação destes relacionamentos.

Um exemplo pode tornar mais claro o problema existente para caracterizar este tipo de relacionamento. Considere uma aplicação geográfica que apresenta cidades de uma região como pontos em um mapa. A semântica de uma operação de seleção de cidade pode variar de acordo com o objetivo da aplicação. A operação pode originar, por exemplo, uma consulta ao SIG que resulta na apresentação dos atributos convencionais da cidade selecionada em uma janela auxiliar (utilizando novos elementos de controle -isto é, objetos de interface - associados ao elemento cartográfico original). Alternativamente, a operação pode indicar que o usuário deseja visualizar detalhes gráficos (isto é, uma operação de *zoom*), envolvendo uma mudança de escala e conseqüentemente uma reorganização dos elementos cartográficos e dos elementos de controle associados. Os mecanismos discutidos a seguir facilitam a implementação destes relacionamentos.

Construção de Mapas no IMOD

Para oferecer facilidades de manipulação de mapas, três classes de objetos são acrescentadas ao núcleo do IMOD: a classe Instância Gráfica, que permite representar graficamente objetos da aplicação; a classe Camada Gráfica, que associa conjuntos de instâncias gráficas em um contexto espacial único; e a classe Janela de Mapa, que permite a apresentação e manipulação integrada de diferentes camadas gráficas (figura 5.3).

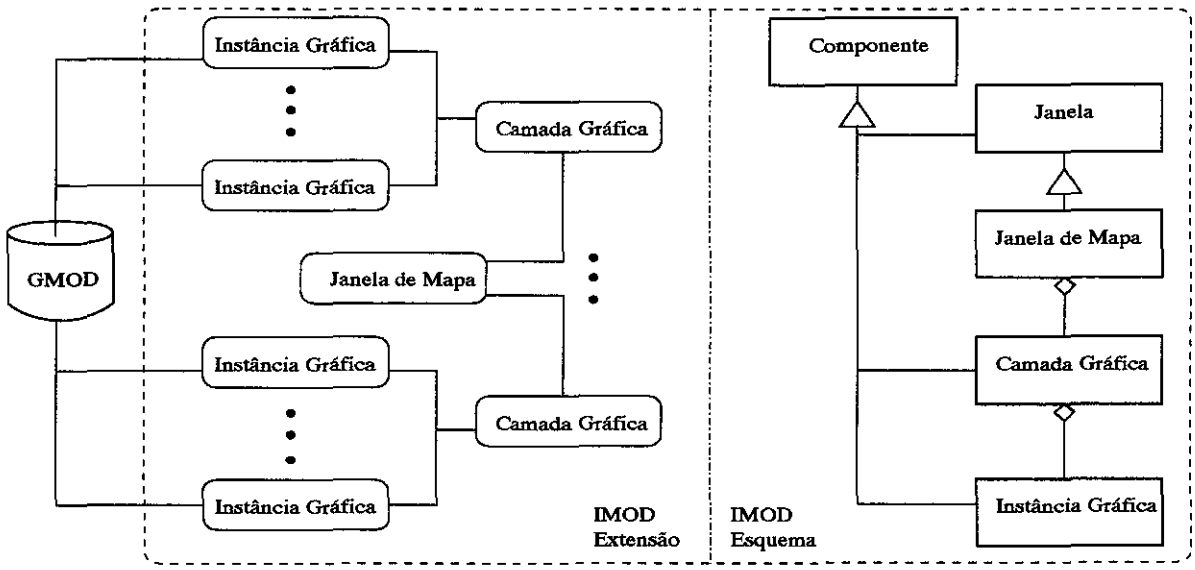

Figura 5.3: Manipulação de informações espaciais no IMOD

Esta organização de interface geográfica é semelhante àquela introduzida por Rigaux em [Rig95]. No entanto, a proposta de Rigaux visa construir uma interface específica (uma interface de consulta para SIG), e não pode ser diretamente utilizada para desenvolver uma interface para outro tipo de aplicação geográfica. Outra limitação de [Rig95] é a impossibilidade de utilização de geo-campos, pois a arquitetura está voltada exclusivamente para SIG vetoriais. Por fim, a arquitetura de interface de Rigaux não prevê classes para manipular os atributos convencionais da informação geográfica.

A discussão a seguir mostra as modificações e extensões efetuadas sobre a proposta de [Rig95] para atender requisitos de aplicações geográficas genéricas, utilizando o GMOD (o que implica em visões de campos e de objetos) e o IMOD (o que torna necessário um mecanismo de integração com as demais classes deste modelo).

Instância Gráfica Uma instância de Geo-Classe do GMOD se caracteriza por um conjunto de atributos covencionais e por um atributo que descreve suas propriedades espaciais (*atributo de localização*). Este último permite a implementação de funções de análise espacial no SIG. Para os objetivos da interface, todavia, o formato (topológico ou numérico) do atributo de localização não é adequado. A classe *Instância Gráfica* é incorporada ao IMOD para resolver esta dificuldade.

```
Class Instancia_Grafica  inherits  Componente
  type tuple(objeto_gmod : OID,
             selecionada :boolean,
             apresentacao: Desenho)
  method
       gera_apresentacao(objeto_gmod:OID,
                         selecionada:boolean,
                         escala:integer) : Desenho,
       seleciona_instancia(selecao:boolean),
       ponto_na_apresentacao(ponto:Ponto) : boolean,
       retangulo_corta_apresentacao(retangulo:Retangulo)
                                          : boolean,
       mostra_atributos_convencionais()
end;
```

Uma instância de Instância Gráfica é a representação, sob a perspectiva da interface, da descrição espacial de uma instância de Geo-Classe. Uma instância gráfica se refere a uma única instância de Geo-Classe (embora diferentes instâncias gráficas possam se referir à mesma instância de Geo-Classe). Esta correspondência biunívoca é importante para permitir o mapeamento das atualizações feitas sobre um objeto de interface para o

objeto correspondente no BD geográfico. Interfaces de consulta não enfrentam, em geral, este tipo de problema. O OID da instância de Geo-Classe representa este relacionamento (atributo `objeto_gmod`).

O atributo `apresentacao` é um *desenho* que representa a descrição espacial da instância de geo-classe, para propósitos de apresentação na interface. O atributo `selecionada` indica se a instância gráfica está ou não selecionada. Isto é necessário pois diversas funções de interface são executadas apenas sobre instâncias selecionadas na apresentação. O método `mostra_atributos_convencionais`, por exemplo, permite criar uma apresentação para os atributos convencionais do objeto do GMOD associado à instância gráfica selecionada. A criação deste tipo de apresentação é discutida na seção 5.4.1. O método `seleciona_instancia` reajusta o valor do atributo `selecionada`.

O método `gera_apresentacao` é uma função de conversão que transforma a geometria da instância de Geo-Classe (definida no atributo de localização da Geo-Classe) em uma variável visual (do tipo Desenho) adequada para visualização pelo usuário. A apresentação de uma instância varia de acordo com a escala, e também pode ser diferente para instâncias selecionadas e não-selecionadas. Este método pode ser utilizado, ainda, para efetuar transformações de generalização cartográfica, ou para selecionar e combinar diferentes representações da instância de Geo-Classe em uma apresentação.

É importante observar que o método `gera_apresentacao` mantém o sistema de coordenadas geográficas original do GMOD na criação do Desenho, pois a interface utiliza estas coordenadas para interação com o usuário. A implementação deste método envolve problemas relativos à produção cartográfica de mapas, ainda em aberto; conseqüentemente, o método é especificado de maneira *ad hoc*.

Os métodos `ponto_na_apresentacao` e `retangulo_corta_apresentacao` implementam buscas espaciais. Eles utilizam os métodos correspondentes na classe Desenho, que por sua vez utilizam os respectivos métodos das classes Pixmap e Geometria.

Camada Gráfica Uma instância da classe *Camada Gráfica* é um objeto de interface complexo formado por um conjunto de instâncias gráficas conceitualmente relacionadas.

A camada permite agrupar em uma mesma janela diversos objetos espaciais definidos sobre uma região geográfica (atributo `instancias`). Os objetos que pertencem a uma camada são determinados através de um critério de seleção como, por exemplo, uma consulta feita ao SIG. O método `obtem_geoinstancias` implementa tal processo de recuperação de objetos geográficos e criação de instâncias gráficas correspondentes.

A região geográfica da camada gráfica é determinada pelas coordenadas de um retângulo delimitador mínimo (*minimum bounding rectangle* – MBR) que envolve todas as instâncias gráficas contidas na camada. Este retângulo é representado pelo atributo `mbr`. O método `clip` permite redefinir a região geográfica e, conseqüentemente, o conjunto de

instâncias gráficas presente na camada.

```
Class Camada_Grafica  inherits  Componente
  type tuple(instancias: set(Instancia_Grafica),
             mbr       : Retangulo)
  method
       obtem_geoinstancias(funcao_seletora: Function)
                          : set(OID),
       clip(area:Retangulo) : set(Instancia_Grafica),
       seleciona_instancias(ponto:Ponto)
                          : set (Instancia_Grafica),
       seleciona_instancias(area:Retangulo)
                          : set (Instancia_Grafica)
end;
```

O método `seleciona_instancias` é polimórfico. Sua semântica consiste em selecionar instâncias gráficas da camada em função de um parâmetro geométrico. Se o parâmetro é um ponto, o método seleciona todas as instâncias gráficas que incluem o ponto dado em sua apresentação. Para isto o método invoca o método `ponto_na_apresentacao` de cada instância gráfica. Se o parâmetro de `seleciona_instancias` é um retângulo, as instâncias gráficas selecionadas são aquelas cujos métodos `retangulo_corta_apresentacao` retornam resultados verdadeiros.

Convém notar a diferença entre a *seleção* feita pelos métodos `obtem_geoinstancias` e `seleciona_instancias`. O primeiro busca objetos do GMOD satisfazendo um critério ou predicado, enquanto o segundo método cria uma marca de seleção nas instâncias gráficas para futuras operações.

Janela de Mapa A classe Janela de Mapa define a apresentação de um conjunto de camadas gráficas em uma janela visual, facilitando a implementação de operações típicas de interfaces geográficas (*zoom*, *pan*, e seleção de objetos gráficos, entre outras).

Uma instância de janela de mapa gerencia uma lista de camadas gráficas relacionadas espacialmente (atributo `camadas`). A apresentação das camadas é feita através de sobreposição gráfica (método `display`), considerando-se a existência de dois tipos de camadas: *opacas* e *transparentes* (atributo `opaca`). Uma camada opaca impede a visualização das camadas inferiores, enquanto uma camada *transparente* realiza uma combinação com os objetos gráficos das camadas inferiores.

As camadas são sobrepostas na ordem inversa à definida pelo construtor *list* do atributo `camadas`, ou seja, a primeira camada desta lista será a última camada a ser sobreposta. A primeira camada a ser apresentada, que será a camada de *background* da

apresentação, será a primeira camada opaca encontrada (ou a última camada da lista, caso não haja camadas opacas). O método `muda_posicao_camada` permite mudar a ordem das camadas na lista de `camadas`.

```
Class Janela_Mapa  inherits  Janela
  type tuple(camadas      : list(tuple(camada: Camada_Grafica,
                                       opaca : boolean)),
             mbr          : Retangulo,
             escala       : integer)
  method
       zoom(nova_area:Retangulo),
       zoom(nova_escala:integer),
       muda_posicao_camada(camada:Camada_Grafica,
                           nova_posicao:integer),
       seleciona_instancias(ponto:Ponto,
                           camadas:list(Camada_Grafica))
                           : list (Instancia_Grafica),
       seleciona_instancias(area:Retangulo,
                           camadas:list(Camada_Grafica))
                           : list (Instancia_Grafica),
       display()
end;
```

O atributo `mbr` define a área geográfica de interesse, permitindo selecionar uma região específica das diversas camadas que compõem o mapa (através do método `clip` de cada Camada_Grafica). É preciso observar que as camadas gráficas podem definir MBRs distintos. O MBR da janela de mapa deve estar contido na interseção dos MBRs das camadas gráficas visualizadas.

A escala de apresentação (atributo `escala`) pode ser calculada automaticamente, a partir do tamanho da área de desenho (ou seja, o tamanho da janela de mapa) e do `mbr` do mapa, de modo que todas as camadas gráficas caibam na janela de mapa:

`escala` = MIN(`largura/mbr.largura, altura/mbr.altura`)

Vale lembrar que os atributos `altura` e `largura` da Janela de Mapa são herdados indiretamente de Componente. O atributo `escala` também pode ser modificado, por exemplo, para implementar uma operação de *zoom*. Na janela de mapa o método `zoom` é polimórfico, suportando duas semânticas distintas: *tradicional* e *cartográfica*.

O *zoom* tradicional é normalmente encontrado em sistemas de janelas e ocorre quando o parâmetro utilizado é do tipo retângulo. A semântica desta operação modifica a região do espaço com a qual se trabalha, e eventualmente a escala. O método redefine o `mbr` de acordo com o parâmetro, calcula a `escala` de apresentação do novo `mbr` na janela de

mapa, e invoca o método `clip` de suas camadas gráficas com o novo `mbr` de interesse. Se a `escala` for mantida, o *zoom* espacial equivale a uma operação de deslocamento de área (*panning*); caso contrário, o método de *zoom* cartográfico deve ser invocado para apresentar a janela com a nova escala.

A semântica do *zoom* cartográfico mantém a região de interesse e varia a escala de trabalho. Esta semântica é executada se o parâmetro do método `zoom` é do tipo inteiro. Neste caso o método mantém a região de interesse (`mbr`), modifica a `escala` de acordo com o parâmetro, e invoca o método `gera_apresentacao` de cada instância gráfica visualizada, através das camadas gráficas que as contêm.

O método `seleciona_instancias` também é polimórfico. Ele permite selecionar instâncias gráficas de uma lista de camadas gráficas em função de um parâmetro geométrico. Se, por exemplo, o parâmetro é um ponto, o método seleciona todas as instâncias gráficas que incluem o ponto dado em sua apresentação. Se o parâmetro é um retângulo, as instâncias selecionadas são aquelas cujas apresentações possuem interseção com o retângulo dado. O método invoca o respectivo método `seleciona_instancias` de cada camada gráfica da lista para realizar a seleção.

Antes de apresentar as camadas gráficas ao usuário, o método `display` deve efetuar a translação das coordenadas da apresentação de cada instância gráfica para uma coordenada na janela de mapa. Para efetuar a translação o método determina uma equivalência entre o canto superior esquerdo do `mbr` e o canto superior esquerdo da janela de mapa. Desta forma, a janela de mapa pode gerenciar tanto coordenadas geográficas quanto coordenadas de interface. A capacidade de conversão entre os dois sistemas de coordenadas é essencial em interfaces geográficas, pois o usuário trabalha com coordenadas geográficas, mas interage com o sistema através de coordenadas de interface.

5.5 O Construtor de Objetos de Interface

O Adaptador Interface–Semântico implementa a comunicação entre os componentes semântico e interativo através do BD GMOD. O esquema deste BD define, para cada objeto da aplicação, todas as funções de interface que são visíveis para o núcleo semântico, e todas as funções de aplicação que podem ser chamadas pelo componente interativo.

A arquitetura de interface propõe, desta forma, um único ponto de contato entre os componentes interativo e semântico da aplicação geográfica, que é o próprio banco de dados BD GMOD compartilhado entre os componentes. O esquema deste BD descreve (1) os dados efetivamente utilizados na interface; e (2) as funções de aplicação e de interface que podem ser aplicadas sobre estes dados. O *Construtor de Objetos de Interface* é utilizado para gerar a estrutura básica de objetos de interface a partir deste BD. Além de automatizar uma parte considerável dos projetos, o Construtor assegura a padronização

entre os diversos projetos de interface de um SAG.

5.5.1 Estrutura de um Modelo de Objetos de Interface

Um modelo de objeto de interface definido no IMOD é um *template* gerado pelo Construtor de Objetos de Interface para cada classe do BD GMOD, especificando:

1. Uma janela IMOD para apresentação de cada atributo.

   A taxonomia apresentada na seção 5.4.1 é utilizada para gerar as janelas de apresentação de cada atributo, de acordo com o seu tipo. Assim, uma Janela de Atributo pode ser simples ou complexa.

2. Uma janela IMOD para apresentação da classe.

   A Janela de Classe é complexa, contendo como sub-objetos as diversas janelas de atributo. Esta janela é responsável pela coordenação das janelas componentes. Ela também mantém a ligação com a instância correspondente do BD GMOD através do OID deste objeto. Utilizando as informações do esquema GMOD e o OID da instância, a janela de classe pode recuperar do BD GMOD os valores dos atributos e repassá-los para as respectivas janelas componentes.

3. Os eventos de usuário que são processados pelo objeto de interface.

   Os eventos básicos definidos pelo Construtor correspondem aos eventos de servidor definidos na arquitetura Chiron-1 [TNB+95]: eventos de teclado (*key*); eventos de *click* de mouse (com o botão da esquerda, da direita, ou central, que correspondem, respectivamente, aos eventos de seleção, de menu, e de ajuste definidos pelo padrão de interface *OpenLook*); e eventos de *drag* de mouse (podem ser de *resize* ou de *move*, dependendo da parte do objeto de interface onde o evento se iniciou). Estes eventos são processados por métodos dos objetos de interface.

4. As funções de aplicação que podem ser chamadas pelo objeto de interface.

   O comportamento de uma classe do GMOD é definido tanto para funções de interface (exportadas pelo Componente Interativo) quanto para funções de aplicação (exportadas pelo Núcleo Semântico). Cada função corresponde a um método de uma classe do esquema GMOD, e cada tipo de função é identificada por um prefixo (*Appl* para funções de aplicação, e *Int* para funções de interface). Os métodos de prefixo *Appl* são definidos pelo Construtor como funções de aplicação disponíveis na janela. O Núcleo Semântico, em contrapartida, pode invocar as funções de interface, as quais devem ser implementadas como métodos da janela de classe.

