Modelo de Objetos para Construção de Interfaces Visuais Dinâmicas

Juliano Lopes de Oliveira * Cleida Queiroz Cunha †
juliano@dcc.unicamp.br cleida@cpqd.br

Geovane Cayres Magalhães ‡
geovane@cpqd.br

Sumário

Apesar de todos os avanços na área de interface usuário-computador, a camada de diálogo com o usuário é ainda hoje responsável por uma parte considerável dos custos de projeto, desenvolvimento, teste e manutenção de sistemas de informação. Neste trabalho apresentamos um mecanismo que permite a reutilização de esforços na elaboração de sistemas, em termos de projeto e de implementação, visando a redução dos custos associados à camada de interface com o usuário. Nossa abordagem utiliza um modelo de interface orientado a objetos para descrever uma biblioteca de elementos que facilita a construção de interfaces. Uma das principais vantagens dessa abordagem é que ela torna viável a reutilização na construção de interfaces dinâmicas, isto é, interfaces cujos componentes são determinados em tempo de execução do sistema.

Palavras-Chave: Interface Visual; Modelo Orientado a Objetos; Reutilização; Construção Dinâmica.

Abstract

In spite of all advances in the area of human-computer interface, the user-dialog layer is still responsible for a reasonable amount of costs in design, development, test and maintenance of information systems. In this work we present a mechanism that allows the reuse of efforts on the elaboration of systems, in terms of design and implementation, towards the reduction of costs associated to the user interface component. Our approach uses an object-oriented interface model to describe a components library that improves the development of user interfaces. One of the main advantages of this approach is to make possible the reuse on the development of dynamic user interfaces, that is, user interfaces whose components are specified in run time.

Keywords: Visual Interface; Object-Oriented Data Model; Reuse; Dynamic Construction.

*Doutorando em Ciência da Computação no DCC - IMECC - UNICAMP: CP 6065, 13081-970, Campinas-SP
†Pesquisadora do CPqD (Telebrás):Rod.Campinas-Mogi Mirim Km 118,5 CP 1579, 13088-610, Campinas-SP
‡Pesquisador do CPqD (Telebrás); Professor do Departamento de Ciência da Computação da UNICAMP

1 Introdução

Um dos fatores predominantes para que um sistema de informações seja bem sucedido é a facilidade de acesso às funções que o software oferece aos seus usuários finais. Em muitos casos, a escolha do produto a ser adquirido é feita com base não no seu poder computacional, mas na sua capacidade de ser facilmente entendido e utilizado.

A constatação desta realidade tem motivado pesquisas intensas em todas as áreas relacionadas ao processo de interação entre usuário e computador. Resultados relevantes têm sido obtidos na modelagem do processo cognitivo do usuário [PH91, Nor91, Mar94], no desenvolvimento de padrões e diretivas para projeto de interfaces [Shn87, Sun90, Ope91, Oli94], e na produção de ambientes e ferramentas para geração automática de sistemas de interface [HH89, Mra91, Mam91, PSI93].

1.1 Motivação e Objetivos

Embora exista um progresso acentuado em diversas áreas, a interação entre um sistema computacional e seus usuários continua sendo uma grande preocupação em novos projetos, notadamente em aplicações que envolvem a utilização intensiva e interativa do software pelos usuários. Neste tipo de aplicação, o total de linhas de código destinado à definição e controle do diálogo com o usuário pode chegar a mais de cinqüenta por cento do código total do sistema [MvD91, MR92].

O objetivo deste trabalho é propor um mecanismo para redução do custo efetivo da camada de interface com o usuário. Com essa finalidade, introduzimos um modelo extensível de elementos de interface que pode ser implementado em uma biblioteca de classes, resultando em componentes reutilizáveis. A biblioteca provê a estrutura de suporte para reutilização de código, enquanto o modelo serve de base para a reutilização de projeto na construção de novos sistemas de interface.

Ambos o modelo de interface e a biblioteca de classes seguem 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 usuário-computador. Os custos tendem a ser menores à medida que cresce o número de elementos especializados de interface contidos na biblioteca. Além de reduzir os custos de novos projetos, a abordagem aqui proposta possibilita uma notável melhoria nos aspectos relacionados à manutenção e evolução de sistemas de interface.

1.2 Ferramentas para Construção de Interfaces

Já que a idéia principal deste trabalho é facilitar a construção de interfaces visuais, é possível imaginar que se trata de mais um *toolkit*[1] para interfaces ou um novo UIMS[2]. No entanto, o tipo de abordagem que estamos propondo é ortogonal às destas ferramentas, situando-se em um nível mais abstrato na arquitetura de um ambiente de desenvolvimento de software.

[1] Conjunto de ferramentas para desenvolvimento de software

[2] Sistema gerenciador de interface com usuário (*User Interface Management System*)

Em geral toolkits e UIMSs 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, a construção de interfaces é feita de maneira *ad hoc*, sem um suporte para estruturação do projeto da interface.

