Técnicas, Modelos e Ferramentas para Suporte à Construção de Interfaces em Sistemas de Aplicações Geográficas*

Juliano Lopes de Oliveira
juliano@inf.ufg.br
Instituto de Informática - UFG
CP 131 – Goiânia-GO – 74001-970

Claudia Bauzer Medeiros
cmbm@dcc.unicamp.br
Instituto de Computação - Unicamp
CP 6176 – Campinas-SP – 13081-970

Resumo

Este trabalho apresenta uma infra-estrutura para suporte ao projeto e implementação de interfaces geográficas. A proposta adota um enfoque pragmático e inovador, considerando dois aspectos chave da interface: a interação com o usuário e a ligação com os sistemas subjacentes. Os principais resultados do trabalho são: uma arquitetura de software para projeto e implementação de interfaces geográficas; um modelo de objetos para construção de interfaces com capacidade de incorporar modificações em tempo de execução (interfaces dinâmicas); e um mecanismo para personalização de interfaces baseado em bancos de dados ativos. Estas propostas foram utilizadas no projeto e implementação de interfaces para dois sistemas de aplicações geográficas, tendo contribuído significativamente para a redução de custos e o aumento da eficiência no desenvolvimento destas interfaces.

1 Introdução

Um *Sistema de Informação Geográfica* (SIG) é um software capaz de manipular e gerenciar dados geo-referenciados. Estes dados descrevem entidades geográficas através de atributos convencionais, espaciais e temporais. Em geral, os dados geo-referenciados são armazenados em um Sistema Gerenciador de Bancos de Dados (SGBD) Geográfico subjacente ao SIG.

Diversos tipos de aplicações geográficas podem ser desenvolvidas sobre um SIG, manipulando tanto dados urbanos (por exemplo, planejamento de tráfego e redes de telefonia) quanto ambientais (tais como, estudos de impacto ambiental). A interface de um SIG exige, além de funcionalidades comuns a interfaces em geral, o tratamento de apresentações cartográficas e mecanismos particulares de interação mais complexos que os tradicionais.

Apesar desta complexidade inerente ao tratamento de dados geo-referenciados, pouco suporte de Engenharia de Software tem sido disponibilizado para orientar a construção de aplicações geográficas. De fato, a maior parte destas aplicações é desenvolvida de maneira *ad hoc*, com base em ferramentas genéricas e diretivas empíricas. Esta realidade pode ser melhorada com o emprego de técnicas adequadas que permitam sistematizar o desenvolvimento de sistemas de aplicações geográficas. Este trabalho apresenta um conjunto de propostas visando atingir esta abordagem sistemática, sendo a base das propostas uma arquitetura de software voltada para o domínio de aplicações geográficas.

*Trabalho realizado com suporte financeiro parcial do CNPq

Vale salientar que uma arquitetura de software não se limita a uma descrição de módulos interligados em um diagrama. Ao contrário, ela deve definir todos os pontos fundamentais para construção de um software, incluindo abstrações que orientem o projetista a lidar com a complexidade do software. A arquitetura deve também contemplar os elementos de composição do software, bem como as maneiras pelas quais estes elementos interagem entre si, os padrões que guiam a composição dos elementos, as restrições previstas sobre estes padrões, e a especificação completa do comportamento de *run-time* do software a ser construído [17, 1].

A abordagem aqui proposta para o projeto e implementação de interfaces de usuário para SIG compreende um arcabouço teórico – mecanismos, modelos, técnicas e ferramentas – organizado em uma arquitetura de software de domínio específico (*domain specific software architecture*). Esta arquitetura foi validada experimentalmente, e atende aos requisitos de desenvolvimento das aplicações geográficas. O enfoque inovador desta proposta está baseado em dois princípios:

- ◇ O projeto e arquitetura de uma interface para SIG devem ser realizados levando em consideração a presença do software geográfico subjacente, separando claramente as funções que precisam ser deixadas a cargo da interface das demais. O enfoque tradicional, ao contrário, considera a interface geográfica isoladamente do restante do sistema, inclusive duplicando funções de SGBDs na interface.

- ◇ Interfaces para SIG não servem apenas para consultas *ad hoc*, devendo também permitir acoplamento de outras aplicações. Os trabalhos existentes, em contraste, consideram apenas interfaces de consulta sobre dados geo-referenciados.

Dadas essas premissas, as principais contribuições deste trabalho são: (i) a especificação de uma arquitetura de software para interfaces de SIG; (ii) a definição de uma hierarquia de classes para construção de interfaces geográficas, permitindo incorporar modificações em tempo de execução (interfaces dinâmicas); (iii) a definição de um mecanismo para adaptação de interfaces aos usuários baseado em bancos de dados ativos; e (iv) a criação de componentes de interface voltados para o domínio de aplicações geográficas.