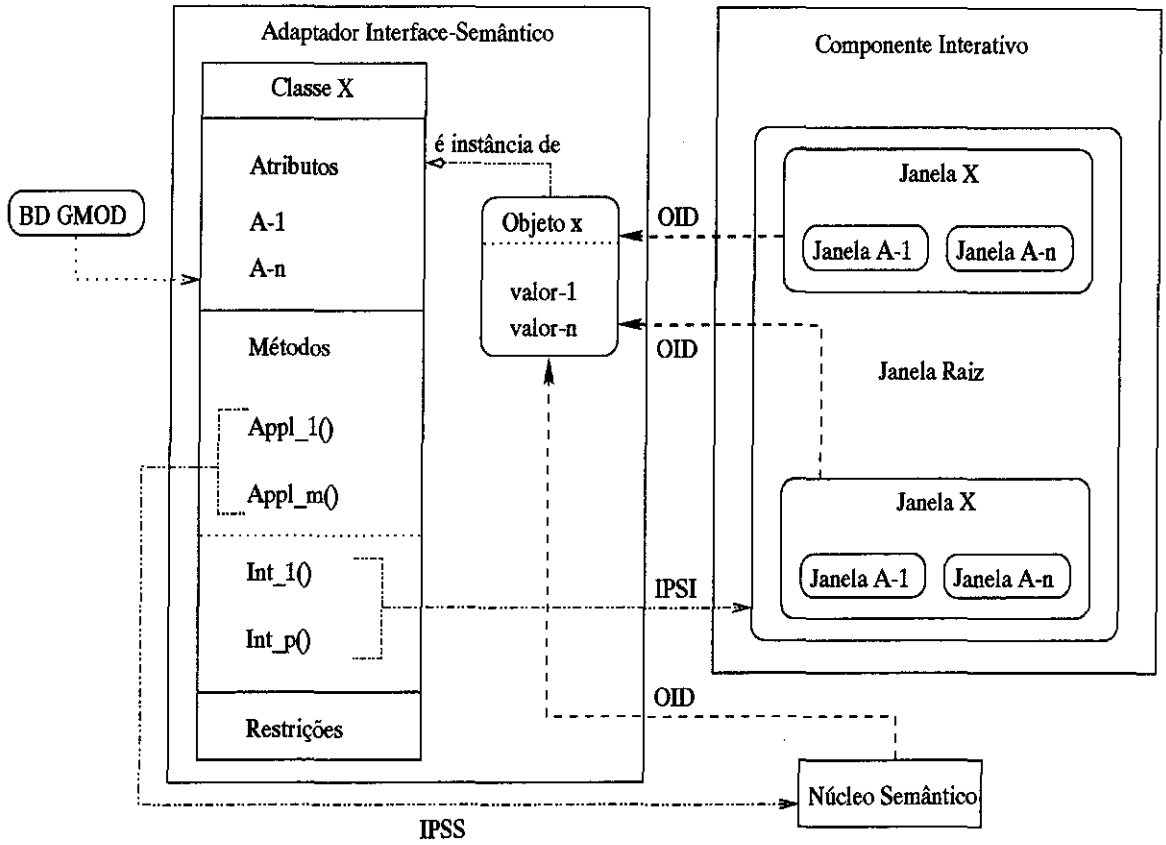

Figura 5.4: Múltiplas apresentações de um objeto conceitual

A figura 5.4 ilustra este esquema, onde duas janelas de classe são usadas para apresentar, na interface, o mesmo objeto conceitual do GMOD (mostrando, por exemplo, duas representações distintas de um objeto geográfico). Esta figura permite identificar dois problemas de comunicação decorrentes da possibilidade de haver múltiplas apresentações (isto é, múltiplas janelas de classe) para o mesmo objeto conceitual do GMOD. Estes problemas e as soluções adotadas são apresentadas a seguir.

5.5.2 Problemas de Comunicação

O primeiro problema desta abordagem é uma dificuldade comum em todas as ferramentas para desenvolvimento de interfaces de usuário: a necessidade de propagar a atualização feita em um conceito da aplicação para as suas diversas apresentações na interface.

O segundo problema é específico da arquitetura proposta nesta tese, e consiste em identificar a(s) janela(s) que devem responder a uma determinada função de interface invocada pelo Núcleo Semântico. Este problema é causado pela total separação entre os componentes semântico e interativo.

Propagação de Atualizações

Praticamente todas as arquiteturas de software de interface apresentam propostas para lidar com o problema de propagar uma atualização feita sobre um dado da aplicação para as diversas representações deste dado na interface. Em sistemas de bancos de dados, em outro nível, o mesmo problema é tratado no contexto de atualizações de visões [Cer96]. No caso de interfaces, há duas abordagens principais:

- *Multicast*: um *multicast* é um *broadcast* destinado a um sub-conjunto específico de destinatários. Diferentes arquiteturas propõem modos distintos de realizar o *multicast* de uma atualização para todas as representações de interface. O modelo MVC define atualizações sobre os objetos *model* [KP88]. Um objeto *model*, ao receber uma atualização, invoca um método predefinido (*Changed*) que gera uma mensagem (*Update*) para todos os objetos *view* associados. Já o modelo PAC prescreve um agente hierarquicamente superior para garantir a consistência entre agentes que implementam múltiplas visões do mesmo conceito [BC91]. O sistema gerenciador de interface Diamant inclui um gerente de representações exclusivamente para manter a correspondência entre representações internas (da aplicação) e externas (da interface) [TZ92].

- *Mecanismo ativo*: ao invés de sobrecarregar a aplicação com o controle da propagação de atualizações, esta abordagem utiliza algum tipo de mecanismo ativo para realizar esta tarefa de maneira independente. Em [NMK91] o mecanismo ativo é implementado por *daemons* que monitorizam a comunicação entre interface e aplicação, repassando as mudanças detectadas para objetos de mapeamento que mantêm o relacionamento entre objeto conceitual e suas representações. As propostas de [DJPaQ94] e [PDDJ96] utilizam facilidades de bancos de dados ativos para manter a consistência entre representações de interface e objetos conceituais do BD. Uma outra variação é apresentada em [TNB+95], na qual um mecanismo ativo dedicado é implementado como módulo de suporte à execução da interface.

Ambas as abordagens mantêm uma lista de objetos interessados em diferentes tipos de atualizações e fazem um *multicast* para estes objetos quando ocorre uma atualização de interesse. A principal diferença é que na primeira abordagem o *multicast* é controlado pela interface (ou pela aplicação), enquanto na segunda abordagem existe um processo em separado para realizar a tarefa.

A arquitetura desta tese adota a solução baseada em bancos de dados ativos. Para isto, o BD geográfico subjacente precisa ser estendido com um mecanismo ativo. O próximo capítulo introduz um mecanismo ativo para resolver problemas de personalização de interface. Este mesmo mecanismo ativo pode ser utilizado para resolver o problema de propagação de atualizações, de acordo com a proposta de [DJPaQ94].

Identificação de Janelas Destinatárias

A separação e independência entre os componentes semântico e interativo é a premissa básica da arquitetura proposta nesta tese. No entanto, este tipo de independência pode acarretar dificuldades na implementação da aplicação, já que os componentes precisam intercambiar informações.

No caso específico desta arquitetura, a dificuldade está em determinar as janelas que devem processar um determinado método de interface solicitado pelo Núcleo Semântico ao Adaptador Interface–Semântico. De acordo com o princípio de independência adotado, esta é uma decisão que concerne somente o componente interativo. O Núcleo Semântico invoca a operação, mas não deve determinar o(s) objeto(s) da interface que serão responsáveis pela sua execução.

A solução para este problema está apresentada na figura 5.4. As solicitações de funções de interface chegam ao Componente Interativo através da API IPSI, e são direcionados para a *Janela Raiz*. Esta é uma janela composta, cujos sub-objetos incluem todas as demais janelas independentes da interface. Uma *janela independente* não está contida em nenhuma outra janela da interface, isto é, a janela só é sub-objeto da Janela Raiz.

Para compreender o papel da Janela Raiz, é importante salientar duas características da classe Janela do IMOD. Primeiro, uma Janela é um objeto de interface abstrato, e não precisa possuir um elemento visual correspondente. Em outras palavras, uma Janela IMOD pode ser utilizada simplesmente como elemento de composição e de organização da interface, e sua definição é ortogonal aos *widgets* de janelas disponíveis em *toolkits* de interface (*base windows* e *pop-up windows*, por exemplo). Em segundo lugar, uma Janela IMOD possui o conhecimento necessário para propagar eventos e dados para os sub-objetos apropriados, isto é, uma janela conhece seus componentes.

Uma idéia semelhante à de uma instância de Janela que engloba todas as janelas de uma interface foi utilizada em [HG97]: um objeto *System* é introduzido para permitir a comunicação entre instâncias isoladas em um modelo orientado a objetos. A Janela Raiz mantém um contexto de interface que determina a sub-janela (ou sub-janelas) que devem receber a solicitação enviada pelo Núcleo Semântico. O projetista de interface, ao definir o comportamento da Janela Raiz em cada contexto, estabelece o destino de cada mensagem que chega ao Componente Interativo.

Existe, desta forma, uma hierarquia de composição em que uma instância de Janela pertence sempre a uma única instância de Janela composta. Toda instância de Janela possui um atributo que indica a instância de Janela imediatamente superior nesta hierarquia (a sua *janela_pai*). As instâncias de Janela independentes são descendentes diretas da instância única de Janela Raiz. O atributo `janela_pai` da Janela Raiz tem valor nulo. O Controle de Diálogo é distribuído ao longo desta hierarquia de composição, do mesmo modo que acontece nas arquiteturas baseadas em agentes (notadamente no modelo

PAC). No entanto, as definições dos objetos e de suas respectivas funcionalidades utilizam abordagens distintas.

5.6 Projeto e Implementação de Interface

Dado o modelo IMOD, a tarefa do projetista de uma interface geográfica reside em construir o modelo de objetos (*template*) que define cada janela da interface. Um modelo de objetos de interface estabelece, em um nível abstrato, a apresentação da interface, o relacionamento entre seus componentes, e o comportamento de cada objeto. A construção deste modelo envolve, em geral, as seguintes etapas:

1. O projeto do Componente Interativo de uma determinada aplicação inicia com a definição de um esquema de BD segundo o modelo intermediário GMOD.

   O esquema do BD é definido cooperativamente pelos projetistas de interface e de aplicação (responsáveis, respectivamente, pelos componentes interativo e semântico da aplicação geográfica). Este esquema estabelece o conhecimento que um componente da aplicação geográfica tem sobre as funções oferecidas pelo outro componente.

2. É feita a geração automática de modelos de objetos de interface para o esquema.

   Cada classe descrita no BD GMOD é utilizada para geração automática de um conjunto de classes no BD IMOD. Assim, cada classe manipulada pela aplicação possui uma janela correspondente gerada automaticamente pelo *Construtor de Objetos de Interface*.

3. O projetista de interface altera, opcionalmente, os modelos definidos.

   Novas janelas podem ser criadas para especificar controles particulares da aplicação, e pode-se estabelecer associações entre os modelos de janelas. Todos os modelos são armazenados em uma Biblioteca de Objetos de Interface (BD IMOD) e recuperados, em tempo de execução, para *construção dinâmica* da interface. Os componentes de uma interface dinâmica são criados pelo módulo Construtor de Objetos de Interface em tempo de execução, com base em valores lidos do BD GMOD. (Em interfaces estáticas, por outro lado, existe um código fixo que define cada um dos componentes de interface em tempo de compilação.)

4. O projetista modifica, opcionalmente, o *comportamento* de cada modelo.

   Os modelos de interface especificados automaticamente podem definir um comportamento *default* para o objeto de interface. Em muitos casos este comportamento precisa ser modificado, e o projetista pode utilizar ferramentas específicas para esta

finalidade. As ferramentas baseadas em diagramas de estado se mostram particularmente eficientes para definir o comportamento de objetos de interface. Em geral, o comportamento de cada componente é expresso por um conjunto de métodos que tratam da apresentação de dados vindos do BD e do controle de eventos originados pelo usuário.

O módulo Construtor de Objetos de Interface, discutido na seção 5.5, oferece suporte tanto para a geração inicial de modelos de objetos de interface quanto para a efetiva utilização dos modelos na construção dinâmica da interface final.

O IMOD pode ser utilizado apenas para projeto, independentemente da estratégia de implementação adotada. Desse modo, o projetista pode planejar seu sistema de interface a partir do modelo de objetos, tendo a liberdade para implementá-lo através de qualquer *toolkit* para construção de interface. Esta não é uma tarefa difícil, já que os conceitos do IMOD, embora mais abstratos, são semelhantes aos usados em *toolkits* de interface.

Uma outra possibilidade é usar o BD IMOD para construir automaticamente a interface. Para isto é preciso que os módulos de suporte Construtor de Interface e Adaptador de *Toolkit* estejam implementados e disponíveis, e que os projetos estejam armazenados no BD IMOD. Este BD é, na verdade, uma Biblioteca de Objetos de Interface, e cada objeto descreve um modelo de interface, conforme discutido a seguir.

5.6.1 Biblioteca de Objetos de Interface

A Biblioteca de Objetos de Interface (BD IMOD) é um BD orientado a objetos que tem como instâncias modelos de objetos de interface. Uma maneira bastante direta de se obter esta estrutura é através do uso de uma metodologia de desenvolvimento orientada a objetos, que traz a vantagem adicional de prover ferramentas para especificação do comportamento da interface. Um bom exemplo é a metodologia UML, que sintetiza os principais métodos de desenvolvimento orientados a objetos, e para a qual já existem ferramentas de modelagem capazes de sintetizar código [HG97].

Assim como o IMOD, a Biblioteca de Objetos de Interface (BD IMOD) segue o paradigma de orientação a objetos. A associação desses componentes a uma metodologia de desenvolvimento também orientada a objetos possibilita um ganho considerável nos projetos de interface. Os custos de desenvolvimento tendem a ser menores à medida que cresce o número de objetos especializados de interface contidos na biblioteca.

A utilização da biblioteca é conceitualmente simples; ela armazena modelos de objetos de vários projetos que podem ser reutilizados ou adaptados. Se nenhum modelo existente satisfaz os requisitos do projeto, é preciso implementar um novo objeto e adicioná-lo à biblioteca.

É claro que este processo de reutilização exige que sejam oferecidas facilidades para pesquisa, recuperação e atualização de objetos da biblioteca. Esta é, em si, uma área de pesquisa em aberto, e a solução para este problema está fora do escopo deste trabalho. A seção 7.2 discute a utilização do IMOD e de uma Biblioteca de Objetos de Interface definida para um ambiente específico, onde ferramentas externas foram utilizadas para resolver os problemas citados.

5.6.2 Construção Dinâmica de Interfaces

Uma janela de interface no IMOD pode ser construída dinamicamente da seguinte maneira:

1. Ambos o componente interativo e o núcleo semântico têm conhecimento sobre o esquema do BD GMOD que descreve as informações manipuladas na aplicação. Este esquema é definido em tempo de compilação, e gerenciado durante a execução da aplicação pelo Adaptador Interface–Semântico.

2. Em tempo de execução, o Construtor de Objetos de Interface utiliza as definições do projetista para gerar o código de criação dos componentes de interface a partir de classes definidas na Biblioteca (BD IMOD).

3. O sistema de interface interpreta os valores enviados pela aplicação, sempre de acordo com as especificações do GMOD, e instancia os modelos correspondentes do BD IMOD, gerando, finalmente, os objetos de interface.

Desta forma os objetos de diálogo não precisam ser definidos estaticamente durante o processo de compilação da interface; ao contrário, eles podem ser inseridos e removidos dinamicamente. A capacidade de construção dinâmica destes objetos possibilita a personalização da interface, o que aumenta a flexibilidade do sistema e conseqüentemente o grau de satisfação do seu usuário. O capítulo 6 apresenta um mecanismo de personalização que explora esta capacidade.

É importante observar ainda que o suporte do modelo de dados GMOD para possibilitar a definição dinâmica de objetos de interface contribui para o isolamento entre aplicação e interface. O próprio modelo de dados serve de canal para intercâmbio de informações entre os dois componentes do sistema, estabelecendo os limites e os compromissos que devem ser honrados para que haja uma perfeita integração entre eles.

5.7 Resumo

Este capítulo apresentou um mecanismo para construção do Componente Interativo da arquitetura de interface, que permite a reutilização de projeto e implementação, e de-

fine uma infra-estrutura adequada para o desenvolvimento sistemático de interfaces para aplicações geográficas. Esta infra-estrutura, baseada em paradigmas e técnicas de engenharia de software, utiliza um modelo de objetos de interface (IMOD) que permite a definição tanto dos *componentes visuais* da interface quanto do *comportamento* associado a estes elementos. Esta proposta é uma variação da abordagem de *geração automática baseada em modelo* (seção 2.3). O IMOD direciona a especificação de *templates* (modelos de objetos) de interface que são armazenados em uma Biblioteca de Objetos de Interface – BD IMOD. O Construtor de Objetos de Interface combina os modelos da biblioteca com dados do BD GMOD para gerar os objetos com os quais o usuário interage na interface.

O IMOD provê aspectos conceituais que refinam o conjunto de diretivas para projeto de interfaces geográficas definido no capítulo 4, permitindo que as diretivas sejam embutidas em uma metodologia de desenvolvimento de propósito geral, orientada a objetos. A Biblioteca de Objetos de Interface (BD IMOD) facilita a transição de projeto para implementação de novos sistemas de interface. Os pontos fortes desta nova abordagem para desenvolvimento de interface são:

- A introdução de um modelo de objetos que permite a descrição de uma interface complexa através da definição recursiva de componentes mais simples.
- A combinação de técnicas de projeto e implementação orientadas a objetos em um único mecanismo que provê a infra-estrutura para reutilização dos componentes de interface. A abordagem torna viável a construção de interfaces dinâmicas, isto é, interfaces cujos componentes são determinados em tempo de execução do sistema.

Uma experiência preliminar no emprego das propostas neste ambiente revelou um ganho real de produtividade não apenas no desenvolvimento, mas também na manutenção e evolução de sistemas de interfaces geográficas (conforme discute o capítulo 7).

É preciso salientar que o objetivo desta tese não é propor uma solução para o problema geral de reutilização de componentes no desenvolvimento de software. A contribuição deste trabalho se restringe ao domínio específico de interfaces geográficas, tendo por base o paradigma de orientação a objetos. Os problemas envolvidos na reutilização de componentes de software em larga escala são numerosos [Jon94]. Mesmo considerando-se um domínio específico, a solução proposta precisa ser estendida para contemplar os aspectos de classificação, recuperação, evolução e transformação de componentes reutilizáveis.

