
# PUC

Monografias em Ciência da Computação
nº 16/92

# Um Prototipador e Gerador de Interfaces Gráficas por Manipulação Direta: Um Modelo Baseado em Eventos

Luís Fernando Diniz Junqueira Barbosa
Carlos José Pereira de Lucena

Departamento de Informática

**PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO DE JANEIRO**
**RUA MARQUÊS DE SÃO VICENTE, 225 - CEP 22453**
**RIO DE JANEIRO - BRASIL**

PUC RIO - DEPARTAMENTO DE INFORMÁTICA

Monografias em Ciência da Computação, Nº 16/92
Editor: Carlos J. P. Lucena Junho, 1992

# Um Prototipador e Gerador de Interfaces Gráficas por Manipulação Direta: Um Modelo Baseado em Eventos *

Luís Fernando Diniz Junqueira Barbosa
Carlos José Pereira de Lucena

* Este trabalho foi patrocinado pela Secretaria de Ciência e Tecnologia da Presidência da República Federativa do Brasil.

# UM PROTOTIPADOR E GERADOR DE INTERFACES GRÁFICAS POR MANIPULAÇÃO DIRETA: UM MODELO BASEADO EM EVENTOS

**Luís Fernando Diniz Junqueira Barbosa**
**Carlos José Pereira de Lucena**
luisfern@inf.puc-rio.br
Departamento de Informática
Pontifícia Universidade Católica, RJ
R. Marques de São Vicente, 225
22453 RJ Brasil

## Resumo

A necessidade de ambientes para o desenvolvimento de software tornou-se cada vez maior nos dias de hoje. Os sistemas desenvolvidos são cada vez mais complexos, exigindo ferramentas de auxílio que atuem em todas as etapas da criação de um software.

O ponto de vista descrito neste trabalho consiste em fornecer um ambiente, no qual se possa desenvolver uma aplicação tendo como ponto de partida sua interface.

Com base em estudos sobre interfaces homem-computador e em ambientes de suporte para criação destas interfaces, definem-se os requerimentos do novo ambiente. A partir desses requerimentos elabora-se um modelo de implementação, no qual se baseia um protótipo desenvolvido segundo as idéias propostas. A seguir, expõe-se um estudo de caso, no qual se descreve o desenvolvimento de uma interface com o apoio do protótipo desenvolvido. Por fim, apresentam-se conclusões, comparações com outros ambientes semelhantes e sugerem-se desdobramentos do trabalho realizado.

PALAVRAS CHAVES: interface com usuário, manipulação direta, prototipação rápida, modelo baseado em eventos e interfaces gráficas

## Abstract

Software developers have come to realize that the user interface paradigm is itself a kind of specification notation that expresses the user's intent and desires in terms of images, as opposesd to words. The user interface implicitly defines most of the functional requirements, i. e., specifying the user interface often suffices to obtain an almost complete system specification. An environment is proposed to support this goal, using an event-based methodology to specify the asynchronous dialogues that exist in an environment that uses direct manipulation with rapid prototyping. To test the environment an interface prototyper and generator system was constructed using direct manipulation based in events.

KEYWORDS: user interface, direct manipulation, rapid prototyping, event-based model, graphic interfaces.

## 1 INTRODUÇÃO ÀS INTERFACES HOMEM-COMPUTADOR

Cada vez mais os computadores estão sendo utilizados por não-profissionais e as possibilidades destes usos estão limitadas não pelo poder computacional, mas pela capacidade de sistemas de se comunicarem com o usuário humano. A partir dessa constatação o desen-

volvimento de interfaces homem-computador passou a fazer parte da engenharia de software. A função primária da interface é fornecer suporte e assistência à aplicação, definindo implicitamente a maioria dos seus requerimentos funcionais. Isto é, ao se prototipar a interface com o usuário consegue-se obter, no caso de aplicações altamente interativas, uma especificação quase completa do sistema [CABR90].

## 1.1 DIÁLOGOS

Neste ponto, torna-se necessário fazer-se uma distinção entre "diálogo homem-computador" e "interface homem-computador". Diálogo é a troca observável de símbolos e ações entre o homem e o computador, já a interface é o software e o hardware que dão suporte a essa troca [HART89].