O restante deste texto introduz os principais aspectos desta nova infra-estrutura. A próxima seção mostra as principais características da arquitetura de software proposta para guiar o desenvolvimento de interfaces geográficas. A seção 3 descreve um modelo de dados voltado para o projeto e implementação das interfaces especificadas nessa arquitetura. A seção 4 apresenta uma abordagem para personalização dessas interfaces geográficas. A seção 5 resume as principais idéias e conclusões do trabalho.

2 Arquitetura para Interfaces Geográficas

As arquiteturas de software de interface estabelecem como princípio fundamental de projeto a separação entre os componentes interativo e semântico [6]. No entanto, estas arquiteturas apresentam diversas limitações. As *arquiteturas de interface genérica* (tais como as descritas em [2] e [20]) são inadequadas para interfaces geográficas porque: (1) não levam em consideração as idiossincrasias das aplicações geográficas; (2) não

consideram a necessidade de integração com diferentes tipos de software de suporte (somente *toolkits* de interface); e (3) não facilitam a transição da fase de projeto para a fase de implementação.

As *arquiteturas de interface geográfica* ([14], [21], e [16], por exemplo) também apresentam soluções limitadas. Algumas destas arquiteturas são direcionadas para um determinado tipo de SIG e propõem soluções *ad hoc*. Outras são independentes de SIG, mas impõem restrições funcionais sobre o SIG ou sobre o Componente Interativo da aplicação geográfica [10].

A arquitetura proposta neste trabalho supera essas limitações. Como nas arquiteturas tradicionais, o software de interface é dividido em um conjunto de componentes abstratos inter-relacionados. No entanto, ao contrário de suas predecessoras, esta nova arquitetura torna explícito o processo de refinamento destes componentes, até atingir o nível de implementação.

A arquitetura de interface geográfica define um *modelo de dados intermediário* para representar os conceitos espaciais usados pelo usuário, mapeando-os para aqueles efetivamente implementados no SIG. Este modelo está baseado no GMOD, um modelo de dados geográfico orientado a objetos [12]. O uso do GMOD como modelo intermediário facilita a transição do modelo mental do usuário (visão geográfica do mundo) para a implementação no SGBD geográfico subjacente ao SIG.

2.1 Componentes da Arquitetura

A arquitetura de software de interface define três componentes principais, organizados em camadas sobrepostas, conforme mostra a figura 1. Cada camada gerencia um conjunto bem definido de tarefas: comunicação com os sistemas de suporte subjacentes (Camada de Ligação); gerenciamento de modelos de dados da 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).

A **Camada de Ligação** é responsável pela comunicação com os sistemas de suporte da aplicação geográfica: o SIG e o *toolkit* de interface. Ela provê um *módulo adaptador* que oferece às camadas superiores da arquitetura um acesso normalizado e independente do sistema de suporte adotado. Cada módulo adaptador define uma máquina abstrata com uma interface de programação uniforme, de modo que as camadas superiores utilizam a mesma interface de comunicação para qualquer sistema subjacente.

O Adaptador de SIG faz a tradução de esquemas, dados e operações do modelo intermediário da arquitetura (GMOD) para o modelo de dados do SIG subjacente, e vice-versa. Para isto o adaptador mantém um *modelo de mapeamento* que define a correspondência entre conceitos dos dois modelos. O Adaptador oferece para o restante da arquitetura uma API e se encarrega de traduzir este protocolo para o mecanismo de comunicação oferecido pelo SIG.

O Adaptador de *Toolkit* permite definir objetos de interface de acordo com a hierarquia de classes de objetos de interface IMOD (definida na próxima seção). Embora cada *toolkit* possua representações diferentes para objetos de interação, existe um conjunto básico desses objetos que é oferecido pela maioria dos *toolkits*. O Adaptador de *Toolkit* define estes tipos fundamentais e implementa regras de mapeamento para *widgets* do *toolkit* adotado.