O próximo capítulo mostra como estender esta proposta para personalização de interfaces. O capítulo 7 apresenta um exemplo de utilização desta abordagem na construção de interfaces para aplicações geográficas urbanas e ambientais.

Capítulo 6

Personalização Ativa de Interfaces

6.1 Apresentação

Um dos desafios enfrentados pelos projetistas de interfaces geográficas é a concepção de um sistema de interação que represente adequadamente os conceitos de diferentes tipos de usuários utilizando os modelos e operações oferecidas pelos SIG.

Duas abordagens podem ser utilizadas para enfrentar este problema: *treinamento de usuários*, permitindo que eles se adaptem ao sistema utilizado: e *construção personalizada de interface*, adequando a interface de uma aplicação geográfica a cada tipo de usuário. Ambas as abordagens envolvem custos adicionais na implantação de um sistema geográfico, de modo que uma solução de compromisso, baseada em algum tipo de personalização da interface, é geralmente adotada.

Este capítulo discute uma nova abordagem para mecanismos de personalização em interfaces geográficas, baseada em estender o funcionamento do Construtor de Objetos de Interface apresentado no capítulo 5. As abordagens atuais para personalização de interfaces de SIG são direcionadas para a incorporação de diferentes tipos de apresentação de dados geo-referenciados. Os problemas envolvidos nesta área são complexos, como por exemplo a generalização cartográfica [LR93], para os quais soluções satisfatórias ainda não existem. Os mecanismos de personalização apresentados neste capítulo são direcionados para os problemas de personalização de *controles* e de *comportamento* da interface; o objetivo é apresentar uma solução implementável em termos de viabilidade técnica e de custos.

A seção 6.2 analisa a necessidade de personalização de interfaces em aplicações geográficas e mostra as abordagens já existentes. A seção 6.3 apresenta os principais componentes da nova abordagem de personalização, que são baseados na incorporação, a SGBD espaciais, de facilidades para construção de interfaces. A seção 6.4 ilustra a utilização destes mecanismos em um processo de personalização de uma aplicação para um

perfil específico de usuário. A seção 6.5 resume as principais idéias introduzidas neste capítulo.

6.2 Abordagens para Personalização de Interface

As idiossincrasias das aplicações e dos usuários de SIG são dificilmente contempladas por um mecanismo de interface único. Os problemas existentes para o desenvolvimento de interfaces geográficas envolvem a heterogeneidade de tipos de usuários e aplicações; a coexistência de características gráficas e textuais; e a importância do *contexto espaço-temporal* (escala, tempo, região de trabalho). A combinação de tais fatores forma um *contexto de utilização* que estabelece requisitos distintos de apresentação e manipulação de informações para uma determinada interface.

Considere, por exemplo, uma consulta típica envolvendo um relacionamento métrico: "determine a distância entre duas entidades geográficas". Para determinados usuários, a consulta pode ser calculada em métrica euclideana; outras aplicações podem exigir que as distâncias sejam consideradas em uma rede de caminhos. As entidades envolvidas na consulta podem ser indicadas através do *mouse*, ou identificadas por valores de seus atributos convencionais. Da mesma forma o resultado pode ser apresentado graficamente através dos trechos do caminho entre as entidades, com respectivas distâncias, ou simplesmente como um valor absoluto.

Diante da ausência de um paradigma de interface adequado a todos os contextos de utilização, diversas propostas têm sido apresentadas para lidar com a necessidade de adaptação de interfaces geográficas aos requisitos de um determinado contexto. Estas propostas podem ser sintetizadas nas duas grandes abordagens citadas anteriormente, e apresentam as seguintes deficiências:

- A abordagem de *treinamento* fere princípios básicos de usabilidade na interação usuário–computador. O fato de que o sistema de interface deve se adaptar ao usuário, e não o contrário, é consensual [Shn87, BC91, PH91]. Além disto, o treinamento necessário envolve custos consideráveis.

- Na abordagem de *construção personalizada* de interface o aumento de custos ocorre inicialmente na fase de desenvolvimento, e permanentemente na manutenção do sistema. Estes custos podem tornar o sistema inviável sob a perspectiva de engenharia de software, já que em sistemas interativos complexos mais da metade do código total da aplicação é dedicado à interface com o usuário [MvD91, MR92].

Soluções de compromisso entre estas duas abordagens têm sido adotadas em muitos sistemas, com destaque para as seguintes propostas:

1. Múltiplos paradigmas de interação: a interface é composta de diferentes paradigmas de interação, e o usuário escolhe o paradigma que é mais adequado às suas necessidades.

2. Estereótipos: a interface é adaptada a um conjunto de modelos mentais de usuário. A interação do usuário com o sistema leva a uma classificação em uma determinada categoria (ou estereótipo) de usuário. O sistema de interface funciona de maneira distinta para cada categoria de usuário.

3. Personalização *ad hoc*: a interface é adaptada a cada tipo de usuário, com o auxílio de um *toolkit* de interface. Neste caso, o usuário depende de um programador de interfaces para realizar a personalização desejada. Mais ainda, se a aplicação não tiver a propriedade de independência de diálogo, as modificações necessárias para adaptação da interface podem afetar a funcionalidade da aplicação.

Nenhuma destas propostas resolve o problema de maneira satisfatória. Nos dois primeiros casos não há personalização efetiva, pois o usuário é limitado a um conjunto predefinido de modos de interação. A escolha do modo mais apropriado é feita pelo usuário, no primeiro caso, e pelo sistema de interface, no segundo caso. A criação de um novo modo de interação ou a alteração dos modos existentes não são, em geral, permitidas.

No caso de personalização *ad hoc* existe efetivamente uma adaptação específica do sistema de interface aos requisitos de um determinado contexto de utilização. Entretanto, o custo desta adaptação é muito alto, envolvendo o desenvolvimento de código por um programador especializado em interface.

Uma solução para este problema é acoplar mecanismos à interface para facilitar o processo de personalização. Um exemplo deste tipo de abordagem é descrito em [GHK+96], onde tecnologias de bancos de dados são incorporadas ao sistema de interface e utilizadas para resolver diferentes tipos de problemas relacionados à interação com o usuário. Apesar de representar um progresso com relação à personalização *ad hoc*, esta abordagem gera um outro problema: a incorporação de novos módulos à arquitetura da interface sobrecarrega ainda mais os custos de construção e compromete o desempenho deste componente da aplicação.

A abordagem adotada nesta tese também se baseia na utilização de funções existentes em SGBD para facilitar o desenvolvimento de interfaces personalizáveis. No entanto, a filosofia adotada é oposta à de [GHK+96]: ao invés de incorporar a funcionalidade do SGBD ao sistema de interface, a idéia é estender o SGBD subjacente com mecanismos que facilitem o desenvolvimento de interfaces personalizáveis para as suas aplicações.

Esta abordagem permite personalização efetiva (isto é, a criação/alteração de modos de interação). Os módulos adicionados ao sistema geográfico fazem uso de facilidades já previstas em SGBD, e não comprometem o desempenho da aplicação. Estes módulos,

associados à arquitetura de interface proposta nesta tese, diminuem sensivelmente o custo do processo de personalização.

6.3 Personalização Baseada em Regras

A abordagem de personalização adotada usa os módulos e mecanismos de construção de interface descritos nos capítulos 4 e 5. O objetivo é possibilitar a personalização de uma interface geográfica de uma maneira simplificada, visando superar os dois maiores problemas das abordagens anteriores: conjunto fixo de modos de interação predefinidos e custo elevado do processo de personalização.

6.3.1 Visão Geral do Mecanismo de Personalização

O mecanismo de personalização, discutido a seguir, segue a proposta de [OMC97] para utilização de facilidades de bancos de dados ativos na construção e personalização de interfaces geográficas. O mecanismo se fundamenta na integração de três componentes principais:

- uma Interface Geográfica Genérica, implementada de acordo com a arquitetura de interface proposta no capítulo 4;
- um Construtor de Objetos de Interface, associado a uma Biblioteca de Objetos de Interface (BD IMOD) projetados conforme o modelo IMOD descrito no capítulo 5;
- um Mecanismo de Bancos de Dados Ativo, implementado no SGBD subjacente, que controle o acesso aos dados armazenados no SIG.

A figura 6.1 mostra os relacionamentos funcionais entre estes componentes e a arquitetura geral do mecanismo de personalização.

Inicialmente, é definida uma interface genérica que provê um comportamento e modo de apresentação *default* para a aplicação. A *interface geográfica genérica* (seção 6.3.5) provê a estrutura básica para interação *ad hoc* com o SIG. O sistema ativo permite a personalização da interface de acordo com o contexto específico. O Construtor de Interface utiliza os elementos da Biblioteca de Objetos de Interface (BD IMOD) para construir dinamicamente tanto a interface genérica quanto a personalizada.

As interações do usuário são interpretadas pela interface e transformadas em *eventos de* BD – solicitações de consulta ou atualização – os quais são repassados ao SGBD geográfico subjacente. Estes eventos sobre dados e metadados são interceptados pelo *Mecanismo Ativo* (seção 6.3.3), que ativa *regras de personalização* (seção 6.3.4) apropriadas. Estas

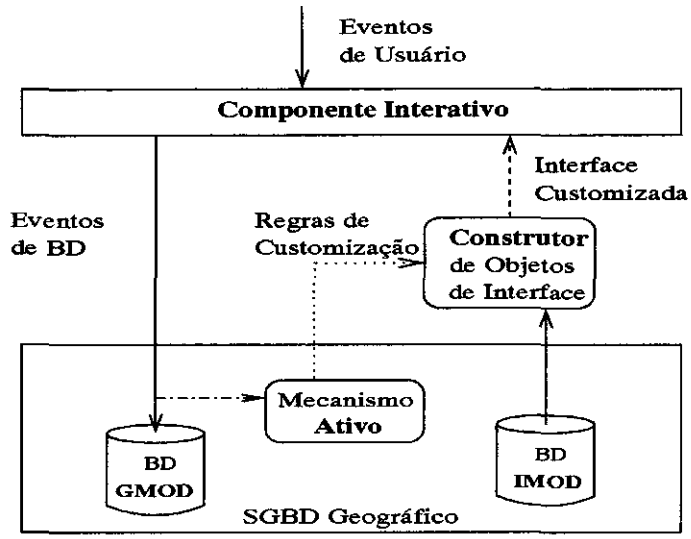

Figura 6.1: Arquitetura e funcionalidade do mecanismo de personalização

regras permitem que o programador redefina o *look-and-feel* da interface, atuando sobre modelos de objetos de interface da Biblioteca de Objetos de Interface (BD IMOD).

Regras e modelos de objetos de interface são processados pelo *Construtor de Objetos de Interface* (seção 6.3.2), gerando uma *definição de interface personalizada*. Esta definição representa um modelo de objetos de interface, que é utilizado para construir dinamicamente os objetos de interface do Componente Interativo (janelas e seus componentes). Estes objetos de interface representam a resposta à solicitação original do usuário.

É preciso observar que a figura 6.1 mostra apenas os detalhes importantes para o mecanismo de personalização. Vale lembrar, ainda, que a aplicação do Mecanismo Ativo de Bancos de Dados nesta tese se restringe aos aspectos de comunicação entre Núcleo Semântico e Componente Interativo (capítulo 5) e de personalização de interface (seção 6.3.3), embora ele possa ser utilizado em outros contextos (manutenção de restrições de integridade, por exemplo).

6.3.2 Biblioteca e Construtor de Objetos de Interface

Cada objeto armazenado na Biblioteca de Objetos de Interface contém a descrição de um modelo de objeto de interface (definição e comportamento genérico). A idéia básica da construção dinâmica de interfaces é a seguinte: o Construtor de Objetos de Interface usa as especificações dos objetos do BD IMOD para construir uma interface. A escolha de objetos apropriados e sua combinação é feita em tempo de execução (em oposição às interfaces pré-compiladas). A personalização é feita em um passo subseqüente.

A Biblioteca de Objetos de Interface (BD IMOD) promove a reutilização de código e de

projeto no domínio de sistemas de interface. O núcleo desta biblioteca é o conjunto de classes do modelo IMOD (seção 5.3).

Este capítulo propõe o uso de um mecanismo ativo de BD para integrar a Biblioteca de Objetos de Interface ao Construtor de Objetos de Interface, visando facilitar o processo de personalização. O leitor interessado pode encontrar um exemplo de utilização de conceitos semelhantes em [OCM95], que mostra a aplicação de bibliotecas de modelos de objetos para o desenvolvimento de um sistema de interface de grande porte (mais de 100 janelas distintas dentro de uma única aplicação geográfica). No entanto, a proposta de [OCM95] não prevê um Construtor de Objetos de Interface genérico, como o proposto nesta tese.

6.3.3 O Mecanismo Ativo de Bancos de Dados

O Mecanismo Ativo se baseia na idéia de estender um sistema de bancos de dados ativo para personalizar a especificação da interface. Sistemas de Bancos de Dados Ativos são SGBD que respondem a eventos gerados interna ou externamente ao sistema sem intervenção do usuário. A dimensão ativa é suportada por mecanismos de regras de produção, oferecidos pelo SGBD. Estas regras são geralmente definidas usando três componentes: **E**vento, **C**ondição, e **A**ção (E-C-A) [Buc94].

O **E**vento representa um sinal que indica a ocorrência de uma situação predeterminada. A **C**ondição é um predicado que deve ser verdadeiro para realizar uma determinada **A**ção. Eventos podem ser internos ao banco de dados (consultas e atualizações, por exemplo), ou externos (tais como interrupções de hardware ou da aplicação).

Uma **C**ondição é, geralmente, um conjunto de predicados sobre o estado do banco de dados. Estes predicados podem ser avaliados realizando uma consulta (expressa, via de regra, através da linguagem de consulta nativa do SGBD).

Ações são expressas na linguagem de programação do SGBD e podem variar de ações atômicas até programas completos. Quando o Mecanismo de Banco de Dados Ativo detecta um evento, ele seleciona um conjunto de regras a serem executadas. Esta execução consiste em avaliar a condição e realizar a ação.

Um amplo espectro de funções de SIG pode se beneficiar das facilidades oferecidas por um Mecanismo Ativo, em diferentes níveis semânticos. Por exemplo, restrições de integridade e ajustes de dados podem ser assegurados por regras nas atualizações e durante entrada de dados geo-referenciados [MC95]. Um outro exemplo é a utilização de mecanismos ativos para propagação de atualizações para as diversas representações de interface referentes a um objeto conceitual do BD (capítulo 5).

Na proposta deste capítulo, o objetivo do Mecanismo Ativo é personalizar a definição do modelo que serve de base para a construção da interface. Neste contexto, **E**ventos são requisições do usuário através da interface (e correspondem, portanto, a pedidos de atualizações e consultas ao BD). A maior diferença está nos componentes de Condição

e **Ação** das regras de produção. Neste caso, uma regra de produção é especificada como mostra a figura 6.2.

On	**E**vento E_i
If	**C**ondição C_j
Then	**A**plique Personalização CT_n sobre objetos de BD $O_1, ..., O_e$ utilizando objetos da Biblioteca de Interface $IO_1, ..., IO_k$

Figura 6.2: Estrutura de uma regra de personalização

A **Ação** – personalização CT – é um código para personalização de interface, escrito na linguagem de personalização descrita na seção 6.3.4. Esta personalização combina dois tipos de objetos de banco de dados: aqueles resultantes do pedido do usuário ao BD geográfico $(O_1, ..., O_e)$ e aqueles recuperados da Biblioteca de Objetos de Interface $(IO_1, ..., IO_k)$. O processo de personalização, descrito na seção 6.4, corresponde a apresentar os dados geográficos em $\{O_1, ..., O_e\}$ de acordo com a especificação de $\{IO_1, ..., IO_k\}$.

O evento E_i é composto de duas partes: um *evento primitivo de banco de dados* e uma *ação de interface*. Ações de interface são eventos tipicamente originados pelo usuário através do *mouse* ou do teclado. Eventos primitivos de banco de dados são as operações primitivas (Get-Schema, Get-Class-Extension, Get-Value, e Exec-Function) definidas no capítulo 4. Estas primitivas são usadas para recuperar e atualizar dados usados nas apresentações. As informações recuperadas pelas operações primitivas são utilizadas pelo Construtor de Objetos de Interface para construção do modelo de interface correspondente. Se não existem regras para o evento, o Construtor usa um procedimento de construção *default*.

A condição C_j corresponde à verificação de um determinado *contexto de aplicação*. Em SIG, um *contexto* descreve, em geral, um ambiente de utilização de aplicação e um modelo mental de usuário. A especificação deste contexto seria demasiado complexa, já que (1) existe um número arbitrário de características que podem compor um modelo mental de usuário; e (2) é possível especificar tantos modelos mentais quanto usuários existentes. Desta forma, um número exponencial de regras de personalização precisaria ser criado.

Por esta razão, a definição de contexto adotada se restringe à tupla <*classe de usuário, domínio de aplicação*>, na qual a classe de usuário e o domínio de aplicação pertencem a partições bem definidas, criadas pelo projetista da aplicação. Esta informação de contexto pode ser conceitualmente estendida para outros dados contextuais (por exemplo, escala

geográfica e estrutura temporal). O fato importante é que Condição não faz a verificação de um estado do banco de dados, mas de um ambiente de trabalho do usuário.

Como ressaltado anteriormente, o componente ativo de um SGBD geográfico também pode ser usado nos papéis tradicionais (autorização de acesso, manutenção de restrições). Sob este ponto de vista, o conjunto de regras pode ser particionado em (pelo menos) dois subconjuntos: regras para personalização de interface, e outras regras (tradicionais). Pelo mesmo motivo, eventos também podem ser considerados de acordo com esta partição entre eventos de interface e eventos (convencionais) de bancos de dados.