Teleuse [Tel91], Uim/X [Vis93] e Xview [Hel90] são exemplos de ferramentas que permitem reutilização de código para criação e gerência de objetos elementares de interface (*widgets*). Nossa proposta permite este mesmo tipo de reutilização estendendo-o para elementos complexos de interface (composições de widgets). Além disso, nosso modelo genérico de interface serve de guia para a reutilização efetiva de projetos de interface.

Nossa proposta pode utilizar os recursos oferecidos por estas ferramentas 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 segundo nossa proposta segue um modelo construtivo único, combinando objetos em diferentes níveis de complexidade. Os UIMSs e toolkits provêem, via de regra, apenas objetos de interação simples, ficando para o programador a carga de gerenciar elementos de interface inter-relacionados.

Além de toolkits e UIMSs, existe um terceiro tipo de ferramenta que é essencial para a construção de sistemas de interface com o usuário. Trata-se dos sistemas para especificação e controle do diálogo. Diversos formalismos e técnicas tem sido apresentados com esta finalidade, como por exemplo [HLN+90], [CHB92] e [LL94]. Nosso modelo de interface faz uso de uma ferramenta subjacente para a definição do diálogo.

Apesar de nossa proposta permitir o uso de qualquer ferramenta que permita a especificação do diálogo, utilizamos o modelo dinâmico da metodologia OMT [RBP+91]. O poder de expressão e a simplicidade deste modelo são bastante adequados para nossos objetivos, mas a principal vantagem é que o modelo faz parte de uma metodologia de desenvolvimento orientada a objetos, de modo que os mesmos conceitos podem ser empregados desde o projeto até a implementação da interface.

O restante deste trabalho descreve o modelo de interface proposto, e está organizado da seguinte maneira: a próxima seção apresenta os conceitos básicos do modelo de interface, introduzindo as classes e relacionamentos que formam o seu núcleo; na seção 3 mostramos uma possível implementação do modelo através de uma biblioteca de classes para uma plataforma específica; a seção 4 discute a reutilização de projeto e de código baseada no modelo e na biblioteca propostos; a seção 5 apresenta conclusões relativas à experiência de utilização dos mecanismos aqui propostos em projetos reais de grande porte.

2 O Modelo de Objetos de Interface

A principal motivação para a construção deste modelo de interface é promover a reutilização não só de código, mas também de projetos de interface. É preciso lembrar, todavia, que a abrangência desta proposta se restringe ao domínio específico de interfaces com o usuário. Esta restrição de domínio para a construção de componentes reutilizáveis é indicada na literatura como um dos pontos chave para o sucesso de tecnologias de reutilização [SM94, BD94, Sta94].

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

A essência do modelo de interface proposto se baseia no paradigma de orientação a objetos. Uma boa razão para essa abordagem vem da constatação de que o emprego de conceitos orientados a objetos em todas as fases do ciclo de desenvolvimento reforça as chances de reutilização [CWC94]. Assim, o modelo é formado por um conjunto de classes inter-relacionadas, onde cada classe descreve um componente típico de um *sistema de interface visual*. Uma interface visual suporta e permite a coexistência de qualquer um dos estilos básicos de interação (linguagens, menus, telas formatadas, e manipulação direta). Portanto, esta classe de interfaces é mais abrangente que a das interfaces gráficas, as quais utilizam primordialmente conceitos de manipulação direta [Shn83].

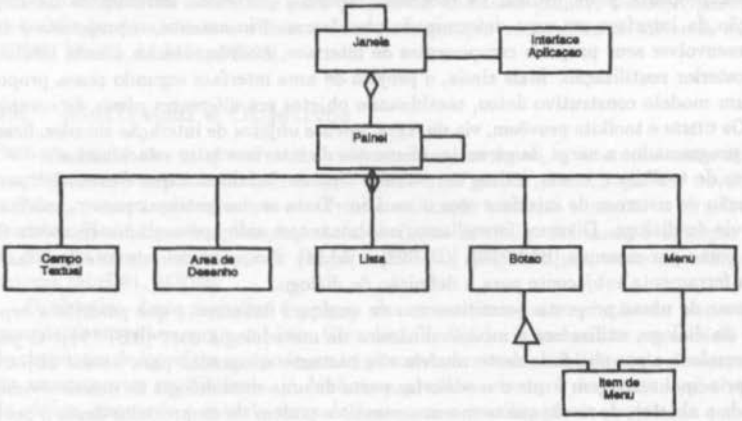

Figura 1: Modelo de Objetos de Interface

A figura 1 mostra as classes e relacionamentos fundamentais do modelo, na notação gráfica proposta pela metodologia OMT [RBP+91]. As convenções utilizadas na representação gráfica do modelo orientado a objetos são:

- Classes são representadas por caixas retangulares;
- Associações são expressas por linhas ligando as classes;
- Agregações são denotadas por um losângulo colocado sobre uma linha de associação;
- Especializações são representadas por um triângulo colocado sobre a linha de associação com a classe genérica.