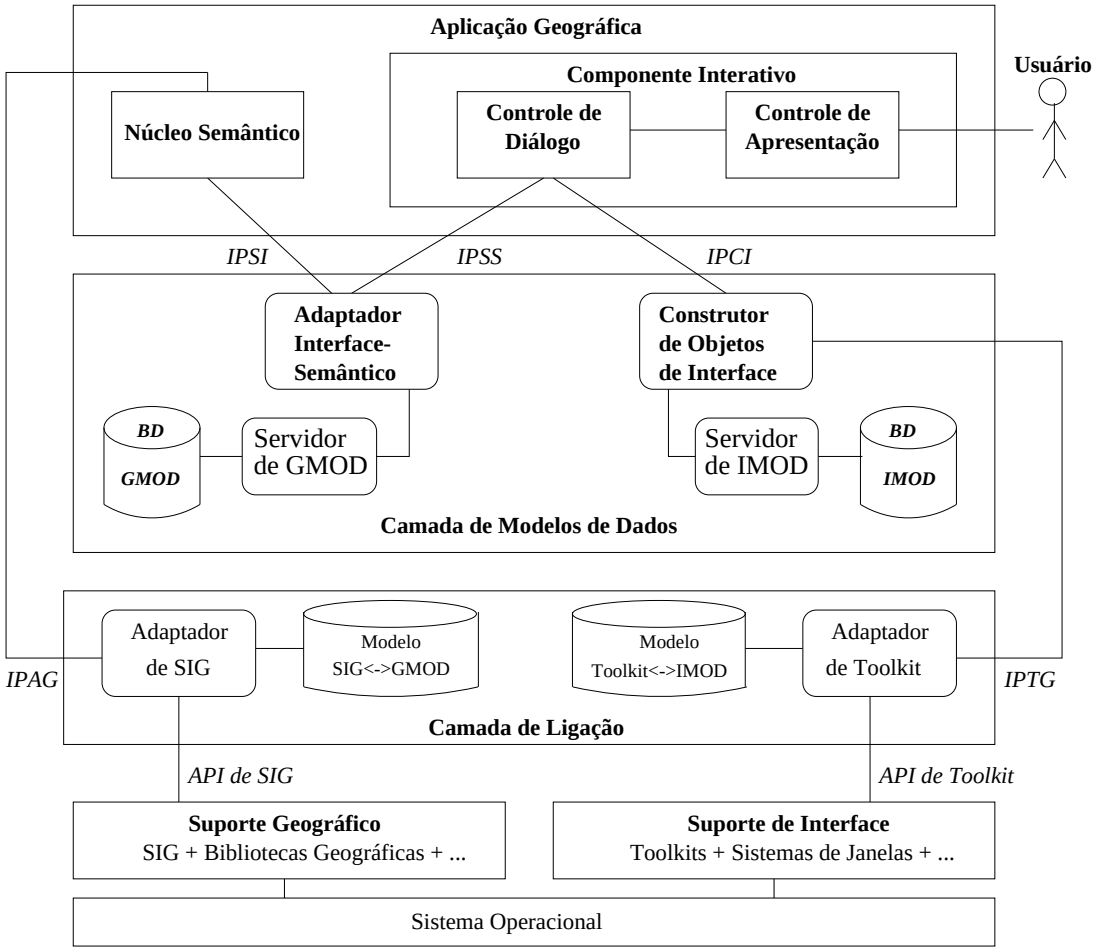

Figura 1: Arquitetura de software de interface geográfica

A **Camada de Modelos de Dados** mantém dois bancos de dados orientados a objetos: um para os dados manipulados pela aplicação (BD GMOD); outro para os objetos que definem como os primeiros irão aparecer na interface (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.

A hierarquia de classes IMOD oferece facilidades para criar diferentes apresentações dos dados contidos no BD GMOD. Estas classes representam os objetos que são usados no diálogo com o usuário. O módulo *Construtor de Objetos de Interface* permite compor objetos de interface complexos a partir de objetos mais simples recuperados do BD IMOD.

A **Camada de Aplicação** define o software da aplicação geográfica que é construída sobre o SIG. Ela é composta de um Componente Interativo (o que o usuário vê e com o qual interage) e o Núcleo Semântico (que encapsula o código da aplicação geográfica). Tanto o Núcleo Semântico quanto o Componente Interativo podem ter acesso aos dados gerenciados pelo servidor do BD GMOD através de um *Adaptador Interface–Semântico*. Este adaptador define os principais aspectos da interação entre os componentes (nível

de abstração dos dados transferidos, componente proprietário de cada dado, mecanismo de descrição de dados intercambiados, e tipo de acesso permitido).

O módulo de *Controle de Apresentação* da Camada de Aplicação realiza duas tarefas fundamentais: (1) a transformação de ações (eventos) do 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, o módulo de Controle de Diálogo gerencia os relacionamentos entre objetos de interação e objetos definidos na aplicação.

Vale notar que o Núcleo Semântico não é um módulo da interface, assim como não o é o SIG subjacente. A arquitetura apenas define os protocolos de comunicação entre Núcleo Semântico e o Componente Interativo.

2.2 Interoperabilidade

As linhas rotuladas (IPAG, IPSI, IPSS, IPCI, IPTG) apresentadas na figura 1 representam canais de comunicação entre camadas distintas, que definem 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. As diferentes camadas da arquitetura se comunicam somente via API. Isto contribui para o encapsulamento dos módulos, facilitando o desenvolvimento, a manutenção, a evolução, e a reutilização do software.

O Adaptador Interface–Semântico encapsula a interação entre o Componente Interativo e o Núcleo Semântico, garantindo a *independência de diálogo*. O Adaptador implementa um mecanismo de propagação de mudanças efetuadas sobre os dados compartilhados, permitindo a implementação de *diálogo concorrente*. O exemplo a seguir ilustra estas idéias. Considere uma aplicação geográfica que permite atualizar informações espaciais através de um mapa interativo. Considere, ainda, que todos os objetos apresentados são resultado da aplicação de uma apresentação a objetos previamente selecionados, extraídos do SGBD geográfico, e que já se encontram no BD GMOD.