Assim, não é necessário um Mecanismo Ativo especialmente desenvolvido para a personalização de interface; é preciso apenas que o Mecanismo Ativo possa manipular um novo conjunto de regras e eventos. Visando este objetivo, eventos devem ser tratados em duas etapas: a interação com o sistema de interface e o efeito no banco de dados. Em outras palavras, mesmo os chamados *eventos atômicos* na terminologia de BD ativos agora se tornam eventos compostos. Um evento atômico de banco de dados AE_i (por exemplo, um pedido de atualização atômico) passa a ser tratado pelo Mecanismo Ativo estendido como um evento composto $IE_i \wedge AE_i$, em que IE_i é um evento de interface (ação sobre o teclado ou *mouse*, por exemplo) e AE_i é um evento tradicional (ação sobre os dados do BD). O evento IE_i é tratado pelas regras de personalização, enquanto AE_i é processado pelo Mecanismo Ativo padrão do SGBD.

É importante observar dois pontos com relação ao sistema de execução de regras neste mecanismo ativo de BD. Primeiro, conflitos podem aparecer com o uso de um Mecanismo Ativo, já que regras podem disparar outras regras conflitantes. Nesta proposta, este não é o caso; a ação de uma regra está limitada à personalização de um objeto de interface. Segundo, é possível existir um conjunto de regras ativadas por um determinado evento. A política de execução das regras em um Mecanismo Ativo é definida por seu *modelo de execução*. Este determina, entre outros, a ordem de execução de regras e os modos de execução (por exemplo, imediato *versus* diferido).

No modelo de execução adotado, se um evento ativa um conjunto de regras, apenas uma regra é executada – aquela que tem a mais alta prioridade. A ordem de prioridade é definida em função da especificidade da regra, isto é, a regra cuja parte de condição (contexto) é mais restritiva tem a maior prioridade. Por exemplo, pode-se definir uma regra para usuários genéricos, para uma categoria particular de usuários, e para um usuário em particular. As três regras podem ser ativadas por um mesmo evento, mas apenas aquela para o usuário específico é executada.

6.3.4 Linguagem de Personalização

A Linguagem de Personalização especifica a parte de **Ação** das regras de personalização. Ela permite a modificação do *look-and-feel* da interface geográfica de acordo com as ne-

cessidades de diferentes usuários. A Linguagem de Personalização provê apenas a ligação para novas funcionalidades dos objetos de interface. A definição destas novas funções está fora do escopo da linguagem.

O usuário alvo desta linguagem é o projetista de aplicações, que tem conhecimento sobre o esquema do banco de dados e sobre as autorizações de acesso do usuário final da interface. A linguagem suporta uma descrição declarativa da parte de controles da interface, em termos de objetos da Biblioteca de Objetos de Interface. O projetista pode adicionar ou especializar objetos de interface nesta biblioteca.

Controles de interface personalizados na interface geográfica genérica correspondem a esquemas, classes, e instâncias de entidades do BD, obtidas através de eventos Get-Schema, Get-Class-Extension, e Get-Value. Estes são os conceitos mais importantes em um banco de dados orientado a objetos geográfico. A simplicidade da linguagem é garantida pela sua natureza declarativa e pela sua sintaxe trivial. O poder de expressão vem da possibilidade de definição de novos comportamentos para os objetos na Biblioteca de Objetos de Interface (ou seja, a codificação de novas funções para substituir o comportamento *default*).

```
For [user <user_name>] [category <category_name>] [application <app_name>]

  schema <schema_name> [display as default | hierarchy | user-defined | null]

    {class <class_name> [display control as <IMOD_object_name>]
                        [display presentation as <format_name>]

      [instances
        {display attribute <attribute_name> as <IMOD_object_name> | null
          [from {<class.attr>} [using <method_name>]]
        }+
      ]
    }*
```

Figura 6.3: As construções básicas da Linguagem de Personalização

A figura 6.3 mostra uma descrição em alto nível da Linguagem de Personalização. Observe que o contexto (componente de Condição da regra) é especificado pela linguagem na cláusula **For**. A cláusula **schema** define a apresentação da informação sobre o esquema do BD geográfico. A apresentação *default* utiliza uma lista com os nomes das classes contidas no esquema. A cláusula **class** especifica a apresentação de cada classe utilizada na aplicação, em termos de objetos de interface utilizados nas área de controle e de

apresentação da Janela de Manipulação de Mapa, descrita na seção 6.4. A cláusula **instances** descreve objetos de interface utilizados para apresentação de cada atributo de uma classe. A cláusula **from** permite agrupar ou redefinir atributos em um objeto de interface complexo; e a cláusula **using** permite redefinir a função *callback* que implementa o comportamento de um objeto de interface.

Como a personalização é baseada em modificar uma interface genérica, não é necessário redefinir todos os controles. O projetista precisa declarar apenas os controles personalizados, sendo os controles omitidos definidos pelos *defaults* da interface genérica.

6.3.5 A Interface de Usuário para SIG

As seções 6.3.2, 6.3.3 e 6.3.4 descreveram mecanismos controlados pelo SGBD geográfico. Esta seção discute a interface geográfica genérica, que corresponde ao Componente Interativo (camada mais externa da figura 6.1).

A interface geográfica genérica suporta diferentes visões conceituais do espaço geográfico. O comportamento *default* desta interface provê facilidades de navegação tanto no nível de dados quanto no nível de esquemas dos bancos de dados geográficos, conforme mostra o exemplo apresentado na seção 6.4.

Cada ação do usuário é capturada pelo módulo lógico de Controle de Diálogo da interface. Este módulo é responsável pela criação e manutenção da hierarquia de objetos de interface, e define o comportamento *default* da interface. A grande diferença deste módulo para os módulos de Controle de Diálogo das interfaces tradicionais é que ele permite a personalização dinâmica e ativa das janelas da interface. O Controle de Diálogo reconhece diferentes tipos de pedidos (manipulações de esquemas e dados), e gera os eventos primitivos capturados pelo Mecanismo Ativo de Banco de Dados.

A personalização de janelas de interface é realizada de modo transparente para o sistema de interface. Todos os módulos lógicos da interface (implementados por objetos de interface do IMOD) possuem o mesmo comportamento, com ou sem personalização, enquanto em interfaces convencionais a personalização envolve a modificação do código (e algumas vezes da estrutura) da interface. Esta transparência é possível porque o Mecanismo Ativo de Banco de Dados se encarrega de disparar as regras de personalização, que modificam objetos da Biblioteca de Objetos de Interface, e tanto o Mecanismo Ativo quanto a biblioteca estão embutidos dentro do SGBD geográfico.

6.4 Exemplo de Personalização

Esta seção descreve o processo de personalização, usando um exemplo simplificado baseado em aplicações geográficas para gerenciamento de redes telefônicas. Uma rede de

telefonia contém centenas de tipos de elementos de rede, tais como postes e caixas de distribuição, que podem ser aéreos ou subterrâneos. O gerenciamento da rede envolve diversas atividades: inclusão de novos elementos, planejamento e projeto de sub-redes, e manutenção da planta.

6.4.1 Janelas da Interface Geográfica Genérica

Considere um banco de dados geográfico que armazena mapas representando os elementos da rede telefônica, acoplado à interface genérica descrita anteriormente. Aplicações típicas sobre este banco de dados exigem consulta e navegação sobre a rede. Uma sessão de trabalho de uma aplicação neste banco de dados pode ser conduzida da seguinte maneira, supondo uma interface construída segundo a proposta do capítulo 5:

1. O usuário inicia uma sessão dando um nome de esquema de banco de dados como parâmetro.
2. Esta ação ativa uma interface geográfica genérica para o esquema.
3. A interface genérica de navegação apresenta uma Janela com a lista de classes que compõem o esquema dado (Janela de Esquema).
4. O usuário seleciona uma classe desta lista e o sistema mostra a extensão da classe em outra janela (Janela de Manipulação de Mapa).
5. O usuário seleciona uma instância da classe nesta janela.
6. O sistema apresenta uma terceira janela descrevendo a instância selecionada.

A figura 6.4 mostra as janelas correspondentes que representam o comportamento *default* da interface. O primeiro tipo de janela é uma Janela de Esquema, onde o usuário pode visualizar o esquema completo do banco de dados (figura 6.4, parte da esquerda). A seleção de classes nesta janela ativa a apresentação do segundo tipo de janela, a Janela de Manipulação de Mapa, que contém uma Janela de Mapa do BD IMOD onde o usuário pode manipular e visualizar as classes selecionadas (figura 6.4, parte central). O terceiro tipo de janela, a Janela de Instância, permite a manipulação e visualização de (geo-)objetos individuais (figura 6.4, parte da direita).

A Janela de Manipulação de Mapa é dividida em duas áreas principais: área de controle e área de apresentação (Janela de Mapa) onde instâncias da Geo-Classe – no caso postes – aparecem como pontos (instâncias gráficas). Estas áreas são compostas, por sua vez, por vários objetos de interface. Por exemplo, a área de controle tem um conjunto de objetos de interface atômicos (botões de menu) e um conjunto de objetos de interface

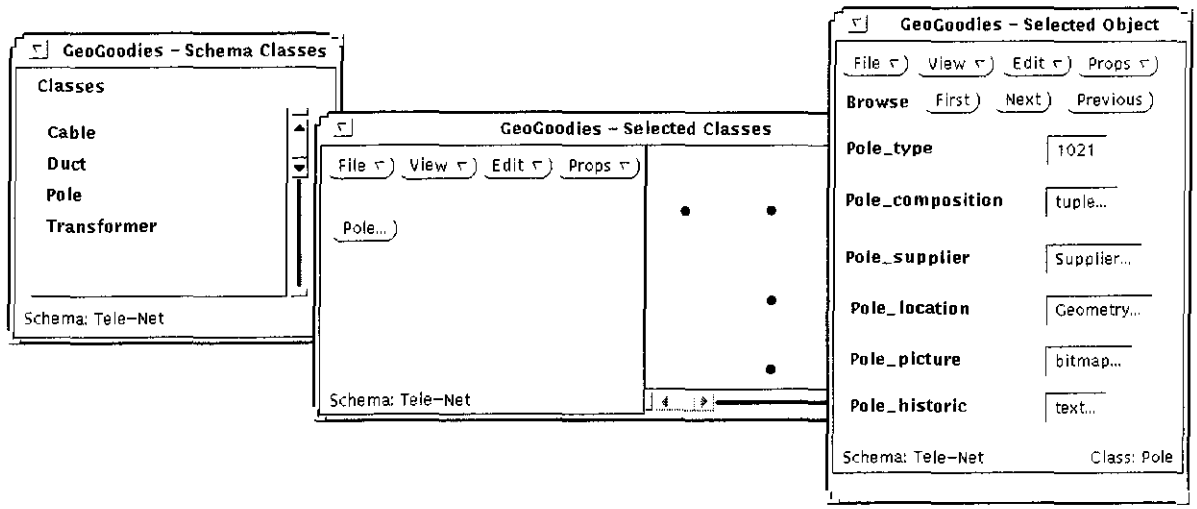

Figura 6.4: Janelas *default* da interface

que representam as classes selecionadas do esquema do banco de dados geográfico. O Construtor de Objetos de Interface utiliza os objetos de interface que descrevem esta estrutura complexa e recursiva para construir cada janela de interface.

6.4.2 Personalização da Interface Geográfica Genérica

Suponha que seja necessário personalizar a interface genérica para um usuário que trabalhe com o controle de postes. Tal usuário não está interessado em todos os elementos da rede, mas simplesmente em elementos do tipo poste. Mais ainda, o usuário quer uma outra visão dos objetos poste, diferente da apresentação *default*.

A figura 6.5 especifica a classe *Pole* (Poste) e a figura 6.6 contém a personalização da sua apresentação. A linha (1) define o contexto < usuário, aplicação >; a linha (2) define o esquema e determina que a Janela de Esquema não será apresentada. Portanto, apenas as classes especificadas na personalização serão apresentadas ao usuário. A classe de interesse para esta aplicação é a classe *Pole*, definida na linha (3). As linhas (4) e (5) definem a apresentação da Janela de Manipulação de Mapa, onde o projetista escolheu um objeto de interface predefinido (poleWidget) para a área de controle e também um formato de visualização predefinido (pointFormat) para a Janela de Mapa. A linha (6) introduz os objetos que são personalizados nas linhas (7) a (12).

Isto simplifica a personalização para a hierarquia de janelas em três níveis para o contexto < *usuário* juliano, *aplicação* pole_manager >. No primeiro nível, o esquema de banco de dados não é apresentado (valor *Null* na linha (2)). As linhas (4) e (5) personalizam a apresentação da classe *Pole*, enquanto as linhas (7) a (12) personalizam instâncias desta classe. Na linha (7), por exemplo, o atributo *pole_composition* é personalizado para ser representado como um objeto de interface predefinido chamado *composed_text*. Os

```
Class Pole{
    pole_type: integer;
    pole_composition: tuple(
        pole_material: text;
        pole_diameter: float;
        pole_height: float;)
    pole_supplier: Supplier;
    pole_location: Geometry;
    pole_picture: bitmap;
    pole_historic: text;
 Methods:
  get_supplier_name(Supplier);}
```

Figura 6.5: A classe *Pole*

```
(1) For user juliano application pole_manager
(2)  schema phone_net display as Null
(3)   class Pole display
(4)     control as poleWidget
(5)     presentation as pointFormat
(6)   instances display attribute
(7)    pole_composition as composed_text
(8)     from pole.material pole.diameter pole.height
(9)     using composed_text.notify( )
(10)  display attribute pole_supplier as text
(11)    from get_supplier_name(pole_supplier)
(12)     display attribute pole_location as Null
```

Figura 6.6: Exemplo de personalização

atributos omitidos (*pole_type*, *pole_picture* e *pole_historic*) são apresentados através da representação *default* definida pela interface genérica. As apresentações *default* seguem a taxonomia de objetos de interface apresentada na seção 5.4.1. Esta personalização resulta na geração de diversas regras, duas das quais são descritas a seguir.

R1: *On* Get-Schema
If < juliano, pole_manager >
Then display < Janela de Manipulação de Mapa > for class < Pole >.

Figura 6.7: Exemplo de uma regra de personalização

Quando o usuário se conecta com a aplicação, um evento Get-Schema é gerado para o banco de dados ativo. A regra correspondente, em uma notação simplificada, é apresentada na figura 6.7

O Construtor de Objetos de Interface cria uma instância de Janela de Esquema, mas não apresenta a janela para o usuário, devido ao parâmetro *Null* especificado na personalização (figura 6.6, linha (2)). A regra **R1** determina que a Janela de Mapa seja ativada diretamente. A ativação da Janela de Manipulação de Mapa para a classe *Pole* gera um evento Get-Class-Extension, necessário para construção da janela. Este evento dispara a regra **R2**, apresentada na figura 6.8.

A execução de **R2** resulta na personalização da Janela de Manipulação de Mapa para

R2: *On* Get-Class < Pole >
If < juliano, pole_manager >
Then display < Janela de Manipulação de Mapa > for class < Pole >
control as < poleWidget >
presentation as < pointFormat >

Figura 6.8: Outro exemplo de regra de personalização

apresentação da classe *Pole*, como mostra a parte da esquerda da figura 6.9. Se o usuário interage com os postes nesta janela, isto irá disparar o terceiro nível de apresentação, para o evento Get-Value, e a ação é a personalização das linhas (7)-(12), resultando na parte da direita da figura 6.9.

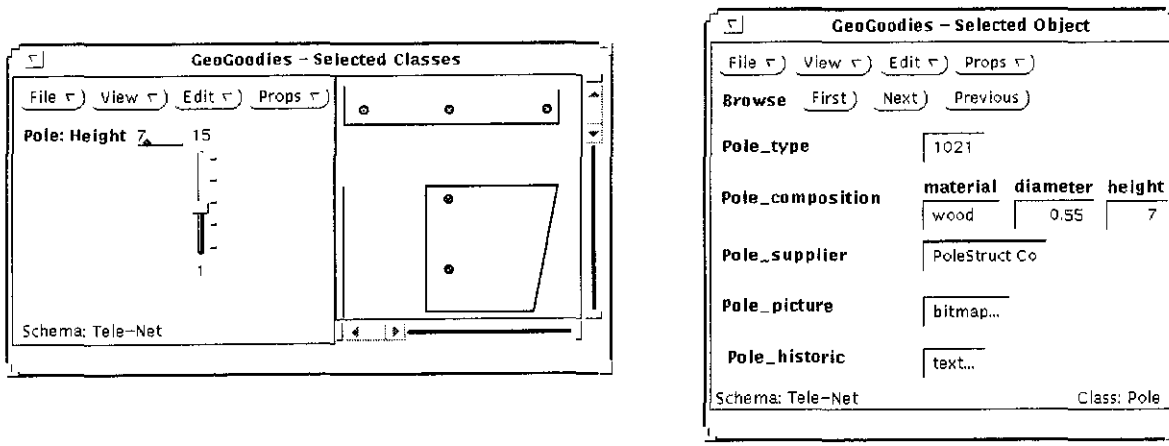

Figura 6.9: Janelas de interface personalizadas

A parte de ação da regra é enviada para o Construtor de Objetos de Interface juntamente com os objetos de interface envolvidos. Por exemplo, a regra **R2** envolve o objeto de interface *poleWidget*. O Construtor substitui este objeto na apresentação *default* e constrói a nova janela de interface com base no modelo de objetos de interface modificado.

6.5 Resumo

Este capítulo descreveu uma nova abordagem para a personalização de interfaces de usuário em aplicações geográficas. Enquanto os trabalhos anteriores centraram o processo de personalização no acréscimo de novos módulos à interface, a abordagem aqui

proposta se baseia na extensão do SGBD subjacente com facilidades para desenvolvimento de interface. Os principais componentes desta abordagem são:

1. uma Interface Geográfica Genérica para SIG, provendo o *look-and-feel* padrão do Componente Interativo.

2. uma Biblioteca de Objetos de Interface que permite a definição e construção dinâmica de componentes de interface.

3. um Mecanismo Ativo de Banco de Dados que garante a independência entre a interface genérica e as regras de personalização.

O componente (1) utiliza a arquitetura de interface proposta nesta tese: o componente (2) foi implementado como parte de um projeto do Centro de Pesquisa da Telebrás (CPqD), mas sem as facilidades de personalização aqui apresentadas [OCM95]; e o componente (3) foi especificado com base em um protótipo de mecanismo ativo utilizado para manutenção de restrições de integridade topológicas em SIG [MC95].