Antes de detalharmos as classes que formam o núcleo do modelo, é preciso enfatizar a característica de extensibilidade deste núcleo. É possível estender o modelo de elementos de interface de duas formas: pelo acréscimo de classes, representando a incorporação de novos elementos de interface, ou pela especialização de classes, redefinindo e particularizando

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

elementos já existentes no modelo. Vale ressaltar ainda que o modelo é efetivamente orientado a objetos, de modo que uma classe define não apenas os atributos de suas instâncias, mas também o seu comportamento (métodos e mensagens).

2.1 As Classes Janela e Painel

O principal elemento de interface representado no modelo 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 em painéis de controle, geralmente de acordo com a funcionalidade dos componentes. Em outras palavras, uma janela é formada por um conjunto de painéis, onde cada painel agrega, por sua vez, elementos de interface relacionados de acordo com as idéias do projetista de interface.

A estrutura de composição de uma janela é representada no modelo de interface pelo auto-relacionamento existente na classe Painel e pelas agregações que associam a classe Janela à classe Painel, e esta às diversas classes que representam elementos de interface. O auto-relacionamento na classe Painel permite a definição de um painel de controle complexo em termos de outros painéis mais simples.

Esta característica de definição recursiva de um painel de controle provê uma grande capacidade de reutilização de componentes na interface. Um exemplo deve tornar mais clara esta capacidade: considere um painel de controle que permite a seleção de arquivos numa árvore de diretórios. Este painel é um elemento de interface complexo, que pode conter listas para visualização e escolha, campos para entrada de nomes de arquivos, botões para seleção, filtros, entre outros elementos. Uma vez definido, este painel complexo pode ser utilizado como componente de outros painéis. Dessa forma, novos painéis podem reutilizar diretamente o projeto e a implementação do painel de seleção de arquivos.

Além de conter componentes de interação (painéis de controle), uma janela pode estar associada a outras janelas. Este tipo de associação representa 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 existente na classe Janela permite a modelagem deste conceito.

Um terceiro tipo de relacionamento que envolve a classe Janela é a associação com a classe Interface da Aplicação. O objetivo deste relacionamento é prover *independência de diálogo* às interfaces baseadas na biblioteca. A independência de diálogo é uma abordagem onde decisões de projeto que afetam apenas o diálogo usuário-computador são isoladas daquelas que afetam apenas o software computacional e a estrutura da aplicação [HH89]. É fácil ver que esta abordagem simplifica as modificações tanto da interface quanto da aplicação, pelo isolamento existente entre os dois componentes do sistema.

2.2 A Classe Interface da Aplicação

A classe Interface da Aplicação encapsula todos os aspectos da aplicação que são visíveis na camada de interface. Para isso, a classe possui como atributos todas as definições necessárias

ao diálogo com o usuário, e como métodos públicos todas as funções da aplicação que podem ser chamadas a partir de solicitações do usuário. Dessa forma, a classe Interface da Aplicação pode ser considerada como um elemento de ligação único que associa os componentes de interface com o usuário aos componentes da aplicação propriamente dita.

Uma característica importante da classe Interface da Aplicação é que ela representa a visão que os componentes de interface possuem da aplicação subjacente. É este conceito de visão que possibilita a utilização do modelo para construção de interfaces dinâmicas. Para tanto, a classe Interface da Aplicação deve prover um meta-modelo que descreve os dados utilizados na aplicação. É necessário que este meta-modelo seja semanticamente rico, de modo a expressar todos os conceitos e tipos de dados da aplicação. Simultaneamente, o meta-modelo deve seguir algum padrão de descrição de dados, a fim de evitar que cada projeto de interface defina seu próprio meta-modelo.

Para satisfazer estes requisitos, o modelo de interface adota a linguagem de definição de interface orientada a objetos (CORBA IDL) definida na arquitetura de gerenciamento de objetos proposta pelo *Object Management Group* (OMG) [Gro91].

Com o uso da CORBA IDL para descrição do meta-modelo de dados da classe Interface da Aplicação garantimos o suporte semântico necessário, que dispõe inclusive de mecanismos para descrição de métodos e passagem de mensagens, e asseguramos a padronização entre os diversos projetos de interface, no que se refere ao diálogo com a aplicação. Mais ainda, os projetos de interface construídos de acordo com esse modelo se beneficiam automaticamente dos resultados de portabilidade e interoperabilidade de aplicações orientadas a objetos obtidos pelo OMG [WT94].

Com base na definição apresentada da classe Interface da Aplicação, uma janela de interface pode ser construída dinamicamente da seguinte maneira:

1. Ambos o sistema de interface e a aplicação têm conhecimento do modelo de dados (CORBA) que descreve formalmente as informações manipuladas na aplicação (tipos de dados, domínio de valores, etc).