- O Controle de Apresentação comunica ao Controle de Diálogo as ações do usuário sobre objetos de interface. Por exemplo, aquele módulo pode converter um evento de *"double click"* em uma operação de seleção de um objeto geográfico $\mathcal{O}$ do BD GMOD.

- O Controle de Diálogo gerencia a correspondência entre objetos de interface (BD IMOD) e objetos da aplicação (BD GMOD). O módulo mantém ainda um *contexto de interação* da interface que permite a identificação da operação a ser realizada em resposta à ação do usuário. Neste exemplo, o Controle de Diálogo ativa as seguintes operações:

  (1) Uma consulta sobre o conteúdo de $\mathcal{O}$ (via IPSS). O Adaptador Interface–Semântico pode executar diretamente esta operação, pois os dados já se encontram no BD GMOD.

(2) Uma solicitação de apresentação de $\mathcal{O}$ (via IPCI). O Construtor recebe do Controle de Diálogo uma descrição do tipo de apresentação e do objeto a ser apresentado, buscando no BD IMOD um objeto de interface apropriado $\mathcal{I}$. Este último é enviado ao Adaptador de *Toolkit* (via IPTG), que traduz a descrição contida em $\mathcal{I}$ para a API do *toolkit* subjacente, criando os *widgets* correspondentes.

(3) Uma mudança no contexto de interação para refletir a presença da nova janela. Em cada contexto, diferentes funções de interface podem estar disponíveis, e diferentes objetos de interação podem estar ativos.

- O usuário realiza as operações desejadas sobre $\mathcal{O}$ (apresentado de acordo com a especificação em $\mathcal{I}$). Novamente, os eventos de usuário são traduzidos pelo Controle de Diálogo, gerando uma operação de alteração do BD GMOD junto ao Adaptador Interface–Semântico.

- O Adaptador Interface–Semântico faz a validação semântica da operação, de acordo com as restrições expressas no BD GMOD, podendo solicitar validação semântica adicional ao Núcleo Semântico. Se nenhum erro é detectado, o Adaptador atualiza o estado de $\mathcal{O}$ no BD GMOD, sinalizando a mudança ocorrida para o Núcleo Semântico (via IPSI).

Esta arquitetura em camadas promove a modularização e a especialização dos componentes da interface, simplificando o desenvolvimento de aplicações geográficas sobre o SIG, e estabelecendo uma divisão clara de responsabilidades. A interface descrita em [9], desenvolvida com base nos princípios desta arquitetura, é um exemplo de software com tais características. Além disso, a arquitetura garante para a camada superior a propriedade de independência de dados e possibilita a reutilização das camadas de suporte das aplicações geográficas. Esta mesma idéia de reutilização baseada em arquiteturas de domínio específico e no uso de técnicas de orientação a objetos pode ser observada em [3].

3 Modelo para Construção de Interface

Esta seção propõe uma abordagem para desenvolvimento do Componente Interativo, que é o mais complexo em qualquer arquitetura de interface geográfica. Os principais objetivos desta abordagem são: (1) dar suporte a *interfaces dinâmicas*, permitindo a modificação de apresentação e comportamento da interface em tempo de execução; (2) automatizar o processo de construção do Componente Interativo, possibilitando a reutilização de elementos de interface; e (3) garantir a separação e independência entre os componentes semântico e interativo da aplicação, mantendo um mapeamento simples do projeto para a implementação.

Para atender estes objetivos, o Componente Interativo é organizado em *Objetos de Interface* compostos por *Objetos de Interação*. 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 funcionalidade

lógica prevista para os módulos de Controle de Diálogo e de Controle de Apresentação da arquitetura de interface.