A solução apresentada é superior às abordagens anteriores nos seguintes aspectos: as modificações efetuadas pela personalização não envolvem diretamente o código de interface já existente; os mecanismos são extensíveis, reutilizáveis, e permitem construção dinâmica de interfaces. Além disto, o processo de personalização utiliza as facilidades do SGBD subjacente para ajudar o projetista de interface, ao invés de sobrecarregar a arquitetura de interface com módulos adicionais.

Um outro ponto interessante desta proposta é a identificação de uma nova família de regras a serem tratadas por um SGBD ativo – regras de personalização de interface. Elas diferem das regras padrão não apenas porque seus propósitos são diferentes, mas também porque a especificação da ação é de uma natureza completamente nova.

Os capítulos 4, 5, e 6 contêm propostas teóricas da tese. O próximo capítulo discute a implementação destas propostas em dois sistemas de aplicações geográficas reais.

Capítulo 7

Exemplos de Utilização dos Mecanismos Propostos

7.1 Apresentação

Os resultados relatados nos capítulos 4, 5 e 6 formam um conjunto homogêneo para projeto, implementação, personalização e manutenção de interfaces geográficas. A característica preponderante é a adaptação e utilização de facilidades e soluções já existentes em sistemas de bancos de dados para resolver problemas de interface.

Este capítulo apresenta exemplos de duas aplicações de SIG em que os mecanismos propostos nos capítulos anteriores foram utilizados no projeto e implementação de interfaces geográficas. A seção 7.2 descreve a aplicação urbana que serviu de motivação para grande parte das propostas desta tese. A seção 7.3 mostra que essas propostas são suficientemente gerais para atender aplicações ambientais. A seção 7.4 resume o capítulo.

7.2 Uma Aplicação Urbana

Um dos pontos chave da abordagem do capítulo 5 para construção do Componente Interativo é o uso de uma Biblioteca de Objetos de Interface em uma metodologia orientada a objetos, visando garantir reutilização e dinamicidade ao processo de construção de interfaces. Esta abordagem é resultado da experiência obtida durante o desenvolvimento da camada de interface do projeto SAGRE, no Centro de Pesquisa e Desenvolvimento da Telebrás (CPqD). O projeto SAGRE é um sistema de aplicações geográficas (SAG) que está sendo desenvolvido desde 1991, e que prevê a automatização das atividades ligadas à gerência da rede externa de telefonia no Brasil.

A rede externa é composta por elementos aéreos e subterrâneos, além das canalizações que fazem a ligação entre a residência de um usuário de serviços de telefonia e a estação

telefônica. As atividades de gerência da rede externa incluem: cadastramento, planejamento, expansão e projeto. Estes serviços são executados pelas empresas operadoras do sistema Telebrás.

7.2.1 Projeto da Interface

O IMOD (capítulo 5) é uma extensão do modelo utilizado para a implementação de parte da camada de interface com o usuário do projeto SAGRE, especificamente do módulo que realiza o cadastramento de elementos da rede externa (SAGRE/CAD). No SAGRE, os mapas que representam a rede externa são armazenados em um SIG que provê funções básicas para a manipulação de dois tipos de dado: dados geo-referenciados e dados convencionais.

A arquitetura do projeto SAGRE organiza o SAG em três camadas que provêem, respectivamente, os serviços de interface com o usuário, o encapsulamento do acesso aos dados e o encapsulamento dos conhecimentos sobre rede externa. Um dos componentes da camada de interface é o módulo denominado Gerente de Atributos (GAT), que implementa o acesso aos atributos convencionais da rede externa. Durante uma sessão de trabalho no SAGRE, o usuário visualiza representações gráficas da rede externa (os mapas). A seleção de um elemento da rede externa ativa o módulo GAT que permite a manipulação dos atributos convencionais do elemento selecionado.

O projeto de interface do GAT foi baseado na utilização de uma hierarquia de modelos de objetos de interface, parcialmente apresentada na figura 7.1. Este modelo permite construir, de maneira dinâmica e automática, apresentações para todos os tipos de elementos de rede.

A implementação do GAT utilizando esta filosofia obteve um coeficiente elevado de reutilização de projeto e de código, motivando a utilização da mesma abordagem em outros módulos do SAGRE.

Os benefícios obtidos pelo uso das técnicas propostas podem ser objetivamente avaliados através da comparação com a primeira implementação feita para a interface do GAT, onde foram utilizadas técnicas tradicionais de desenvolvimento baseadas em toolkits de interface.

Os ganhos mais expressivos foram no tempo de desenvolvimento, que foi reduzido em 2/3, e na possibilidade de construção dinâmica de interfaces, que facilita a manutenção do código do sistema.

Comparação com o IMOD

O IMOD estendeu o modelo de objetos de interface usado no projeto de interface do GAT. Uma comparação entre as classes do núcleo do IMOD (figura 5.2) e as classes da interface

do GAT (figura 7.1) permite ver suas semelhanças.

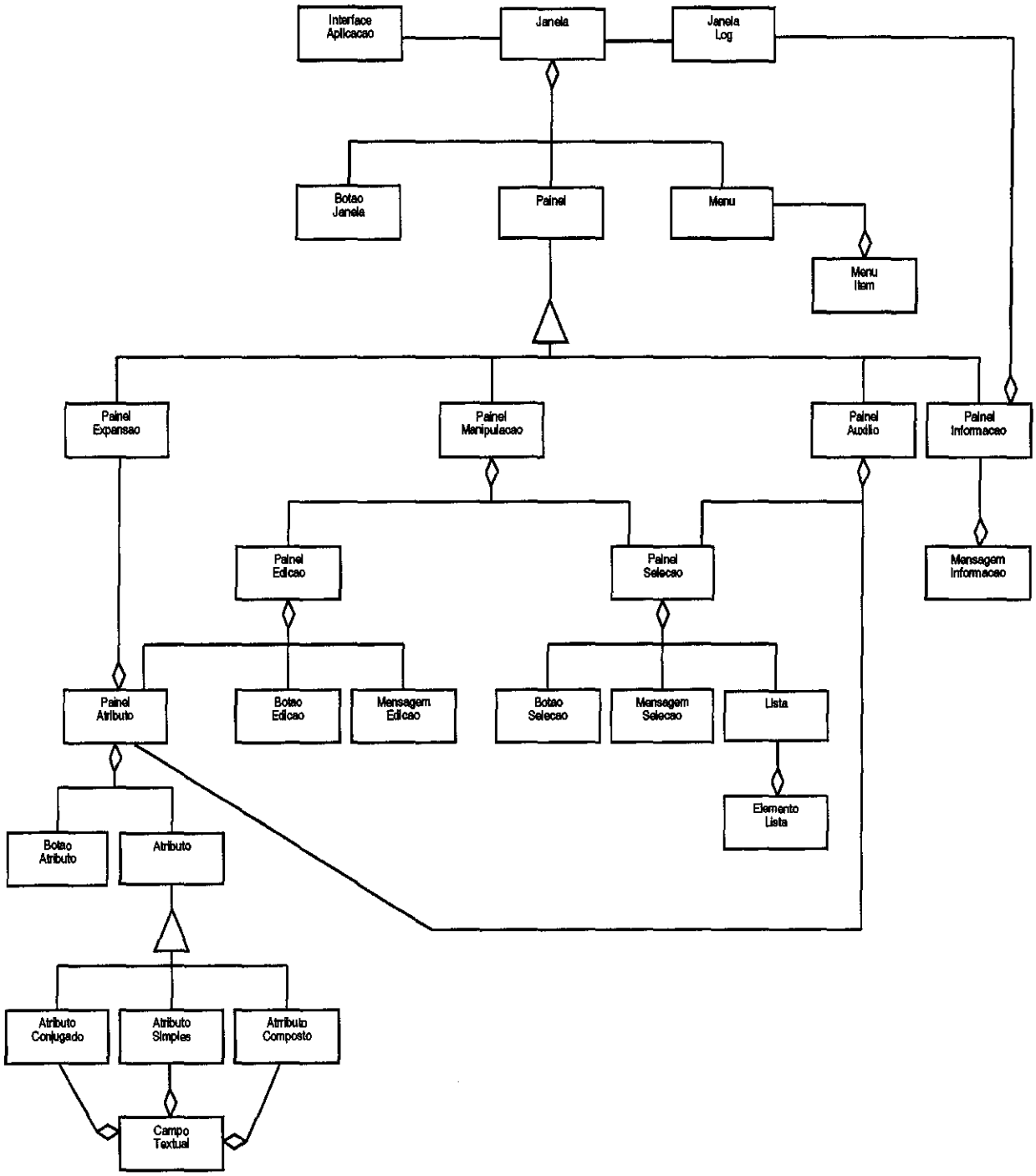

Figura 7.1: Exemplo de Modelo de Objetos de Interface utilizado no GAT

Existem, no entanto, três grandes diferenças estruturais entre o núcleo do IMOD e

o modelo de objetos de interface utilizado no desenvolvimento do GAT [OCM95]. A primeira diferença está na definição de objetos compostos. No IMOD, existe um único elemento de composição (a classe Janela), enquanto no modelo do GAT existem dois tipos de objetos compostos: Janelas e Painéis. Esta duplicidade gera dificuldades para que o elemento componente faça referência ao seu objeto compositor, que pode ser uma Janela ou um Painel. O IMOD elimina esta redundância, de modo que não há ambigüidade nas referências entre objetos nos relacionamentos de composição.

A segunda diferença estrutural entre os modelos está no mecanismo de comunicação entre interface e núcleo semântico. No modelo utilizado no GAT, uma classe de objetos definida na interface é responsável pela interação entre os componentes semântico e interativo. O IMOD (e a arquitetura de interface subjacente) retira esta tarefa do Componente Interativo, colocando-a a cargo do software de suporte (o Adaptador Interface–Semântico).

Finalmente, o modelo do GAT se limita à especificação de objetos de interface para atributos convencionais, enquanto o IMOD define também objetos de interface para atributos espaciais.

7.2.2 Implementação do Modelo

O GAT associa o modelo de objetos de interface a uma biblioteca de objetos para permitir a reutilização de código. Esta seção discute a implementação efetuada utilizando uma plataforma padrão: estações de trabalho executando alguma variante do sistema operacional Unix, usando a linguagem de programação C e os recursos gráficos definidos pelo padrão Motif de interfaces gráficas.

Metodologia de Desenvolvimento

A metodologia utilizada para o desenvolvimento da biblioteca foi a OMT [RBP+91]. Esta metodologia mostrou-se eficiente tanto nas fases iniciais quanto na implementação do projeto. Basicamente são utilizados três modelos para descrever a estrutura e relacionamentos entre objetos (modelo de objetos), o comportamento e alterações de estado de cada objeto (modelo dinâmico) e os processos que transformam os dados durante o funcionamento do sistema (modelo funcional).

Os modelos de objeto e dinâmico provêem a base necessária para a especificação detalhada da estrutura e do comportamento dos elementos da interface. O modelo funcional é secundário, pois em geral os algoritmos que realizam transformações complexas nos dados estão dentro do escopo do núcleo semântico da aplicação, e não da interface com o usuário.

Orientação a Objetos em Linguagem Convencional

O uso da metodologia OMT permitiu manter a visão orientada a objetos, mesmo utilizando uma linguagem convencional (a linguagem C) para a implementação. Para isso, foram definidas regras para a representação dos conceitos de orientação a objetos em uma linguagem convencional. No caso específico da implementação da Biblioteca de Objetos de Interface do GAT, foram utilizadas as seguintes convenções para implementação em linguagem C:

- Cada classe do modelo de objetos foi implementada em um arquivo contendo código fonte em C. Cada arquivo contém a estrutura C (*struct*) que descreve os atributos da classe e as funções que definem os métodos públicos e privados da classe.

- Cada método contém como primeiro parâmetro o identificador do objeto a que ele se refere (um ponteiro para a *struct* C). Nomes de métodos têm como prefixo o nome da classe a que pertencem.

- Toda classe possui pelo menos um método construtor e um método destrutor, responsáveis pela criação e destruição de instâncias da classe. Somente estes métodos alocam e desalocam memória dinâmica.

- A herança de atributos é obtida com a inclusão da definição da estrutura da superclasse (*#include* de C). A herança de métodos é realizada pela delegação de operações à superclasse que implementa o método herdado.

- O conceito de polimorfismo não é implementado. A resolução de chamadas de métodos é feita em tempo de compilação, de acordo com a convenção de nomes de métodos. Além disso, existe um único fluxo de controle (*thread*), já que C não permite tarefas concorrentes.

- Associações entre objetos são implementadas por ponteiros. As estruturas de uma classe são por definição privadas, mas podem ser, como foi visto, herdadas. A classe provê métodos para o acesso necessário aos atributos das instâncias.

Utilizando estas convenções foi possível manter as características do modelo de objetos de interface, embora a linguagem C não ofereça mecanismos diretos para sua representação. Com isso a biblioteca pode ser estendida pela inclusão de novas classes e pela especialização das existentes.

A figura 7.2 mostra partes da implementação da classe Janela, com o objetivo de ilustrar as convenções descritas anteriormente e também para mostrar como pode ser feita a implementação do modelo de objetos de interface em uma biblioteca.

```
typedef struct Janela *JANELA;
struct Janela                                   /* estrutura principal da classe */
{
JANELA jan_janela_associada_ptr[];                    /* auto-relacionamento */
INTERFACE_APLICACAO jan_interf_aplic_ptr;           /* ligacao com aplicacao */
PAINEL jan_painel_ptr[];                              /* paineis agregados */
Widget jan_wd_janela;                                      /* widget Motif */
};

/* prototipos dos metodos publicos da classe Janela */

extern JANELA Janela_constroi(JANELA janela_associada);          /* construtor */
extern void Janela_destroi(JANELA janela_ptr);                    /* destrutor */
extern void Janela_inicializa(JANELA janela_ptr);   /* inicializa componentes */
extern void Janela_inibe(JANELA janela_ptr);       /* bloqueia funcionalidade */
extern void Janela_habilita(JANELA janela_ptr);    /* habilita funcionalidade */
extern void Janela_reg_msg(JANELA janela_ptr, char* msg);     /* registra msg */
```

Figura 7.2: Exemplo de implementação da classe Janela

7.2.3 Comparação com a Proposta da Tese

A implementação do GAT comprovou na prática como os mecanismos do capítulo 5 podem ser usados para construção de interfaces a custo mais reduzido. De fato, o GAT está centrado no uso de uma metodologia orientada a objetos e de uma Biblioteca de Objetos de Interface.

A tese estende os procedimentos de desenvolvimento utilizados no GAT em diversos pontos. O primeiro é o efetivo estabelecimento de uma metodologia de desenvolvimento de interfaces, usando o Construtor de Objetos de Interface e considerando a expansão gradual da biblioteca, onde o Construtor interage com a biblioteca para construção de interfaces geográficas genéricas.

Outra extensão importante é a separação entre os componentes interativo e semântico da aplicação geográfica. No modelo do GAT, o componente interativo define uma classe especial para armazenar uma cópia do modelo de dados utilizado no núcleo semântico (a classe *Interface Aplicação* apresentada na figura 7.1). Na abordagem proposta nesta tese o Componente Interativo fica livre da tarefa de manter a réplica dos dados utilizados na aplicação. O Adaptador Interface–Semântico mantém um BD (BD GMOD) onde as informações manipuladas na aplicação são compartilhadas pelos componentes semântico e interativo.

7.3 Uma Aplicação Ambiental

Uma parte considerável das propostas desta tese está sendo validada na construção da interface para o UAPÉ [OPM97]. O UAPÉ é um ambiente computacional para modelagem e projeto de aplicações geográficas ambientais [Pir97]. O ambiente é voltado para usuários finais que são especialistas em seus respectivos domínios de aplicação, mas que não possuem conhecimentos adequados de bancos de dados e de engenharia de software, e são, portanto, incapazes de tirar proveito total das ferramentas disponíveis nos SIG atuais.

O objetivo do sistema é reduzir a impedância existente entre a visão que os usuários têm do mundo real e a sua implementação em SIG. O sistema foi projetado para ser uma camada auxiliar associada ao SIG, e suas principais características são: a arquitetura aberta, independente de SIG; a abstração de detalhes de implementação, oferecendo ao usuário uma visão conceitual da realidade geográfica; o suporte a uma metodologia de projeto de aplicações geográficas, totalmente integrada com um modelo de dados semântico de alto nível (o GMOD), de forma que não há impedância entre o projeto da aplicação e a modelagem de dados.

7.3.1 Projeto do Ambiente UAPE

O projeto de interface do ambiente UAPÉ foi concebido para suportar dois tipos de atividade do usuário: modelagem de dados e processos; e saída de dados. As atividades de modelagem incluem representação do mundo real, projeto e implementação de aplicações, e especificação de saídas.

O ambiente auxilia o usuário nestas atividades de acordo com uma metodologia de projeto voltada a aplicações ambientais, embutida no ambiente. Os esquemas dos bancos de dados projetados são modelados segundo o GMOD e são mapeados pelo ambiente para o SIG subjacente, de modo a diminuir a diferença entre a visão do usuário e as estruturas usadas para sua representação.

A atividade de produção de saída de dados corresponde a uma seqüência de formulação de consultas e navegações pelo esquema e pelos dados dos BD geográficos, de modo que o usuário tenha uma melhor compreensão dos resultados obtidos. A saída consiste em uma série de mapas, diagramas, e saídas textuais.