2. O meta-modelo é instanciado, em tempo de compilação, usando a IDL na classe Interface da Aplicação. Em tempo de execução a aplicação envia valores específicos que definem os componentes de interface a serem criados.

3. O sistema de interface interpreta os valores enviados pela aplicação, sempre de acordo com as especificações do meta-modelo, e cria os elementos de interface correspondentes no modelo de objetos de interface proposto na seção 2.

Desta forma os componentes de diálogo não precisam ser definidos estaticamente durante o processo de compilação da interface; ao contrário, eles podem ser inseridos e removidos dinamicamente. Apenas a instanciação do meta-modelo, que estabelece um protocolo de comunicação entre os dois componentes, precisa ser realizada de maneira estática.

A capacidade de construção dinâmica de elementos possibilita a customização da interface, o que aumenta a flexibilidade do sistema e conseqüentemente o grau de satisfação do usuário. É importante observar ainda que a utilização de um meta-modelo para possibilitar a definição dinâmica de componentes contribui para o isolamento entre a aplicação e a interface. O

próprio meta-modelo serve de modelo comum para intercâmbio de informações entre os dois componentes do sistema, estabelecendo, de maneira rigorosa, os limites entre os componentes e os compromissos que devem ser honrados para que haja uma perfeita integração entre eles.

2.3 Outras Classes

O modelo de interface da figura 1 define ainda alguns elementos básicos de diálogo, que são representados pelas classes agregadas à classe Painel. Com estes elementos básicos é possível especificar um vasto conjunto de interfaces visuais, e a extensibilidade do modelo possibilita a inclusão de novos elementos. As classes básicas de elementos de interface são:

- Campo Textual: objetos desta classe possuem valores alfanuméricos e podem ser usados para entrada e saída de dados no sistema de interface. Exemplos: um campo para entrada de senha ou uma tela de editor de textos. Uma subclasse que aparece com freqüência é a classe Rótulo, que representa objetos de interface que servem apenas para a apresentação de informações alfanuméricas, mas não para entrada de dados. Um exemplo típico desta subclasse seria o título de um painel da interface.

- Área de Desenho: nesta classe estão os objetos de interface que permitem a entrada e saída de valores gráficos no sistema de interface. Exemplos de instâncias desta classe seriam a tela de um editor de diagramas e um mapa em um Sistema de Informações Geográficas. Uma subclasse comum é a classe Imagem, a qual apresenta desenhos que não podem ser editados pelo usuário, como por exemplo um ícone.

- Botão: os objetos desta classe permitem a ativação de uma função específica da interface. Esta função pode ser restrita à camada de interface ou pode causar a chamada de funções da aplicação, através de métodos da classe Interface da Aplicação. Um tipo de especialização muito utilizado define a subclasse Botão de Escolha, a qual estabelece um conceito de chaveamento (ligado ou desligado) ao invés de ativar uma função.

- Lista: um elemento de interface desta classe representa uma coleção de elementos homogêneos, os quais podem ser selecionados individual ou coletivamente. Esta classe possui um vasto conjunto de subclasses, contemplando listas ordenadas, conjuntos matemáticos, vetores, tabelas multi-colunadas, planilhas, entre outras.

- Menu: são objetos que permitem a escolha de uma opção entre várias. De fato, a classe é composta por uma agregação de itens de menu, sendo que cada item de menu é um objeto da classe Botão, conforme mostra a figura 1. Uma especialização típica desta classe seria a subclasse Menu em Cascata, cujos componentes podem ser menus e não botões. Outra subclasse comum é Escolha Exclusiva que representa menus onde cada item pertence à subclasse Botão de Escolha, e cuja ativação causa a desativação de todos os outros.

A discussão apresentada nesta seção mostra os aspectos mais importantes do modelo de interface proposto, como a reutilização de projeto, a extensibilidade e a facilidade para construção dinâmica de interfaces. Uma descrição formal completa deste modelo é omitida,

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDF

por questões de espaço. O leitor interessado pode consultar [CM95] para obter maiores detalhes do modelo, bem como um exemplo prático de sua utilização no desenvolvimento de um sistema de interface de grande porte.

3 Implementação do Modelo em uma Biblioteca de Classes

O modelo de objetos de interface é, em si, uma ferramenta importante para reutilização de projetos. A associação do modelo a uma biblioteca de classes permite a reutilização de código e serve como diretiva para a implementação de novas interfaces. Nesta seção vamos mostrar 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. A ênfase na escolha desta plataforma foi a portabilidade e a independência de fornecedor.

3.1 Metodologia de Desenvolvimento

A metodologia utilizada para o desenvolvimento da biblioteca foi a *Object-Oriented Modeling and Design* (OMT) [RBP+91]. Esta metodologia mostrou-se bastante eficiente tanta nas fases iniciais quanto na implementação do projeto. Basicamente são utilizados três modelos para descrever a estrutura e relacionamento 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).