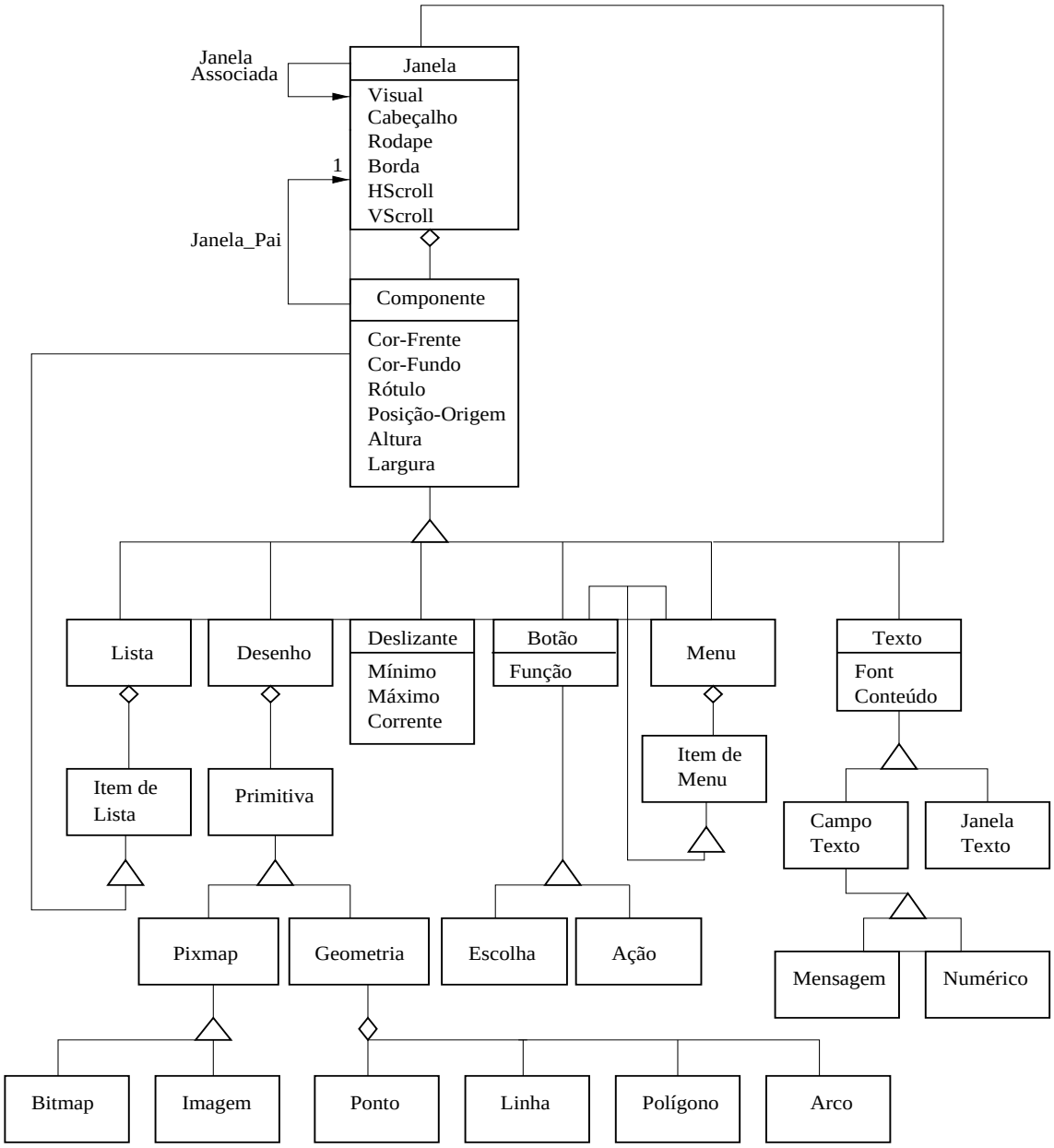

Figura 2: Núcleo do modelo de objetos de interface IMOD

O princípio fundamental do processo de construção do Componente Interativo é a hierarquia de classes orientadas a objetos IMOD, que permite definir objetos de interface de maneira abstrata. A idéia é equivalente ao conceito de *template*, ou seja, uma estrutura que dirige a construção de cada objeto da interface. Na literatura relativa a projeto de interfaces, estes *templates* são frequentemente denominados *modelos de objetos de interface*, e formam a base de uma das principais abordagens para geração automática de interfaces – as chamadas ferramentas de interface baseadas em modelos [18, 15].

A hierarquia de classes de interface é armazenada no BD IMOD, sendo que cada classe

contém objetos que definem a apresentação (estado) e o controle (comportamento) de um aspecto da interface. A especificação de objetos de interface é auxiliada pelo *Construtor de Objetos de Interface*, que gera, automaticamente, objetos de interface para os objetos do BD GMOD.

A construção de uma interface se divide em projeto e implementação. A **fase de projeto** especifica os *templates* (modelos de objetos) para os elementos de diálogo da interface, armazenando-os no BD IMOD, que pode ser considerado uma *Biblioteca de Objetos de Interface*. Os objetos desta biblioteca podem ser reutilizados e especializados em diferentes aplicações. 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** abrange a instanciação de objetos da biblioteca, ligando-as com as instâncias do BD GMOD. O Construtor é responsável por criar, junto ao Adaptador de *Toolkit*, os *widgets* envolvidos. Esta fase é a que efetivamente associa as instâncias de dados $\mathcal{O}$ e de interface $\mathcal{I}$ (vide seção anterior).

3.1 O Modelo Dinâmico de Objetos de Interface

O IMOD – *Interface Data Model* – é uma extensão voltada para interfaces geográficas do modelo de objetos de interface proposto em [7]. O seu núcleo é formado por um conjunto de classes comuns aos *toolkits* de interface, associadas a primitivas para construção de mapas e apresentações cartográficas em geral (figura 2). O IMOD facilita o projeto de *interfaces visuais*, permitindo a coexistência de diversos estilos de interação (linguagens visuais, menus, e manipulação direta).