A figura 7.3 mostra a arquitetura do UAPÉ sob um ponto de vista conceitual em alto nível. Os módulos desta arquitetura representam a visão que o usuário possui do sistema. Os cinco blocos lógicos do ambiente UAPÉ são responsáveis pelas seguintes funções:

- Módulo de Interface com o Usuário: oferece interfaces gráficas para que o usuário realize as atividades de modelagem e projeto e manipule os bancos de dados ge-

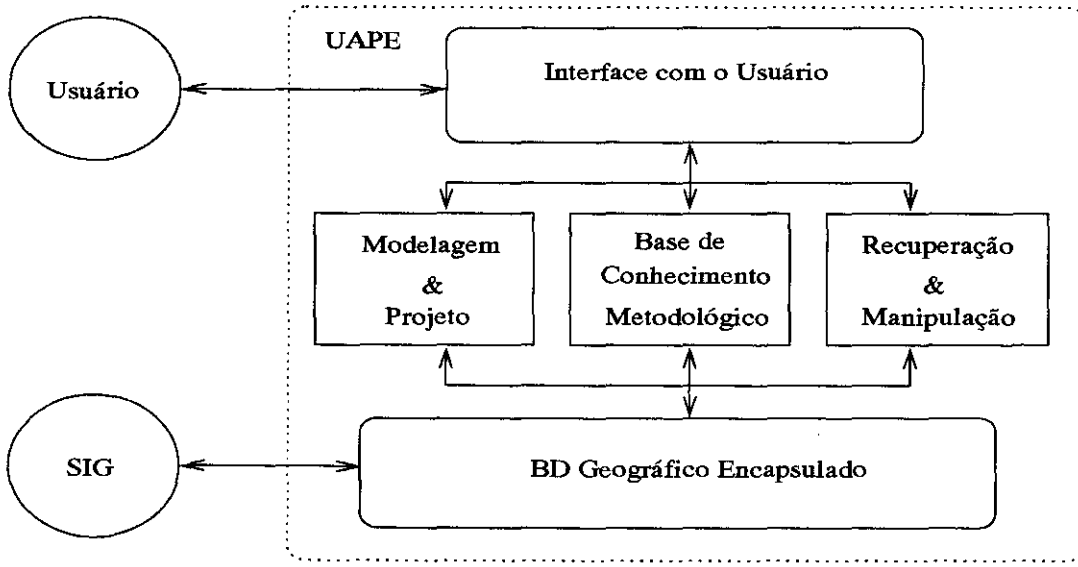

Figura 7.3: A arquitetura conceitual do ambiente UAPE

ográficos. As facilidades oferecidas incluem editores gráficos para projeto de esquemas e ferramentas de navegação e consulta.

- Módulo de Modelagem e Projeto: suporta as atividades de especificação de dados e processos de acordo com o modelo GMOD.
- Módulo Base de Conhecimento Metodológico: assiste ao usuário no projeto da aplicação geográfica, de acordo com a metodologia embutida no UAPÉ.
- Módulo de Recuperação e Manipulação: oferece suporte aos demais módulos no acesso aos dados e meta-dados geográficos.
- Banco de Dados Geográfico Encapsulado: organiza os dados originados do SIG subjacente de acordo com o modelo GMOD.

Uma descrição completa desta arquitetura aparece em [Pir97]. A discussão a seguir está restrita ao projeto e implementação do módulo de Interface com o Usuário.

7.3.2 O Módulo de Interface

O projeto da interface do ambiente UAPÉ foi baseado na arquitetura proposta nesta tese. O módulo de Interface com o Usuário provê um ambiente de interação homogêneo que oferece facilidades de acesso às funcionalidades dos demais módulos do UAPÉ. Sob este ponto de vista, os módulos subjacentes formam o Núcleo Semântico do módulo de interface.

A interface do ambiente precisa realizar duas tarefas básicas: criação e gerenciamento de apresentações, e transformação de operações do usuário em funções específicas dos módulos subjacentes. Como estes módulos adotam os conceitos do GMOD, as tarefas não envolvem mapeamentos complexos de modelos de dados. Uma outra característica importante é que o GMOD também é utilizado para projetar as aplicações geográficas. Desta forma, é possível utilizar a arquitetura de interface proposta nesta tese para geração automática de interfaces para as aplicações projetadas no UAPÉ.

A apresentação das janelas do UAPÉ segue um formato padronizado que permite identificar duas partes principais: uma área de controle, onde se define os critérios de seleção de dados, e uma área de apresentação, onde os dados selecionados são apresentados. As principais funções de interface são as operações gráficas e a geração dinâmica de objetos de interação. Estas funções são definidas, respectivamente, nas áreas de apresentação e de controle.

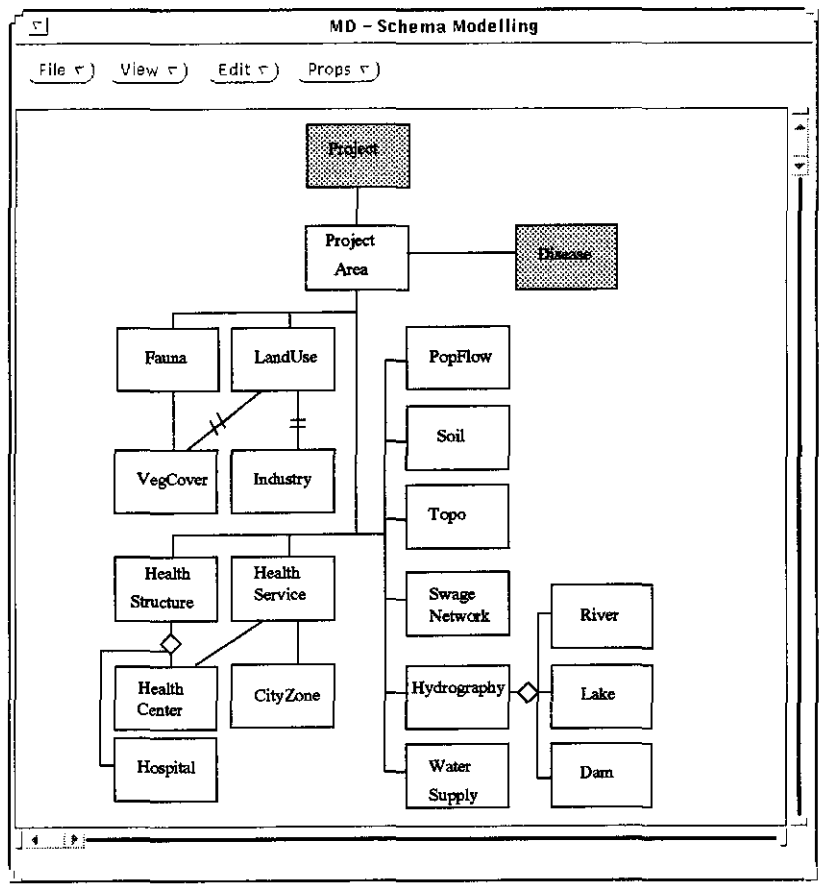

Figura 7.4: Janela do módulo de Modelagem e Projeto

A figura 7.4 mostra um exemplo de janela definida para o módulo de Modelagem e Projeto. A área de controle contém menus que permitem editar uma especificação

diagramática do GMOD, enquanto a área de apresentação contém o diagrama de classes do esquema editado.

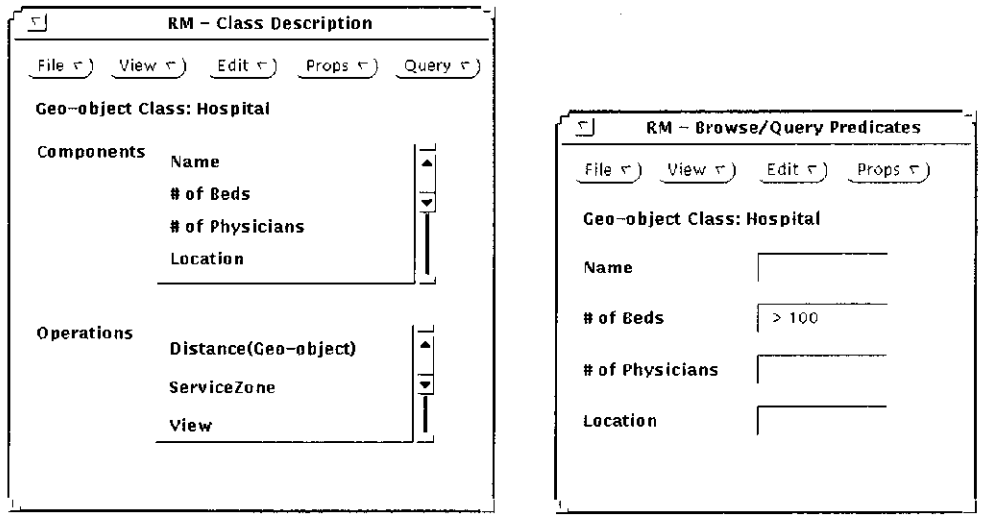

Figura 7.5: Projeto das janelas do módulo de Recuperação e Manipulação

Outro exemplo da interface do UAPÉ aparece na figura 7.5. Estas janelas são definidas para o módulo de Recuperação e Manipulação, e possuem a característica particular de não especificarem uma área de apresentação. Isto ocorre porque o módulo de Recuperação e Manipulação é um "servidor" dos demais módulos; os dados selecionados deste servidor são apresentados em janelas de outros módulos. A janela da esquerda permite que o usuário visualize a definição do esquema de uma classe de algum BD GMOD; na janela da direita o usuário pode especificar predicados que definem as instâncias da classe que serão recuperadas do BD.

Os pontos de destaque da construção da interface do UAPÉ com o auxílio da arquitetura proposta nesta tese foram a visão homogênea de um conjunto de modelos independentes, facilitando o trabalho do usuário, e a integração destes módulos através dos mecanismos de comunicação propostos na arquitetura de interface.

7.4 Resumo

Esta tese introduz mecanismos, modelos, técnicas e ferramentas para auxiliar o projeto, implementação e personalização de interfaces de aplicações geográficas. Este capítulo discutiu a aplicação de partes deste arcabouço teórico no projeto e implementação de interfaces para aplicações geográficas de SAG.

A primeira aplicação está voltada para problemas de gerência de redes de telefonia, enquadrando-se, portanto, na classe de aplicações urbanas da taxonomia de aplicações

geográficas apresentada na seção 1.2.4. O modelo de objetos de interface utilizado para o desenvolvimento da interface desta aplicação é uma versão inicial do modelo de objetos apresentado no capítulo 5.

A segunda aplicação geográfica descrita neste capítulo teve sua interface projetada de acordo com a arquitetura proposta no capítulo 4. Esta aplicação pode ser considerada uma *meta-aplicação ambiental*, pois está voltada para a definição (modelagem e projeto) de aplicações ambientais.

O capítulo descreveu as principais características dessas duas aplicações geográficas, com ênfase no projeto e implementação da interface. Foram discutidas as idéias básicas utilizadas no desenvolvimento das interfaces destas aplicações, e as discussões foram ilustradas com exemplos (de projeto e de implementação) utilizados em cada aplicação.

Capítulo 8

Conclusões

8.1 Apresentação

Esta tese definiu um conjunto de técnicas e modelos para apoiar a construção de interfaces para aplicações geográficas. As propostas do trabalho visam prover suporte de engenharia de software para o desenvolvimento destas interfaces, no sentido de garantir uma transição suave entre as fases de projeto e implementação, e facilitando ainda as tarefas de manutenção. Grande parte das propostas da tese são fundamentadas na utilização de tecnologias de bancos de dados para resolver os problemas das interfaces geográficas.

A próxima seção sintetiza as principais contribuições desta tese. A secao 8.3 apresenta uma revisão geral do requisito de projeto que fundamenta todas as soluções propostas na tese para facilitar a construção de interfaces geográficas. A seção 8.4 faz uma análise do processo de desenvolvimento descrito nesta tese, comparando-o com abordagens já existentes. A seção 8.5 discute os resultados experimentais obtidos com a utilização das propostas da tese. A seção 8.6 mostra possíveis extensões e direções para trabalhos futuros.

8.2 Contribuições da Tese

As técnicas e ferramentas propostas neste trabalho contribuem para a resolução de alguns dos problemas computacionais que dificultam a construção de interfaces para aplicações geográficas. As soluções propostas utilizam teorias de três áreas da computação: Interfaces, Bancos de Dados e Engenharia de Software. Estas teorias são consideradas dentro de um enfoque inovador que envolve dois aspectos fundamentais para aplicações geográficas, os quais não são tratados pelos trabalhos anteriores:

- A interface não é um componente isolado, e tampouco é o único componente de software importante em um sistema geográfico. Ela faz parte de um ambiente hete-

rogêneo, onde o software de suporte precisa ser levado em consideração. Este tipo de abordagem vai de encontro à maioria dos trabalhos em interfaces, que consideram o componente interativo como o "centro do universo", e que não tratam software de suporte a não ser os *toolkits* de interface. Nestes trabalhos ocorre, via de regra, a replicação de funções do SGBD espacial na interface.

- É preciso suportar o projeto e implementação de interfaces para aplicações do usuário, e não apenas para consulta *ad hoc* ao SIG. Praticamente todas as propostas de sistemas de interface para SIG consideram apenas consultas, justamente pelo fato de serem projetados exclusivamente sob a perspectiva de interface, sem levar em consideração os requisitos e particularidades do software geográfico subjacente.

Dentro deste contexto, a tese introduziu novas abordagens para os problemas de concepção e implementação de interfaces geográficas. Os resultados apresentados permitem ao projetista de interface desenvolver um projeto já direcionado à implementação, graças às abstrações e técnicas de desenvolvimento providas:

1. A especificação de uma **arquitetura de software de interface** que organiza os componentes do sistema de interface em camadas, definindo suas funcionalidades e interoperabilidade. O objetivo da arquitetura é organizar o software de interface de modo a facilitar o seu projeto, implementação e manutenção.

2. Um **modelo de dados intermediário** para a interface geográfica que permite formalizar o protocolo de comunicação entre os componentes interativo e semântico da aplicação geográfica. Uma das maiores dificuldades para construção de interfaces geográficas está no mapeamento do modelo conceitual da interface para o modelo físico adotado no SIG subjacente. A solução proposta para esse problema utiliza um modelo intermediário rico em semântica associado a facilidades de bancos de dados ativos para promover um elevado grau de independência entre os dois componentes da aplicação geográfica.

3. Um **modelo de objetos de interface** que suporta o **processo de construção de interfaces dinâmicas**. O modelo permite a especificação de objetos de interface do Componente Interativo da arquitetura de interface. Estes objetos variam em forma e comportamento de acordo com o tipo de aplicação geográfica, e o modelo é suficientemente poderoso para suportar os requisitos de diversos tipos de aplicações. Além disto, o modelo é próximo dos conceitos utilizados pelas ferramentas de implementação, de forma que é possível determinar um mapeamento simples entre o projeto e a implementação da interface.

4. A definição da estrutura necessária para criação de uma **Biblioteca de Objetos de Interface** que pode fazer parte do SGBD subjacente. Foram discutidas as técnicas de projeto (e programação) que permitem o uso deste tipo de biblioteca no domínio específico de interfaces geográficas. O uso dos objetos de interface em uma biblioteca demonstrou uma redução de custos de desenvolvimento e manutenção em aplicações geográficas.

5. Um **mecanismo de personalização** de interface baseado no paradigma de bancos de dados ativo. O mecanismo permite adaptar uma interface genérica, construída com base na arquitetura e no modelo de objetos propostos, para necessidades específicas de um usuário ou grupo de usuários.

6. Uma contribuição adicional do trabalho de tese foi a revisão detalhada das atividades de pesquisa recentes em interfaces para SIG [OM96a, OM96b].

As técnicas e modelos propostos promovem a independência entre os componentes de interface e semântico de uma aplicação geográfica, facilitando os processos de manutenção e evolução destes dois componentes. As soluções propostas foram validadas através da sua utilização no projeto e implementação de interfaces para dois tipos de aplicações geográficas (urbana e ambiental).

8.3 Separação entre Interface e Núcleo Semântico

A premissa fundamental de todas as arquiteturas de software de interface é que os componentes semântico e interativo da aplicação podem ser identificados e adequadamente separados [Edm92]. Estas arquiteturas de interface determinam a organização interna do componente interativo e o mecanismo de comunicação deste com o componente semântico.

A separação entre os componentes semântico e interativo se justifica pelo princípio da modularidade funcional - uma das bases da engenharia de software - que estabelece a divisão de funcionalidades ortogonais em módulos específicos. Diversos benefícios decorrem desta separação, com destaque para a especialização de cada componente em suas próprias tarefas; a possibilidade de acoplamento de interfaces específicas para cada tipo de usuário; e o desenvolvimento e modificação independente de cada componente.

Apesar de desejável, e mesmo necessária, a implementação desta separação entre os componentes não é uma tarefa fácil. A maior dificuldade envolvida é que, embora as funcionalidades dos dois componentes sejam distintas, o componente interativo depende de um certo grau de conhecimento sobre a semântica da aplicação (embutida no núcleo

semântico) para realizar suas funções [HS89]. Portanto, uma dificuldade importante consiste em prover o conhecimento semântico necessário na interface sem comprometer a modularidade da aplicação e a independência de diálogo. Cada arquitetura de interface oferece uma solução diferente para este problema, sendo que as abordagens mais utilizadas envolvem:

1. a comunicação entre os componentes, tipicamente utilizada nas arquiteturas de interface baseadas no modelo de Seeheim [Gre85];

2. a incorporação de conhecimento sobre a aplicação no componente interativo, adotada nas arquiteturas de interface baseadas em agentes [BC91].

O problema na primeira abordagem é a degradação do desempenho da aplicação devido ao excesso de tráfego de informação entre os componentes. Na segunda abordagem ocorre a quebra (ou diminuição) da modularidade da aplicação, e podem surgir problemas de inconsistência e perda de integridade dos dados devido à redundância de informações (isto é, de conhecimento semântico).

A arquitetura de interface proposta nesta tese apresenta uma nova abordagem para este problema, baseada na utilização de um banco de dados compartilhado entre os componentes semântico e interativo da aplicação geográfica. O esquema deste BD expressa objetos e funções intercambiados entre os dois componentes, e está organizado segundo o modelo de dados GMOD, um modelo orientado a objetos direcionado para as necessidades de aplicações geográficas [Pir97]. O GMOD permite representar uma parte considerável da semântica de uma aplicação geográfica, através de três tipos de informação [OPM97]:

- Estruturas de dados: as classes de objetos do GMOD podem representar os dados (geo-referenciados e convencionais) manipulados pela aplicação. Uma característica importante é que a descrição dos dados é independente das estruturas de dados efetivamente utilizadas pelo componente semântico.

- Métodos e funções: as operações disponibilizadas pelo componente semântico para o componente de interface (e vice-versa) são especificadas como métodos das classes ou como funções genéricas definidas no esquema GMOD. Deste modo, o BD reflete não apenas a visão que o componente interativo tem do componente semântico (comum em abordagens anteriores), mas também a perspectiva que o componente semântico tem do componente interativo (frequentemente desconsiderada em arquiteturas de interface).