No caso de sistemas interativos de interface, os dois primeiros modelos são os mais importantes já que os elementos de diálogo podem ser arbitrariamente complexos tanto do ponto de vista de estrutura quanto de comportamento. O modelo funcional é secundário, pois em geral os algoritmos que realizam transformações complexas nos dados estão dentro do escopo da aplicação, e não da interface com o usuário. Os modelos de objeto e dinâmico provêem a base necessária para a especificação detalhada da estrutura de hierarquia e composição dos elementos da interface, assim como para a definição do comportamento de cada um destes elementos.

3.2 Orientação a Objetos em Linguagem Convencional

Com o uso da metodologia OMT foi possível manter a visão orientada a objetos, mesmo utilizando uma linguagem convencional para a implementação do projeto. Para isso, a metodologia propõe uma série de 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, utilizamos 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.

PDF compression, OCR, web optimization using a watermarked evaluation copy of CVISION PDFCompressor

- 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, 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, apesar de a linguagem C não oferecer mecanismos diretos para sua representação. Com isso a biblioteca pode ser estendida pela inclusão de novas classes e pela especialização das existentes, da mesma forma como o modelo que lhe serve de base.

3.3 Exemplo de Implementação

Apresentamos a seguir uma parte da implementação das classes Janela e Painel, com o objetivo de ilustrar as convenções descritas anteriormente e também para mostrar como pode ser feita a implementação do modelo de interface aqui proposto.

```
typedef struct Janela *JANELA;
struct Janela /* estrutura principal da classe janela */
{
  JANELA jan_janela_associada_ptr[];        /* auto-relacionamento */
  INTERFACE_APLICACAO jan_interf_aplic_ptr; /* comunicacao 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_registra_msg(JANELA janela_ptr,    /* registra mensagem */
    char * msg);

typedef struct Painel *PAINEL;
struct Painel /* estrutura principal da classe painel */
{
  PAINEL pai_painel_componente_ptr[]; /* auto-relacionamento */
  JANELA pai_janela_ptr;              /* janela ancestral */
  CAMPO_TEXTUAL pai_texto_ptr[];      /* textos agregados */
  AREA_DESENHO  pai_desenho_ptr[];    /* desenhos agregados */
  LISTA pai_lista_ptr[];              /* listas agregadas */
  BOTAO pai_botao_ptr[];              /* botoes agregados */
  MENU  pai_menu_ptr[];               /* menus agregados */
  Widget pai_wd_painel;               /* widget do painel */
};
/* prototipos dos metodos publicos da classe Painel */
extern PAINEL
Painel_constroi(JANELA janela_ancestral); /* construtor */
extern void
Painel_destroi(PAINEL painel_ptr);        /* destrutor */
extern void
Painel_inicializa(PAINEL painel_ptr);     /* inicializa componentes */
extern void
Painel_inibe(PAINEL painel_ptr);          /* bloqueia funcionalidade */
extern void
Painel_habilita(PAINEL painel_ptr);       /* habilita funcionalidade */
```

A próxima seção apresenta o contexto em que o modelo de interface e uma biblioteca de classes como esta podem ser utilizados.

4 Utilização do Modelo e da Biblioteca

O modelo de objetos proposto pode ser utilizado independentemente da biblioteca de classes. Desse modo, o projetista pode planejar seu sistema de interface a partir do modelo construtivo de objetos, tendo a liberdade para implementá-lo através de um *toolkit* para geração de interfaces ou através da biblioteca de classes. Uma outra possibilidade é usar um toolkit para definir novos componentes de interface, inserindo-os em seguida na biblioteca de classes.

Para usar a biblioteca de classes de acordo com nossa proposta é preciso partir do modelo de objetos como ferramenta de estruturação do sistema de interface. Uma maneira bastante

simples e adequada 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. Como foi visto na seção 3, é possível implementar, mesmo em linguagens convencionais, muitos dos conceitos de orientação a objetos.

De posse do modelo de objetos e da especificação de comportamento que definem a interface desejada, a utilização da biblioteca é conceitualmente simples. Os componentes da interface que foram reutilizados de outros projetos podem estar disponíveis na biblioteca, e neste caso, a reutilização é direta. Se um componente desejado se assemelha a algum componente existente na biblioteca, pode ser necessário algum tipo de adaptação do componente e, possivelmente, a criação de um componente mais especializado. Por fim, se nenhum componente existente satisfaz os requisitos do projeto, é preciso implementar o novo componente e adicioná-lo à biblioteca.

É claro que este processo de reutilização exige que sejam oferecidas facilidades para pesquisa, recuperação e atualização de componentes da biblioteca. Esta é, em si, uma área de pesquisa em aberto, e a solução para este problema esta fora do escopo deste trabalho. No restante desta seção vamos discutir a utilização do modelo de interface e da biblioteca em um ambiente específico, onde ferramentas externas foram utilizadas para resolver os problemas citados.