O elemento básico de composição no IMOD é a classe Janela. 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, ou pode servir apenas para organizar logicamente os componentes mostrados na interface. Exemplos dos atributos de uma janela são bordas, cabeçalho, rodapé e barras de rolamento.

A classe abstrata Componente é a base para especificação dos diversos tipos de elementos de diálogo utilizados em interfaces visuais. A classe Componente define ainda propriedades genéricas de apresentação que são herdadas pelos componentes especializados. As propriedades incluem cores, rótulos, posição do componente na janela que o contém, e dimensões do componente.

Um componente é apresentado na sua `janela_pai`, ocupando um retângulo de dimensões e posição fixas. O método `muda_local` modifica a posição do retângulo do componente, enquanto `muda_forma` redefine as dimensões do retângulo. 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 nos *toolkits* de interface atuais [5]. De fato, estas subclasses utilizam as funções oferecidas pelos *widgets* correspondentes (atributo `widgets`), mas não se limitam a estas funções. A subclasse Lista ilustra este fato: uma instância de Lista é formada por componentes de interface arbitrários, os quais possuem, como foi visto anteriormente, métodos que permitem o ajuste de seu leiaute. Assim, o projetista de interface pode organizar os itens de uma

instância de Lista de diferentes maneiras, ao contrário do que acontece em *toolkits* de interface.

3.2 Uso do IMOD em Aplicações Geográficas

O IMOD permite a construção automática de apresentações de objetos geográficos complexos definidos no BD GMOD, de acordo com a seguinte taxonomia para apresentação em interfaces visuais:

1. Atributos *simples*: 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.

   (b) Textos: são apresentados através de janelas específicas que provêm facilidades padronizadas para visualização e edição de textos.

   (c) Imagens: assim como textos, imagens são apresentadas em janelas dedicadas, e geralmente suportadas diretamente pelos *toolkits*.

   (d) Atributos multimídia: podem ser considerados *simples* (de acordo com esta taxonomia), porque são apresentados por um componente (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. Pode haver três tipos de atributos complexos:

   (a) Atributos espaciais: são apresentados através de um objeto de interface composto, a *Janela de Mapa*.

   (b) Atributos construtores: dividem-se em homogêneos (*list* e *set*) e heterogêneos (*tuple*). São apresentados como objetos de interface compostos contendo os objetos de interface que representam os diversos atributos agrupados pelo construtor.

   (c) Sub-objetos: um sub-objeto de um *objeto complexo* é apresentado em uma janela composta de acordo com o tipo de sua classe.

O Construtor de Objetos de Interface é uma ferramenta que constrói projetos de interface a partir das definições do GMOD, utilizando esta taxonomia para apresentação de atributos. Uma das dificuldades neste processo é a integração de dados convencionais e geográficos, já que uma interface geográfica inclui elementos *cartográficos* (organizados, via de regra, em mapas) e também elementos de *controle* (envolvendo dados convencionais e operações da aplicação representados por objetos do IMOD). As ferramentas atuais de interface não têm mecanismos genéricos para descrever os relacionamentos entre objetos gráficos e de controle de uma aplicação geográfica.

Para oferecer facilidades de manipulação de mapas, três classes 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.

Ao contrário das demais propostas para interfaces geográficas ([21] e [16], por exemplo) esta proposta tem três aspectos fundamentais: (1) o modelo serve para construir interfaces genéricas, e não apenas uma interface de consulta para SIG; (2) o modelo intermediário de interface suporta tanto a visão de objetos (vetorial) quanto a de campos (matricial); e (3) o modelo prevê classes para manipular de maneira integrada os componentes convencionais e espaciais da informação geográfica.

4 Personalização Ativa de Interfaces

A arquitetura descrita na seção anterior é genérica. No entanto, as idiossincrasias das aplicações e dos usuários de SIG são dificilmente satisfeitas por um mecanismo de interface único. Uma abordagem para este problema é acoplar mecanismos à interface para facilitar o processo de personalização. Em [4, 13], por exemplo, tecnologias de SGBD são incorporadas à interface e utilizadas para resolver diferentes tipos de problemas de interação com o usuário. No entanto, 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.

A abordagem proposta aqui também utiliza funções existentes em SGBD mas, ao invés de duplicar a funcionalidade do SGBD na interface, estende o SGBD subjacente com mecanismos para construção de interfaces personalizáveis. Os módulos adicionados aproveitam facilidades já previstas em SGBD, e não comprometem o desempenho da aplicação.