- Restrições de integridade: o GMOD oferece facilidades para definir diversos tipos de restrições sobre o esquema do BD geográfico, envolvendo aspectos estruturais (restrições estáticas e dinâmicas) e comportamentais (pré- e pós-condições para execução de métodos, invariantes, e temporizadores, entre outros tipos de restrições) [Pir97].

Apesar do seu poder de expressão, pode haver características semânticas da aplicação geográfica que não são adequadamente representadas pelo GMOD. Neste caso, os componentes interativo e semântico podem se comunicar para obter os detalhes semânticos não explicitados no esquema do BD compartilhado.

Esta abordagem não elimina totalmente os problemas das abordagens anteriores, mas reduz sua complexidade e seus efeitos negativos. Sob o ponto de vista de desempenho da aplicação, o tráfego de informações entre os componentes é bastante reduzido, já que grande parte das informações semânticas se encontram no BD compartilhado.

No que diz respeito à redundância de conhecimento, o esquema especificado segundo o modelo GMOD ainda é uma replicação dos dados e procedimentos definidos no componente semântico. Todavia, esta replicação não compromete a modularidade da aplicação pois a informação redundante está isolada em um único local (o BD GMOD) e é devidamente controlada por um módulo de suporte da arquitetura (o Adaptador Interface–Semântico). Nenhuma carga adicional é imposta aos componentes semântico e de interface para gerenciar o BD compartilhado; tudo o que eles precisam fazer é processar as respectivas operações definidas no esquema GMOD.

8.4 Análise das Propostas da Tese

Diversos mecanismos de suporte têm sido propostos para auxiliar o desenvolvimento de interfaces, entre os quais se destacam as arquiteturas de software (para suporte ao projeto) e as ferramentas de interface (para implementação de código) [Mye95]. As propostas desta tese têm o objetivo de auxiliar tanto o projeto quanto a implementação de interfaces para aplicações geográficas; esta seção compara as abordagens propostas na tese com arquiteturas e *toolkits* de interface existentes.

8.4.1 O Projeto da Interface

A definição de uma arquitetura de software apropriada é fundamental para o desenvolvimento, evolução e manutenção de interfaces complexas. Muitas arquiteturas têm sido propostas para atender a diferentes requisitos de sistemas interativos genéricos (seção 2.4) e em particular de aplicações geográficas (seção 3.3).

Um problema comum nestas arquiteturas é que elas são, em geral, modelos abstratos para os quais inexistem ferramentas de apoio à implementação. As ferramentas mais comuns, *toolkits* de interface, oferecem pouca ajuda nesse sentido, e até mesmo dificultam o emprego de determinadas arquiteturas. Na maioria dos casos, não existe um mapeamento simples das decisões de alto nível estabelecidas pelo projeto arquitetônico da interface para os conceitos implementados em *toolkits* de interface.

A ausência de suporte à implementação de arquiteturas de software de interface é resultante da distância semântica existente entre os modelos abstratos das arquiteturas e os paradigmas de implementação adotados pelas ferramentas de interface. Com isso, a implementação passa a ser uma medida de qualidade das arquiteturas: o valor prático de uma arquitetura é diretamente proporcional à facilidade de implementação (através de ferramentas de interface). Duas linhas de pesquisa procuram superar esta impedância semântica entre arquiteturas e ferramentas de implementação.

Na primeira linha de pesquisa, o esforço está direcionado para o desenvolvimento de ferramentas de implementação mais poderosas, capazes de representar os conceitos das arquiteturas abstratas de interface. O sistema Xchart (linguagem Xchart e ambiente de apoio ao emprego da linguagem) é um exemplo típico deste tipo de ferramenta [LBL96, LHL96]. Esta linha de pesquisa tem grande potencial, mas os resultados obtidos ainda não se comparam aos das ferramentas convencionais em termos de disponibilidade e de custo.

A segunda linha de pesquisa direciona os esforços no sentido de definir um mapeamento simples entre conceitos abstratos de arquiteturas de software para técnicas de implementação disponíveis nos *toolkits* de interface atuais. As propostas desta tese seguem esta segunda linha de pesquisa.

A idéia principal é organizar o projeto da interface utilizando o IMOD, um modelo orientado a objetos que define um núcleo extensível de classes de objetos de interface abstratos. As classes deste núcleo correspondem a classes de widgets presentes na maior parte dos *toolkits* de interface atuais. O IMOD permite compor objetos complexos a partir das classes do seu núcleo, e o projeto de uma interface consiste na definição de um conjunto de objetos de interface complexos. Os modelos de objetos podem ser armazenados em uma biblioteca para reutilização em outros projetos de interface.

A experiência adquirida com a aplicação destas propostas, relatada nas seções 7.2 e 7.3, mostra que elas podem ser efetivamente utilizadas tanto para o projeto quanto para a implementação do "código complexo" de uma interface. Os conceitos adotados na arquitetura permitem expressar tanto as propriedades de *interação usuário-computador* quanto os requisitos de *engenharia de software* necessários para suportá-las. A implementação de requisitos de engenharia de software afeta, principalmente, a definição dos diversos módulos de suporte introduzidos na arquitetura de interface, deixando para o Componente Interativo apenas as tarefas diretamente relacionadas ao diálogo com o usuário.

A integração entre os mecanismos de suporte da arquitetura de software de interface e o modelo de construção do Componente Interativo forma um conjunto de técnicas e ferramentas que pode ser usado para suportar, de maneira direta, as atividades de projeto e implementação de interfaces geográficas. O projetista de software pode trabalhar, em cada fase do ciclo de vida da interface, com um nível de abstração apropriado. As técnicas

e ferramentas propostas nesta tese permitem a automatização de certas etapas deste processo, como por exemplo a transição entre um projeto de modelo de objetos de interface e a criação de uma interface que implementa o projeto.

8.4.2 A Implementação da Interface

As pesquisas na área de interfaces usuário–computador vêm produzindo ferramentas úteis para implementação de interfaces, entre as quais se destacam os *toolkits* de interface, que são ferramentas eficientes para implementação de interfaces de propósito geral. A abordagem proposta nesta tese é ortogonal às destas ferramentas, situando-se em um nível mais abstrato na arquitetura de um ambiente de desenvolvimento de interfaces.

Em geral, *toolkits* implementam componentes de interface que seguem o comportamento e a aparência (*look-and-feel*) especificados por algum padrão (por exemplo, *Motif*, *OpenLook* ou *Windows*). No entanto, estas ferramentas não oferecem suporte para estruturação do projeto da interface, ou seja, a especificação de relacionamentos entre objetos de interface é feita de maneira totalmente *ad hoc*.

As classes de objetos de interface definidas no IMOD são mais abstratas que as classes de widgets oferecidas pelos *toolkits* de interface. As classes do IMOD descrevem a estrutura e comportamento da interface, e o seu relacionamento com o modelo de dados (GMOD) subjacente, independentemente do *toolkit* escolhido para a implementação da interface. As classes visuais de um *toolkit* de interface implementam a estrutura e o comportamento de *widgets* específicos. Por exemplo, uma "janela" do IMOD pode representar qualquer dos tipos de "janela" gráfica disponíveis nos diversos *toolkits* (janelas principais ou subordinadas, gráficas ou textuais, entre outras).

Uim/X [Vis93], Motif [Ope91], Xt [NO90] e Xview [Hel90] são exemplos de ferramentas que permitem reutilização de código para criação e gerência de widgets. O modelo IMOD estende este tipo de reutilização para elementos complexos de interface (composições de widgets), servindo de guia para a reutilização efetiva de projetos de interface.

A infra-estrutura de desenvolvimento proposta pode utilizar os recursos oferecidos por *toolkits* na implementação da interface em uma determinada plataforma. No entanto, o projetista é livre para desenvolver seus próprios componentes de interface, e acrescentá-los a uma biblioteca para posterior reutilização. Mais ainda, o projeto de uma interface segue um modelo único, combinando objetos em diferentes níveis de complexidade. Os *toolkits* provêem, via de regra, apenas objetos de interação simples, ficando para o programador a carga de gerenciar elementos de interface inter-relacionados.

O *controle do diálogo* é uma parte essencial da definição de uma interface, que determina o comportamento dos componentes interativos [LL94]. *Toolkits* não oferecem, em geral, ferramentas para especificação e controle do diálogo. O diálogo é diretamente implementado através de funções *callback*. Na abordagem proposta, a especificação do

diálogo é feita no nível de projeto de interface, isto é, na definição do modelo de objetos de interface, utilizando os formalismos para modelagem de comportamento definidos nas metodologias de desenvolvimento orientadas a objetos.

O IMOD não adota um formalismo específico para definição do diálogo. No entanto, existe na comunidade de pesquisa em interfaces um padrão *de facto* para este tipo de modelagem, que é o formalismo conhecido como *Statecharts* [Har87]. Os pontos fortes desta notação são o seu poder de expressão e simplicidade, além de estar presente nas principais metodologias de desenvolvimento orientadas a objetos, existindo, inclusive, ferramentas capazes de síntese de código [HG97]. Utilizando Statecharts (ou uma de suas variações), é possível empregar os mesmos conceitos (orientados a objetos) desde o projeto até a implementação da interface, o que torna este formalismo bastante adequado para os objetivos do IMOD.

8.5 Avaliação dos Resultados Experimentais

A tese propõe uma arquitetura a partir de experiências de implementação e análise de propostas da literatura. Diversas propostas da tese ainda estão em um nível teórico e portanto fica difícil provar objetivamente a sua superioridade prática com relação às abordagens já existentes. No entanto, resultados experimentais de implementação de partes da aquitetura indicam vantagens quanto aos seguintes fatores:

- O conceito de utilização do IMOD como base para construção de componentes reutilizáveis de interface.

  O capítulo 7 compara as diferenças entre o uso de *toolkits* padrão e o emprego do IMOD, mostrando que há um ganho considerável em termos de projeto e implementação. Este ganho foi constatado no desenvolvimento do sistema GAT da Telebrás: o emprego de uma primeira versão da interface usando técnicas padrão de desenvolvimento e uma segunda versão usando uma biblioteca de modelos de objetos de interface acusou um ganho de tempo de codificação (em meses) de mais de 2/3. Além disto, com a utilização da biblioteca a interface passou a ser construída dinamicamente, a partir dos modelos, e não através de um código fixo e predefinido para cada componente de interface.

- O conceito de separação entre as camadas da arquitetura como fator de suporte à independência de dados e ao encapsulamento de funcionalidade.

  O desenvolvimento da interface do ambiente UAPÉ, conforme discute o capítulo 7, mostra que diferentes módulos desenvolvidos por diversos implementadores puderam ser combinados no nível de interface, oferecendo ao usuário final um ambiente

homogêneo de interação. Os fundamentos básicos utilizados, no caso, foram o uso de primitivas comuns de comunicação com o SIG (*Get-Schema*, *Get-Class-Extension*, *Get-Value*, e *Exec-Function*) e a utilização do GMOD como modelo de dados intermediário da interface.

Falta, ainda, mais experiência de implementação e mais dados concretos de uso para permitir comparar a abordagem de desenvolvimento proposta na tese com as propostas já existentes. Falta, também, identificar métricas adequadas para realizar esta comparação. A própria especificação e comparação destas métricas já constitui uma possível extensão desta tese. Outras extensões são discutidas na próxima seção.

8.6 Algumas Extensões e Trabalhos Futuros

Esta tese estabeleceu as linhas gerais de uma estrutura de suporte ao projeto, implementação, manutenção e personalização de interfaces geográficas. Embora uma parte considerável desta estrutura tenha sido validada experimentalmente, muito trabalho de implementação e de pesquisa ainda precisa ser feito para se obter um ambiente ideal de desenvolvimento de interfaces geográficas.

Uma primeira extensão seria a implementação dos módulos de suporte da arquitetura descrita no capítulo 4. Estes módulos realizam duas tarefas importantes: (1) garantem a independência da aplicação geográfica com relação à plataforma de implementação; e (2) mantêm a separação entre os componentes semântico e interativo da aplicação.

É importante que os módulos de suporte, notadamente os Adaptadores da arquitetura, sejam implementados visando eficiência, já que boa parte dos fluxos de informação passa por estes módulos. Este tipo de preocupação não foi considerado, por exemplo, no desenvolvimento do protótipo discutido na seção 7.3. Além disso, o Adaptador de *Toolkit* não foi efetivamente implementado. Este adaptador reflete uma característica importante da arquitetura de interface proposta nesta tese: a flexibilidade para adequação a diferentes requisitos de projeto. No caso do protótipo implementado, a independência de *toolkit* não era um requisito de projeto, de forma que o Adaptador de *Toolkit* pôde ser simplesmente eliminado, sem prejuízo das demais propriedades da arquitetura.

Outra extensão que envolve esforço de implementação adicional é a integração do protótipo de mecanismo ativo apresentado em [Cil96] a um SGBD espacial, e o desenvolvimento do compilador para a Linguagem de Personalização. Com isto seria possível implementar o mecanismo de personalização ativa de interfaces proposto no capítulo 6.

Do ponto de vista de pesquisa teórica, uma extensão interessante seria a incorporação de um Mecanismo de Visões ao servidor do BD GMOD. Este tipo de mecanismo é útil nos casos em que existe uma grande distância semântica entre o modelo mental do usuário

(representado pelos objetos IMOD) e o modelo de dados utilizado na aplicação (representado pelo BD GMOD). A extensão pode seguir a abordagem introduzida em [OCM97], que propõe o uso de um mecanismo de visões para facilitar o mapeamento entre o modelo mental do usuário e os conceitos do modelo de dados do SIG subjacente.

Outra extensão do ponto de vista teórico é o estudo das implicações de incorporar ao SGBD ativo regras de personalização de interface, cuja semântica e funcionamento diferem totalmente das encontradas em mecanismos ativos convencionais. É possível que este estudo resulte em novos modelos de execução e processamento de regras no contexto de bancos de dados ativos.

Por fim, é preciso investigar a viabilidade de utilização das propostas desta tese em domínios de aplicação relacionados. Este tipo de investigação deve identificar as propostas que podem atender a diversos domínios especificando, ainda, extensões necessárias para atender aos requisitos de diferentes domínios de aplicação. Sistemas de aplicações de CAD são candidatos naturais para este estudo, pois compartilham muitos dos problemas existentes em interfaces geográficas, apesar de possuirem suas próprias idiossincrasias.

Bibliografia

[ABC+91] J. Antenucci, K. Brown, P. Croswell, M. Kevany, e H. Archer. *Geographic Information Systems - a guide to the technology*, capítulo 3 - Applications. Van Nostrand Reinhold, 1991.

[ABD+89] M. Atkinson, F. Bancilhon, D. DeWitt, K. Dittrich, D. Maier, e S. Zdonik. The Object-Oriented Database System Manifesto. In *Proc. 1st International Conference on Deductive and Object-Oriented Databases*, pp. 40–57, dezembro de 1989.

[AKK94] M. Arikawa, H. Kawakita, e Y. Kambayashi. Dynamic Maps as Composite Views of Varied Geographic Database Servers. In *Proc. International Conference on Applications of Databases*, 1994.

[AM95] C. Aguiar e C. Medeiros. Uma arquitetura para integrar bancos de dados heterogêneos aplicada a sistemas de planejamento urbano. In *Proc. XXII SEMISH*, pp. 551–562, 1995.

[AYA+92] D. Abel, S. Yap, R. Ackland, M. Cameron, D. Smith, e G. Walker. Environmental Decision Support System Project: an Exploration of Alternative Architectures for Geographical Information Systems. *International Journal of Geographical Information Systems*, 6(3):193–204, 1992.

[BC91] L. Bass e J. Coutaz. *Developing Software for the User Interface*. Addison-Wesley, 1991.

[BD94] M. Becker e J. Diaz-Herrera. Creating Domain Specific Libraries: a methodology and design guidelines. In *Proc. IEEE 3rd International Conference on Software Reuse: Advances in Software Reusability*, novembro de 1994.

[BDK92] F. Bancilhon, C. Delobel, e P. Kanellakis, editores. *Building an Object-Oriented Database System: The Story of O2*. Data Management Systems. Morgan Kaufmann Publishers, 1992.

[Bee89] C. Beeri. Formal Models for Object-Oriented Databases. In *Proc. 1st International Conference on Deductive and Object-Oriented Databases*, pp. 370–395, 1989.

[BF94] F. Bancilhon e G. Ferran. ODMG-93: The Object Database Standard. *Bulletin of the Technical Committee on Data Engineering*, 17(4):3–14, 1994. Special Issue on Emerging Object Query Standards.

[BFL+92] L. Bass, R. Faneuf, R. Little, N. Mayer, B. Pellegrino, S. Reed, R. Seacord, S. Sheppard, e M. Szczur. A Metamodel for the Runtime Architecture of an Interactive System. *SIGCHI Bulletin*, 24(1):32–37, janeiro de 1992.

[BM91] E. Bertino e L. Martino. Object-Oriented Database Management Systems: Concepts and Issues. *IEEE Computer*, 24(4):33–47, abril de 1991.

[BM92] P. Boursier e M. Mainguenaud. Spatial Query Languages: Extended SQL vs Visual Languages vs Hypermaps . In *Proc. 5th International Symposium on Spatial Data Handling*, pp. 249–259, 1992. Volume 1.

[Buc94] A. Buchmann. Active Object Systems. In A. Biliris A. Dogac, M. Oszu e T. Sellis, editores, *Advances in Object oriented Database Systems*, pp. 201–224. Springer Verlag, 1994.

[CCH+96] G. Câmara, M. Casanova, A. Hemerly, G. Magalhães, e C. Medeiros. *Anatomia de Sistemas de Informação Geográfica*. Décima Escola de Computação, julho de 1996.

[CCN97] G. Calvary, J. Coutaz, e L. Nigay. From Single-User Architectural Design to PAC*: a Generic Software Architecutre Model for CSCW. In *Proc. ACM Conference on Human Factors in Computing Systems (CHI'97)*, pp. 242–249, março de 1997.

[Cer96] N. Cereja. Visões em Sistemas de Informações Geográficas – modelo e mecanismos. Master's thesis, Instituto de Computação – Unicamp, dezembro de 1996.

[CFS+94] G. Câmara, U. Freitas, R. Souza, M. Casanova, A. Hemerly, e C. Medeiros. A model to cultivate objects and manipulate fields. In *Proc. 2nd ACM Workshop on Advances in GIS*, pp. 30–37, 1994.