4.1 Uma Aplicação Real

O sistema SAGRE é um projeto que está sendo desenvolvido, desde 1991, no Centro de Pesquisa e Desenvolvimento da Telebrás (CPqD) em Campinas-SP, contando atualmente com a participação de cerca de trinta pesquisadores [sag91, Mag94b, Mag94a]. Este projeto prevê a automatização das atividades ligadas a gerência da rede externa de telecomunicações.

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 que constituem o sistema Telebrás.

O modelo de objetos para construção de interfaces apresentado neste trabalho foi utilizado na evolução de parte da camada de interface com o usuário do projeto SAGRE, especificamente no módulo que realiza o cadastramentro (SAGRE/CAD) [cad93].

No SAGRE os mapas que representam a rede externa são convertidos para a forma digital e armazenados em um Sistema de Informações Geográficas (SIG). Um SIG pode ser compreendido como um sistema de bancos de dados não-convencional que provê funções básicas para a manipulação de dois tipos de informações: dados espaciais e dados descritivos [MP94]. Dados espaciais são georeferenciados, e envolvem representações gráficas e operações geométricas e topológicas. Dados descritivos são as informações alfanuméricas comumente utilizadas em sistemas de bancos de dados tradicionais.

A arquitetura do projeto SAGRE organiza o sistema 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 com o usuário e o módulo denominado Gerente de Atributos (GAT) [gat94], que

implementa o acesso aos atributos alfanuméricos dos componentes da rede externa. Durante uma sessão de trabalho no SAGRE, o usuário visualiza representações gráficas da rede externa, ou seja, mapas. A seleção de um elemento da rede externa ativa o módulo GAT que permite a manipulação dos atributos do elemento selecionado.

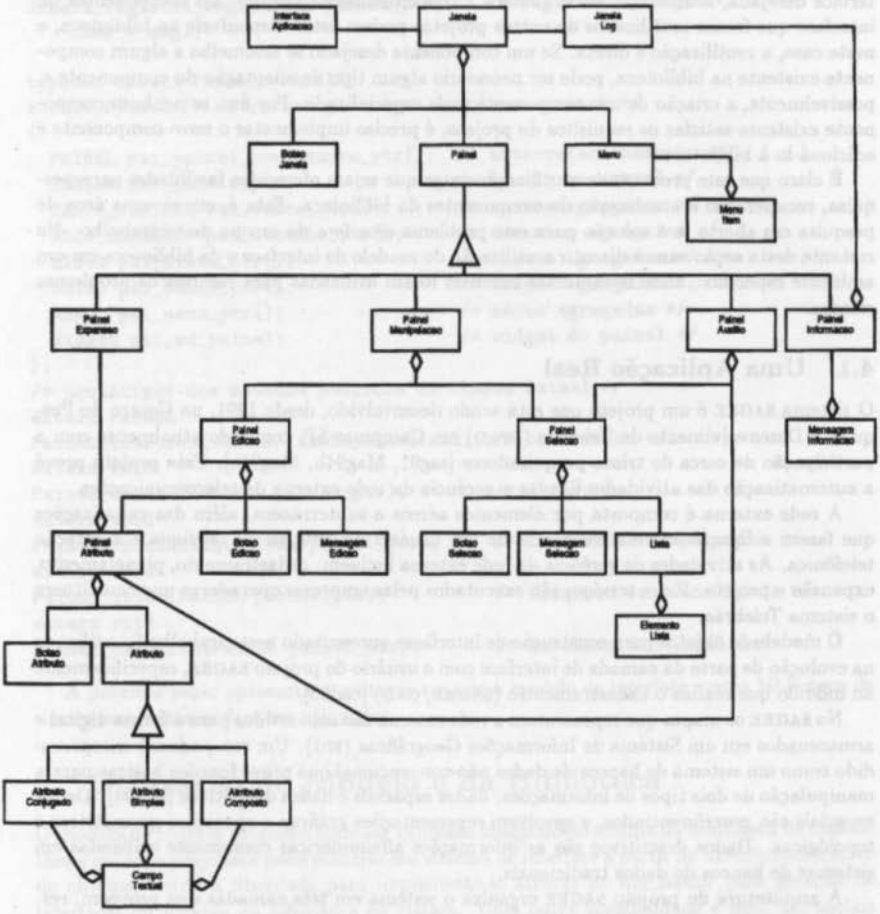

Figura 2: Exemplo de utilização do Modelo de Objetos de Interface

O modelo de interface descrito na seção 2 foi inicialmente proposto para o desenvolvimento do módulo GAT. A figure 2 mostra o projeto de interface do GAT, construído a partir do modelo de objetos da figura 1.

Os resultados da implementação deste módulo demonstraram um coeficiente bastante elevado de reutilização de projeto e de código, o que nos motivou a estender e formalizar o modelo de interface e a definir a biblioteca de componentes reutilizáveis aplicados para interface usuário-computador no ambiente do projeto SAGRE. Estabeleceu-se, a partir deste ponto, um novo método de trabalho dentro do projeto: todos os requisitos de interface com o usuário são atendidos mediante a busca de componentes já existentes na biblioteca, com a incorporação de outras classes de componentes no caso de novas necessidades ou de adaptação de componentes já existentes.