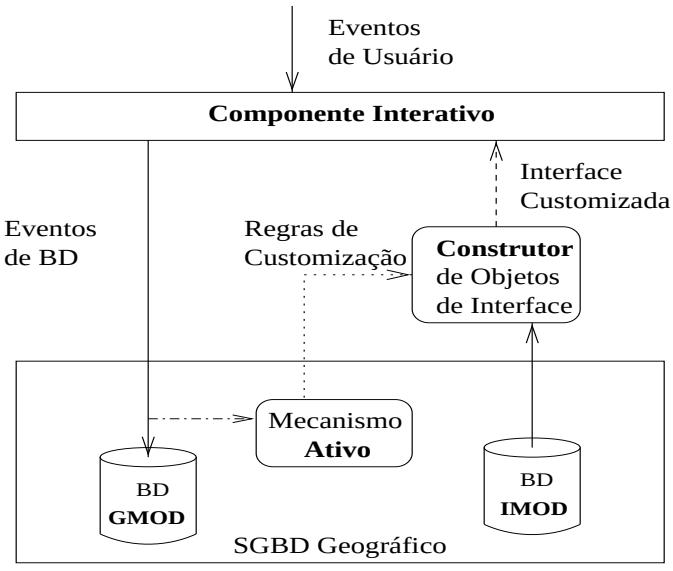

Figura 3: Arquitetura e funcionalidade do mecanismo de personalização

O mecanismo de personalização se fundamenta na integração de três componentes [11]: uma *Interface Geográfica Genérica*, implementada de acordo com a arquitetura

já apresentada; (b) um *Construtor de Objetos de Interface*, associado a uma *Biblioteca de Objetos de Interface* (BD IMOD); e (c) um *Mecanismo de Bancos de Dados Ativo*, implementado no SGBD subjacente, que controle o acesso aos dados armazenados no SIG (figura 3).

A construção de objetos de interface, como visto, é baseada em associar $\mathcal{O}$ do BD GMOD a $\mathcal{I}$ do BD imod. A personalização intervém nesta associação, modificando-a em tempo de execução.

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 do SIG subjacente. Estes eventos são interceptados pelo Mecanismo Ativo, que dispara *regras de personalização* apropriadas. Estas regras permitem que o programador redefina o *look-and-feel* da interface, atuando sobre objetos da Biblioteca (BD IMOD).

Regras e objetos de interface são processados pelo Construtor, gerando uma *definição de interface personalizada*. Esta definição representa um modelo de objetos de interface, que é utilizado para construir dinamicamente os objetos do Componente Interativo.

O objetivo do Mecanismo Ativo no contexto deste trabalho é personalizar as definições armazenadas no BD IMOD. Da mesma forma que em mecanismos tradicionais ativos em SGBD, **E**ventos são solicitações do usuário, através da interface, para acesso aos dados (geográficos). A maior diferença está nos componentes de **C**ondição e **A**ção das regras. Enquanto em SGBD ativos estas manipulam dados do banco de dados, aqui elas manipulam objetos da interface, sendo especificadas como:

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 do BD IMOD $IO_1, ..., IO_k$

A **A**ção – personalização CT – é um código para personalização de interface, escrito na linguagem de personalização apresentada em [8]. Esta personalização combina dois tipos de objetos: aqueles resultantes do pedido do usuário ao BD geográfico ($O_1, ..., O_e$) e aqueles recuperados do BD IMOD ($IO_1, ..., IO_k$). O processo de personalização 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 na verdade composto de duas partes: um *evento de acesso ao* BD GMOD e uma *ação de interface*. As informações recuperadas do BD são utilizadas pelo Construtor para definir o modelo de interface correspondente. Caso não haja uma regra de personalização definida para o evento, adota-se um procedimento de construção *default*, baseado na taxonomia de atributos da seção anterior. Caso contrário, a personalização é executada a partir da regra definida.

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 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 estendida para outros dados contextuais (por exemplo,

escala geográfica e estrutura temporal).

Vale notar que a ação de uma regra está limitada à personalização de um objeto de interface, de forma que nenhuma regra conflitante pode ser gerada. Além disso, a política de execução de regras determina que apenas a regra de mais alta prioridade é executada em cada evento. 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.

A solução apresentada é superior às outras abordagens de personalização 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. Em [11, 8] são apresentados diversos exemplos do uso do mecanismo aqui descrito para personalização de interfaces geográficas.

5 Conclusões

As técnicas, modelos e ferramentas propostas neste trabalho contribuem para a resolução de problemas computacionais existentes na construção de interfaces para aplicações geográficas. As soluções propostas utilizam teorias de três áreas da computação: Interfaces Usuário–Computador, Bancos de Dados e Engenharia de Software.

Estas teorias são consideradas dentro de um enfoque inovador, envolvendo aspectos que não são tratados pelos trabalhos anteriores. Primeiro, a interface não é considerada isoladamente, e tampouco é vista como o componente mais importante em um sistema geográfico (ao contrário do enfoque usual encontrado na pesquisa em interfaces). Ela faz parte de um ambiente heterogêneo, onde diferentes softwares trabalham em cooperação.

Em segundo lugar, a proposta dá suporte ao projeto e implementação de interfaces para aplicações geográficas genéricas, e não apenas para interfaces de consulta. Praticamente todas as propostas de sistemas de interface para SIG consideram apenas consultas, justamente por serem projetados exclusivamente sob a perspectiva de interface, sem levar em consideração as particularidades do software geográfico subjacente.