[CHF93] M. Casanova, A. Hemerly, e A. Furtado. Cooperative Environments For Geographic Databases: A Prescriptive Analysis. In *Anais do Oitavo Simpósio Brasileiro de Banco de Dados*, pp. 266–279, 1993.

[Cil96] M. Cilia. Banco de Dados Ativos como Suporte a Restrições Topológicas em Sistemas de Informação Geográfica. Master's thesis, Instituto de Computação – Unicamp, março de 1996.

[CM91] D. Calcinelli e M. Mainguenaud. The Management of the Ambiguities in a Graphical Query Language for GIS. In *Proc. 2nd Symposium on Spatial Database Systems*, pp. 99–118. Spring Verlag Lecture Notes in Computer Science 525, 1991.

[Cou92] H. Couclelis. People Manipulate Objects (but Cultivate Fields): Beyond the Raster-Vector Debate in GIS. In *Proc. International Conference on GIS - From Space to Territory: Theories and Methods of Spatial Reasoning*, Springer Verlag Lecture Notes in Computer Science 639, pp. 65–77, 1992.

[CWC94] A. Cima, C. Werner, e A. Cerqueira. The Design of Object Oriented Software with Domain Architecture Reuse. In *Proc. IEEE International Conference on Software Reuse: Advances in Software Reusability*, novembro de 1994.

[Dit94] K. Dittrich. *Advances in Object-Oriented Database Systems*, pp. 29–45. Springer Verlag, 1994. Chapter Object-Oriented Data Model Concepts.

[DJPaQ94] O. Diaz, A. Jaime, N. Paton, e G. al Qaimari. Supporting Dynamic Displays Using Active Rules. *ACM SIGMOD Record*, 23(1):21–26, 1994.

[Edm92] E. Edmonds. *The Separable User Interface*, capítulo The Emergence of the Separable User Interface, pp. 5–18. Academic Press, 1992.

[EF88a] M. Egenhofer e A. Frank. Designing Object-oriented Query Languages for GIS: Human Interface Aspects. In *Proc. 3rd International Symposium on Spatial Data Handling*, pp. 79–98, 1988.

[EF88b] M. Egenhofer e A. Frank. Towards a Spatial Query Language: User Interface Considerations. In *Proc. International Conference on Very Large Data Bases*, pp. 124–133, 1988.

[Ege92] M. Egenhofer. Why not SQL! *International Journal of Geographical Information Systems*, 6(2):71–86, 1992.

[Ege94] M. Egenhofer. Spatial SQL: A Query and Presentation Language. *IEEE Transactions on Knowledge and Data Engineering*, 6(1):86–95, 1994.

[EH93] M. Egenhofer e J. Herring. *Human Factors In Geographical Information System*, capítulo Querying a Geographical Information System, pp. 124–135. Belhaven Press, 1993.

[Env92] Environmental Systems Research Institute, Inc. *ARC-INFO: GIS Today and Tomorrow*, 1992.

[FG90] A. Frank e M. Goodchild. Two Perspectives on Geographical Data Modelling. Technical Report 90-11, National Center for Geographic Information and Analysis, 1990.

[GG96] D. Gentner e J. Grundin. Design Models for Computer-Human Interfaces. *IEEE Computer*, 29(6):28–35. 1996.

[GHK+96] N. Gopal, C. Hoch, R. Krishnamurthy, B. Meckler, e M. Suckow. Is GUI Programming a Database Research Problem? In *Proc. ACM SIGMOD International Conference on Management of Data*, pp. 517–528, 1996.

[Goh89] P-C. Goh. A Graphic Query Language for Cartographic and Land Information Systems. *International Journal of Geographical Information Systems*. 3(3):245–256, 1989.

[Gou93] M. Gould. *Human Factors In Geographical Information System*, capítulo Two views of the user interface, pp. 101–110. Belhaven Press, 1993.

[GR93] O. Günter e W. Riekert. The Design of GODOT: an object-oriented Geographic Information System. *Bulletin of the Technical Committee on Data Engineering*, 16(3):4–9, setembro de 1993. Special Issue on Geographical Information Systems.

[Gre85] M. Green. Report on Dialogue Specification Tools. In G. E. Pfaff, editor, *User Interface Management Systems*, pp. 9–20. Springer-Verlag, 1985.

[Gut88] R. Guting. Geo-relational Algebra: a Model and Query Language for Geometric Database Systems. In *Proc. EDBT Conference*, pp. 506–527. 1988.

[Gut92] J. Guttenberg. Towards a Behavioral Theory of Regionalization. In *Proc. International Conference on GIS - From Space to Territory: Theories and Methods of Spatial Reasoning*, Springer Verlag Lecture Notes in Computer Science 639, pp. 110–121, 1992.

[Har87] D. Harel. Statecharts: A Visual Formalism for Complex Systems. *Science of Computer Programming*, 8(3):231–274, June de 1987.

[Hel90] D. Heller. *XVIEW Programming Manual*, volume 7 de *The X Window System Series*. O'Reilly & Associates, abril de 1990.

[HG97] D. Harel e E. Gery. Executable Object Modeling with Statecharts. *IEEE Computer*, 30(7):31–42, julho de 1997.

[HH89] H. Hartson e D. Hix. Human–Computer Interface Development: Concepts and Systems for its Management. *ACM Computing Surveys*, 21(1):5–92, março de 1989.

[HS89] W. Hurley e J. Sibert. Modelling User Interface–Application Interactions. *IEEE Software*, 6(1):77–77, 1989.

[Jon94] C. Jones. Economics of Software Reuse. *IEEE Computer*, 27(7):106–108, 1994.

[KP88] G. Krasner e S. Pope. A Cookbook for Using the MVC User Interface Paradigm in Smalltalk. *Journal of Object-Oriented Programming*, 1(3):26–49, 1988.

[Kuh91] W. Kuhn. Are Displays Maps or Views? In *Auto-Carto 10*, pp. 588–598, 1991.

[LBL96] F. Lucena, L. Buzato, e H. Liesenberg. Xchart: Um Sistema de Gerenciamnto de Interfaces Homem-Computador. In *Proc. Workshop de Sistemas Hipermidia (WoSH'96/SBRC'96)*, maio de 1996.

[LHL96] F. Lucena, M. Harada, e H. Liesenberg. Operational Semantics of Extended Statecharts (Xchart). In *Proc. 3rd Workshop on Logic, Language, Information and Computation (WoLLIC'96)*, maio de 1996.

[LL94] F. Lucena e H. Liesenberg. Reflections on Using Statecharts to Capture User Interface Behaviour. In *Proc. XIV International Conference of the Chilean CSS*, outubro de 1994.

[LR93] J. Lagrange e A. Ruas. Etat de l'art en generalization. Technical report, IGN/DT/SR/COGIT, abril de 1993.

[LS92] M. Lindholm e T. Sarjakoski. User Models and Information Theory in the Design of a Query interface for GIS. In *Proc. International Conference on GIS - From Space to Territory: Theories and Methods of Spatial Reasoning*, Springer Verlag Lecture Notes in Computer Science 639, pp. 328–347, 1992.

[MC95] C. Medeiros e M. Cilia. Maintenance of Binary Topological Constraints through Active Databases. In *Proc. 3rd ACM Workshop on Advances in GIS*, pp. 127–134, 1995.

[MD91] D. Maguire e E. Dangermond. *Geographical Information Systems - volume I - Principles*, capítulo The Functionality of GIS, pp. 319–335. John Wiley and Sons, 1991.

[MGR91a] D. Maguire, M. Goodchild, e D. Rhind, editores. *Geographical Information Systems - Principles and Applications*, volume 1 - Principles. John Wiley and Sons, 1991.

[MGR91b] D. Maguire, M. Goodchild, e D. Rhind, editores. *Geographical Information Systems - Principles and Applications*, volume 2 - Applications. John Wiley and Sons, 1991.

[MM91] J. Mamou e C. B. Medeiros. Interactive Manipulation of Object-Oriented Views. In *Proc. IEEE International Conference on Data Engineering*, pp. 60–69, abril de 1991.

[MMM+97] B. Myers, R. McDaniel, R. Miller, A. Faulring, B. Kyle, A. Mickish, A. Klimovitski, e P. Doane. The Amulet Environment: New Models for Effective User Interface Software Development. *IEEE Transactions on Software Engineering*, 23(6):347–365, junho de 1997.

[MR92] B. Myers e M. Rosson. Survey on User Interface Programming. In *Proc. Human Factors in Computing Systems*, pp. 195–202, May de 1992.

[Mra91] H. El Mrabet. Outils de Generation de Interfaces: Etat de La Art et Classification. Technical report, Institut National de Recherche en Informatique et en Automatique, fevereiro de 1991. Rapport Techiniques 126.

[MSH93] D. Medyckyj-Scott e H. M. Hearnshaw, editores. *Human Factors In Geographical Information System*. Belhaven Press, 1993.

[MvD91] A. Marcus e A. van Dam. User-Interface Developments for the Nineties. *IEEE Computer*, 24(9):49–57, setembro de 1991.

[MW92] J. Morrison e K. Wortman. Cartography and Geographic Information Systems - Special Issue: Implementing the Spatial Data Transfer Standard. *American Congress on Surveying and Mapping*, 19(5), 1992.

[Mye95] B. Myers. User Interface Software Tools. *ACM Transactions on Computer-Human Interaction*, 2(1):64–103, março de 1995.

[NMK91] H. Nakatsuyama, M. Murata, e K. Kusumoto. A New Framework for Separating User Interfaces from Application Programs. *SIGCHI Bulletin*, 23(1):88–91, 1991.

[NO90] A. Nye e T. O'Reilly. *X Toolkit Intrinsics Programming Manual*, volume 4 de *The X Window System Series*. O'Reilly & Associates, setembro de 1990.

[Nor91] K. Norman. Models of the Mind and Machine: Information Flow and Control between Humans and Computers. *Advances in Computers*, 32(1):201–254, 1991.

[Nye90] A. Nye. *Xlib Programming Manual*, volume 1 de *The X Window System Series*. O'Reilly & Associates, outubro de 1990.

[OA93a] J. L. Oliveira e R. Anido. Browsing and Querying in Object Oriented Databases. In *Proc. 2nd International Conference on Information and Knowledge Management - CIKM*, pp. 364–373, novembro de 1993.

[OA93b] J. L. Oliveira e R. Anido. Integração de uma Interface para Navegação em Diferentes Sistemas de Bancos de Dados Orientados a Objetos. In *Anais do Décimo Terceiro Congresso da Sociedade Brasileira de Computação*, pp. 61–75, setembro de 1993.

[OCM95] J. L. Oliveira, C. Q. Cunha, e G. C. Magalhães. Modelo de objetos para contrução de interfaces visuais dinâmicas. In *Anais do Nono Simpósio Brasileiro de Engenharia de Software*, pp. 143–158, outubro de 1995.

[OCM97] J. L. Oliveira, N. Cereja, e C. B. Medeiros. Modelo Intermediário de Interface para Sistemas de Informações Geográficas. In *Proc. GIS Brasil*, maio de 1997.

[Oli93] R. L. Oliveira. Transparência de Modelos de Dados em Sistemas de Bancos de Dados Heterogêneos. Master's thesis, Universidade Estadual de Campinas - Departamento de Ciência da Computação, agosto de 1993.

[Oli94] J. L. Oliveira. On the Development of User Interface Systems for Object-Oriented Databases. In *Proc. ACM Workshop on Advanced Visual Interfaces*, pp. 237–239, junho de 1994.

[OM95] J. L. Oliveira e C. B. Medeiros. A Direct Manipulation User Interface for Querying Geographic Databases. In *Proc. 2nd International Conference on Applications of Databases*, pp. 249–258, dezembro de 1995.

[OM96a] J. L. Oliveira e C. B. Medeiros. User Interface Architectures, Languages, and Models in Geographic Databases. Tutorial. In *Anais do Décimo Primeiro Simpósio Brasileiro de Banco de Dados*, pp. 20–42, outubro de 1996.

[OM96b] J. L. Oliveira e C. B. Medeiros. User Interface Issues in Geographic Information Systems. Technical Report 96-06, Institute of Computing – University of Campinas, 1996.

[OMC97] J. L. Oliveira, C. B. Medeiros, e M. A. Cilia. Active Customization of GIS User Interfaces. In *Proc. IEEE International Conference on Data Engineering*, pp. 487–496, 1997.

[Ooi90] B. Ooi. *Efficient Query Processing in Geographic Information Systems*. capítulo 2. Springer Verlag Lecture Notes in Computer Science 471, 1990.

[Ope91] Open Software Foundation. *OSF/Motif - OSF/Motif Programmer's Guide*. Prentice Hall, Inc, 1991.

[Ope96] Open GIS Consortium. *The OpenGIS Abstract Specification: An Object Model for Interoperable Geoprocessing*, 1996. Revision 1 - OpenGIS Project Document Number 96-015R1.

[OPM97] J. L. Oliveira, F. Pires, e C. B. Medeiros. An environment for modeling and design of geographic applications. *GeoInformatica*, 1(1):29–58, 1997.

[PDDJ96] N. Paton, D. Doan, O. Diaz, e A. Jaime. Exploitation of Object-Oriented and Active Constructs in Database Interface Development. In *Proc. 3rd International Conference on Interfaces to Databases*, 1996.

[Peu94] D. Peuquet. It's About Time: A Conceptual Framework for the Representation of Temporal Dynamics in Geographic Information Systems. *Annals of the Association of American Geographers*, setembro de 1994.

[PH91] T. Palanivel e M. Helander. Human-Factors Issues in Dialog Design. *Advances in Computers*, 33(1):115–171, 1991.

[Pir97] F. Pires. *Um Ambiente Computacional de Modelagem de Aplicações Ambientais*. PhD thesis, Instituto de Computação – Unicamp, dezembro de 1997.

[PMP93] N. Pissinou, K. Makki, e E. Park. Towards the Design and Development of a New Architecture for Geographic Information Systems . In *Proc. 2nd International Conference on Information and Knowledge Management - CIKM*, pp. 565–573, novembro de 1993.

[Pro96] J. Prosise. *Programming Windows 95 with MFC*. Microsoft Press, 1996.

[Pue97] A. Puerta. A Model-Based Interface Development Environment. *IEEE Software*, 14(4):40–47, 1997.

[RBP+91] J. Rumbaugh, M. Blaha, W. Premerlani, F. Eddy, e W. Lorensen. *Object-Oriented Modeling and Design*. Prentice-Hall, 1991.

[Rig95] P. Rigaux. *Interfaces Graphiques pour Bases de Données Spatiales: Application à la Représentation Multiple*. PhD thesis, CEDRIC - Conservatoire National des Arts et Metiers, 1995.

[RM92] J. Raper e D. Maguire. Design Models and Functionality in GIS. *Computers & Geosciences: An international journal*, 18(4):387–400, 1992.

[SH91] P. Svensson e Z. Huang. Geo-Sal: A query Language for Spatial Data Analysis. In *Proc. 2nd Symposium Spatial Database Systems*, pp. 119–140. Springer Verlag Lecture Notes in Computer Science 525, 1991.

[Shn87] B. Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction*. Addison-Wesley, 1987.

[SL90] A. Sheth e J. Larson. Federated database systems for managing distributed, heterogeneous, and autonomous databases. *ACM Computing Surveys*, 22(3):183–236, sep de 1990.

[SLN93] P. Szekely, P. Luo, e R. Neches. Beyond Interface Builders: Model-Based Interface Tools. In *INTERCHI'93*, pp. 383–390, April de 1993.

[Sta94] W. Staringer. Constructing Applications from Reusable Components. *IEEE Software*, 11(5):61–68, 1994.

[Sun90] Sun Microsystems. *OpenLook - Graphical User Interface Applications Style Guidelines*. Addison-Wesley, junho de 1990.

[Sur94] Surveys and Resource Mapping Branch. *Spatial Archive and Interchange Format: Formal Definition*. Ministry of Environment, Lands and Parks, British Columbia - Canadá, 1994. 3.1 edition - Reference Series.

[SVP+96] M. Scholl, A. Voisard, J-P. Peloux, L. Raynal, e P. Rigaux. *SGBD Géographiques - Spécificités*. International Thomson Publishing, 1996.

[TNB+95] R. Taylor, K. Nies, G. Bolcer, C. MacFarlane, e K. Anderson. Chiron-1: A Software Architecture for User Interface Development, Maintenance, and Run-Time Support. *ACM Transactions on Computer-Human Interaction*. 2(2):105–144, junho de 1995.

[TZ92] B. Trefz e J. Ziegler. *The Separable User Interface*, capítulo The User Interface Management System DIAMANT, pp. 241–259. Academic Press, 1992.

[Val89] J. Valdez. XVT, a Virtual Toolkit. *Byte*, 14(3), 1989.

[Vis93] Visual Edge Software. *UIM/X - Developer's Guide*. Visual Edge Software Ltd. 1993.

[Voi91] A. Voisard. Towards a Toolbox for Geographic User Interfaces. In *Proc. 2nd Symposium on Spatial Database Systems*, pp. 75–97. Springer Verlag Lecture Notes in Computer Science 525, 1991.

[Voi94] A. Voisard. Designing and Integrating User Interfaces of Geographic Database Applications. In *Proc. ACM Workshop on Advanced Visual Interfaces*, pp. 133–142, junho de 1994.

[VS94] A. Voisard e H. Schweppe. A Multilayer Approach to the Open GIS Design Problem. In *Proc. 2nd ACM Workshop on Advances in GIS*, pp. 23–29, dezembro de 1994.

[VvO92] T. Vijlbrief e P. van Oosterom. The GEO++ system: an extensible GIS. In *Proc. 5th International Symposium on Spatial Data Handling*, pp. 40–50, 1992.

[Wel88] J. Weldom. Database Interfaces. *Journal of Information Systems Management*, pp. 80–82, 1988.

[Woo93] M. Wood. *Human Factors In Geographical Information System*, capítulo Interacting with maps, pp. 111–123. Belhaven Press, 1993.

[Wor95] M. Worboys. *GIS: A Computing Perspective*. Taylor & Francis, 1995.

[ZM90] S. Zdonik e D. Maier. *Readings in Object-Oriented Database Systems*, capítulo Fundamentals of Objetc-Oriented Databases. Morgan Kaufmann, 1990.

Índice