5 Conclusões e Trabalhos Futuros

Neste trabalho apresentamos uma abordagem para reutilização de esforços tanto no projeto quanto na implementação de interfaces visuais. Nossa proposta constitui-se basicamente de um modelo de componentes de interface orientado a objetos e de bibliotecas que implementam este modelo em diferentes plataformas.

O modelo de interface aqui proposto provê sobretudo aspectos conceituais, desenvolvendo uma metodologia para projeto de interfaces. Essa metodologia deve ser embutida em uma metodologia de desenvolvimento de propósito geral, preferencialmente orientada a objetos, para que o projeto de interface se beneficie dos avanços já consolidados nesta área. A associação do modelo a uma biblioteca de componentes de interface simplifica e facilita a passagem do projeto para a implementação de novos sistemas de interface.

Acreditamos que uma importante contribuição desta nova abordagem é a introdução de um modelo de objetos construtivo que permite a descrição de uma interface complexa através da definição recursiva de componentes mais simples. Também relevante nesta proposta é 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 dos sistemas de informação.

É preciso salientar que o objetivo deste trabalho não é propor uma solução para o problema geral de reutilização de componentes no desenvolvimento de software. Nossa contribuição se restringe ao domínio específico de interfaces usuário-computador, tendo por base o paradigma de orientação a objetos. A combinação destas características contribui de modo decisivo para promover a reutilização de componentes de software.

Os problemas envolvidos na reutilização de componentes de software em larga escala são numerosos [Jon94]. Mesmo considerando-se um domínio específico, a solução proposta neste trabalho precisa ser estendida para contemplar os aspectos de classificação, recuperação, evolução e transformação de componentes reutilizáveis.

A abordagem proposta foi implementada e utilizada no desenvolvimento da camada de interface usuário-computador do projeto SAGRE, no Centro de Pesquisa e Desenvolvimento da Telebrás (CPqD). A experiência 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 visuais.

Agradecimentos

Este trabalho foi realizado no Centro de Pesquisa e Desenvolvimento (CPqD) da Telebrás, no Departamento de Sistema de Operações, e teve a participação de diversas pessoas do projeto SAGRE, às quais os autores creditam uma contribuição muito relevante. Gostaríamos de agradecer também ao Professor Fábio Lucena pelas frutíferas discussões sobre sistemas de interface usuário-computador.

Referências

[BD94] Marcel Becker and Jorge L. Diaz-Herrera. Creating Domain Specific Libraries: a methodolog and design guidelines. In *IEEE Proceedings of the Third Internacional Conference on Software Reuse: Advances in Software Reusability*, Rio de Janeiro, Brasil, November 1994.

[cad93] Especificação de Objetivos e Requisitos do Módulo de Cadastramento: Sagre/Cad. Technical Report 04/93, Centro de Pesquisa e Desenvolvimento da Telebrás - CPqD - Campinas - Departamento de Sistema de Operações, April 1993.

[CHB92] Derek Coleman, Fiona Hayes, and Stephen Bear. Introducing ObjectCharts or How to Use Statecharts in Object-Oriented Design. *IEEE Transactions on Software Engineering*, 1(18):9–18, January 1992.

[CM95] Cleida Queiroz Cunha and Geovane Cayres Magalhães. Using an Object-Oriented Metodology in the Development of User Interfaces: a Case Study. Technical Report 05/95, Centro de Pesquisa e Desenvolvimento da Telebrás (CPqD) - Projeto Sagre, May 1995.

[CWC94] Alberto M. de Cima, Cláudia M. L. Werner, and Alessandro A.C. Cerqueira. The Design of Object Oriented Software with Domain Architecture Reuse. In *IEEE Proceedings of the Third Internacional Conference on Software Reuse: Advances in Software Reusability*, Rio de Janeiro, Brasil, November 1994.

[gat94] Projeto de Interface com Usuário: Módulo GAT. Technical Report 10/94, Centro de Pesquisa e Desenvolvimento da Telebrás - CPqD - Campinas - Departamento de Sistema de Operações, October 1994.

[Gro91] Object Management Group. *The Common Object Request Broker: Architecture and Specification.* OMG Document Number 91.12.1, Rev. 1.1, 1991.

[Hel90] Dan Heller. *XVIEW Programming Manual*, volume 7 of *The X Window System Series.* O'Reilly & Associates, April 1990. Second Printing.

[HH89] H. Rex Hartson and Deborah Hix. Human–Computer Interface Development: Concepts and Systems for its Management. *ACM Computing Surveys*, 21(1):5–92, March 1989.