Dentro deste contexto, este trabalho apresentou uma integração de diversas propostas para definir uma abordagem sistemática para os problemas de concepção e implementação de interfaces geográficas. Os principais resultados obtidos foram:

- A especificação de uma *arquitetura de software* voltada para o domínio específico de interfaces geográficas. A arquitetura organiza os componentes do sistema em camadas, definindo suas funcionalidades, interoperabilidade, e ligação com os softwares de suporte.

- ◇ Um *modelo de objetos de interface* para construção de *interfaces geográficas dinâmicas*. O modelo permite definir e modificar, em tempo de execução, os componentes de interação da arquitetura.

- ◇ Um *modelo de dados intermediário* para interfaces geográficas. Este modelo permite o mapeamento entre aspectos conceituais e físicos, utilizando 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. O mecanismo permite adaptar uma interface geográfica de acordo com as necessidades específicas de um usuário (ou grupo), levando em conta os requisitos de cada aplicação.

Estas propostas foram validadas no projeto e implementação de interfaces para aplicações geográficas, nas áreas de infraestrutura urbana [7] e de planejamento e controle ambiental [12]. As propostas também foram utilizadas com sucesso na construção da interface de um ambiente computacional para auxílio ao projeto de bancos de dados geográficos [19].

Referências

[1] L. Bass, P. Clements, R. Kazman. Software Architecture in Practice. Addison-Wesley, Series in Software Engeneering, 1998.

[2] L. Bass, R. Faneuf, R. Little, et al. A Metamodel for the Runtime Architecture of an Interactive System. *SIGCHI Bulletin*, 24(1):32–37, Jan. 1992.

[3] A. Cima, C. Werner, A. Cerqueira. The Design of Object-Oriented Software with Domain Architecture Reuse In *IEEE International Conference on Software Reuse*, nov. 1994.

[4] N. Gopal, C. Hoch, R. Krishnamurthy, et al. Is GUI Programming a Database Research Problem? In *ACM SIGMOD Conference*, 517–528, 1996.

[5] B. Myers, R. McDaniel, et al. The Amulet Environment: New Models for Effective Interface Software Development. *Trans. Soft. Eng.*, 23(6):347–365, 1997.

[6] B. Myers. User Interface Software Tools. *ACM Transactions on Computer-Human Interaction*, 2(1):64–103, Mar. 1995.

[7] J. L. Oliveira, C. Q. Cunha, G. C. Magalhães. Modelo de objetos para contrução de interfaces visuais dinâmicas. In *IX SBES*, 143–158, 1995.

[8] J. L. Oliveira. *Projeto e Implementação de Interfaces para Sistemas de Aplicações Geográficas*. PhD thesis, IC – Unicamp, Dez. 1997.

[9] J. L. Oliveira, C. B. Medeiros. A Direct Manipulation User Interface for Querying Geographic Databases. In *Int. ADB Conference*, 249–258, 1995.

[10] J. L. Oliveira, C. B. Medeiros. Tutorial: User Interface Architectures, Languages, and Models in Geographic Databases. In *XI SBBD*, 20–42, 1996.

[11] J. L. Oliveira, C. B. Medeiros, M. A. Cilia. Active Customization of GIS User Interfaces. In *IEEE ICDE Conference*, 487–496, 1997.

[12] J. L. Oliveira, F. Pires, C. B. Medeiros. An environment for modeling and design of geographic applications. *GeoInformatica*, 1(1):29–58, 1997.

[13] N. Paton, D. Doan, O. Diaz, A. Jaime. Exploitation of Object-Oriented and Active Constructs in Database Interface Development. In *III International Conference on Interfaces to Databases*, 1996.

[14] N. Pissinou, K. Makki, E. Park. Towards the Design and Development of a New Architecture for GIS . In *II CIKM*, 565–573, 1993.

[15] A. Puerta. A Model-Based Interface Development Environment. In *IEEE Software*, 14(4):40–47, 1997.

[16] P. Rigaux. *Interfaces Graphiques pour Bases de Données Spatiales: Application à la Représentation Multiple*. PhD thesis, CEDRIC - CNAM, 1995.

[17] M. Shaw, D. Garlan. Software Architecture - Perspectives on an Emerging Discipline. Prentice-Hall, 1996.

[18] P. Szekely, P. Luo, R. Neches. Beyond Interface Builders: Model-Based Interface Tools. In *INTERCHI*, 383–390, 1993.

[19] M. Salles, F. Pires, C. B. Medeiros, J. L. Oliveira. Development of a Computer Aided Geographic Database Design System. In *XIII SBBD*, 1998.

[20] R. Taylor, K. Nies, G. Bolcer, et al. Chiron-1: A Software Architecture for User Interface Development, Maintenance, and Run-Time Support. *Trans. Computer-Human Inter.*, 2(2):105–144, 1995.

[21] A. Voisard. Designing and Integrating User Interfaces of Geographic Database Applications. In *ACM AVI Workshop*, 133–142, 1994.