[HLN+90] D. Harel, H. Lachover, A. Naamad, A. Pnueli, M. Politi, R. Sherman, and A. Shtul-Trauring. Statemate: A Working Environment for the Development of Complex Reactive Systems. *IEEE Transactions on Software Engineering*, April 1990.

[Jon94] Capers Jones. Economics of Software Reuse. *IEEE Computer*, 27(7):106–108, 1994.

[LL94] Fábio Lucena and Hans Liesenberg. Reflections on Using Statecharts to Capture User Interface Behaviour. In *Proceedings of XIV International Conference of the Chilean CSS*, October 1994.

[Mag94a] Geovane Cayres Magalhães. The Development of Open System for Engineering Applications. In *International Conference on Automated Mapping/Facility Management*, Denver - USA, March 1994.

[Mag94b] Geovane Cayres Magalhães. The Development of Outside Plant Management Systems. In *Seminário Internacional CINTEL*, Cali - Colômbia, September 1994.

[Mam91] Jean-Claude Mamou. *Du Disque à le ecran: Genération D'Interfaces Homme–Machine Pour Objects Persistants.* PhD thesis, Université de Paris - Sud - Centre d'Orsay, May 1991.

[Mar94] Aaron Marcus. Managing Metaphors for Advanced User Interfaces. In *Prooceedings of the ACM Workshop on Advanced Visual Interfaces*, pages 237–239, Bari, Italy, June 1994.

[MP94] Claudia Bauzer Medeiros and Fátima Pires. Databases For GIS. *ACM SIGMOD Record*, 1994.

[MR92] Brad A. Myers and Mary Beth Rosson. Survey on User Interface Programming. In *Human Factors in Computing Systems CHI'92 Conference Proceedings*, pages 195–202, Monterey, California, May 1992.

[Mra91] Hamid El Mrabet. Outils de Generation de Interfaces: Etat de La Art et Classification. Technical report, Institut National de Recherche en Informatique et en Automatique, February 1991. Rapport Techiniques 126.

[MvD91] Aaron Marcus and Andries van Dam. User-Interface Developments for the Nineties. *IEEE Computer*, 24(9):49–57, September 1991.

[Nor91] Kent L. Norman. Models of the Mind and Machine: Information Flow and Control between Humans and Computers. *Advances in Computers*, 32(1):201–254, 1991.

[Oli94] Juliano Lopes de Oliveira. On the Development of User Interface Systems for Object-Oriented Databases. In *Prooceedings of the ACM Workshop on Advanced Visual Interfaces*, pages 237–239, Bari, Italy, June 1994.

[Ope91] Open Software Foundation. *OSF/Motif - OSF/Motif Programmer's Guide.* Prentice Hall, Inc, 1991.

[PH91] Thiagarajan Palanivel and Martin Helander. Human-Factors Issues in Dialog Design. *Advances in Computers*, 33(1):115–171, 1991.

[PSI93] Arturo Pisano, Yukari Shirota, and Atsushi Iizawa. Automatic Generation of Graphical User Interfaces for Interative Database Applications. In *Proceedings of the ACM Second International Conference on Information and Knowledge Management*, pages 344–355, Washington, DC, USA, 1993.

[RBP+91] James Rumbaugh, Michael Blaha, Wiliam Premerlani, Frederick Eddy, and William Lorensen. *Object-Oriented Modeling and Design.* Prentice-Hall, 1991.

[sag91] Especificação de Objetivos e Requisitos do Projeto Sagre. Technical Report 03/91, Centro de Pesquisa e Desenvolvimento da Telebrás - CPqD - Campinas - Departamento de Sistema de Operações, March 1991.

[Shn83] Ben Shneiderman. Direct Manipulation: A Step Beyond Programming Languages. *IEEE Computer*, 16(8):57–69, August 1983.

[Shn87] Ben Shneiderman. *Designing the User Interface: Strategies for Effective Human-Computer Interaction.* Addison-Wesley, 1987.

[SM94] A.G. Sutcliffe and N.A.M. Maiden. Domain Modeling For Reuse. In *IEEE Proceedings of the Third Internacional Conference on Software Reuse: Advances in Software Reusability*, Rio de Janeiro, Brasil, November 1994.

[Sta94] Werner Staringer. Constructing Applicatinos from Reusable Components. *IEEE Software*, 11(5):61–68, 1994.

[Sun90] Sun Microsystems. *OPENLOOK - Graphical User Interface Applications Style Guidelines.* Addison-Wesley, June 1990. Third Printing.

[Tel91] Telesoft AB Sweden. *TeleUSE - Teleuse Reference Manual.* Telesoft AB Sweden, 1991.

[Vis93] Visual Edge Software. *UIM/X - Developer's Guide.* Visual Edge Software Ltd, 1993.

[WT94] David L. Wells and Craig W. Thompson. Evaluation of the Object Query Service Submissions to the OMG. *IEEE Bulletin of the Technical Committee on Data Engineering*, 17(4):36–45, December 1994.