Os diálogos podem ser classificados como sequenciais, quando o usuário descreve o que fazer, por exemplo, usando uma linguagem de comandos (neste caso só está disponível uma tarefa por vez) e assíncronos, quando o usuário mostra o que fazer manipulando representações de objetos (existe uma multiplicidade de "caminhos" de tarefas que podem ser seguidos).

Uma questão importante é a independência do diálogo [EHRI81], isto é, a separação, baseada numa definição formal, entre a interface e a aplicação. Neste caso, as decisões sobre o diálogo podem ser isoladas, isto é, não afetam a aplicação. A independência permite uma multiplicidade de soluções para um mesmo resultado de diálogo.

Aparece então a figura do desenvolvedor de diálogos, que vem a ser um especialista em fatores humanos preocupado com o projeto, implementação e avaliação da forma, estilo, conteúdo e sequência de comandos em interfaces. Este desenvolvedor faz a análise de tarefas e especificação de requerimentos do sistema, e deve ser sensível às necessidades cognitivas do usuário final.

## 1.2 INTERFACES GRÁFICAS

As interfaces gráficas com o usuário [HAYE89], isto é, os softwares utilizados para construir estas interfaces, compreendem, basicamente, três componentes: um sistema de janelas, que é um conjunto de ferramentas e comandos de programação para construir janelas, menus e janelas de diálogos; um modelo de imagem que define como fontes e gráficos aparecem na tela; e uma interface com o programa de aplicação que é um conjunto de chamadas a funções expressas em uma linguagem de programação, isto é, como o sistema especifica qual janela, menu, barra de seleção ou icone aparecerá na tela.

## 1.3 MANIPULAÇÃO DIRETA

Sistemas nos quais as ações são incrementais, rápidas e reversíveis e nas quais troca-se a sintaxe de linguagens de comandos complexas pela manipulação direta do objeto de interesse, são considerados muito satisfatórios pelos usuários [SHNE83].

A manipulação direta exige uma representação contínua do objeto de interesse, ações físicas ao invés de sintaxe complexa, operações reversíveis, incrementais e rápidas cujo

impacto no objeto de interesse se torna imediatamente visível e um enfoque "em níveis ou em espiral" para o aprendizado que permita o uso com o mínimo de conhecimento.

Sistemas que trabalham com manipulação direta, apresentam os seguintes aspectos favoráveis: novatos aprendem mais rápido, usualmente através de demonstração; especialistas podem trabalhar de forma extremamente rápida; mensagens de erro são raramente necessárias; usuários podem ver imediatamente se suas ações estão alcançando seus objetivos; usuários apresentam menos ansiedade e ganham confiança e domínio sobre a aplicação, além de outros.

O único problema da manipulação direta reside no fato de ser necessária uma representação visual contínua do objeto de interesse. Esta representação, se não for muito bem elaborada, não consegue transmitir o significado que se pretende, podendo confundir o usuário.

## 1.4 DIÁLOGO ASSÍNCRONO VIA MANIPULAÇÃO DIRETA

Existem várias técnicas para descrever um diálogo que usa manipulação direta. É necessário encontrar um meio para capturar o ponto de vista do usuário em uma interface com manipulação direta. A questão é descrever a interface ou diálogo entre o sistema e o usuário final, ao invés de descrever a estrutura do sistema ou seus componentes em qualquer nível.

A especificação do diálogo deve se centrar na sequência de entradas abstratas e eventos de saída que o compõe. A sintaxe de uma interface interativa com o usuário – seja ela convencional ou por manipulação direta – pode ser descrita por tal sequência, com a especificação dos significados dos eventos em termos de ações de entrada específicas ou imagens na tela.

O usuário deve ter a capacidade de interromper o trabalho em determinado objeto, mudar para outro e retornar, sem perder o que já tinha feito com este objeto. Este esquema é muito parecido com o das corotinas.

Então uma técnica de especificação para interfaces de manipulação direta deve permitir que diálogos individuais sejam especificados isoladamente e que estes possam trocar o controle entre si, através de um mecanismo que se assemelha ao das corotinas.

## 1.5 UTILIZAÇÃO DA DEMONSTRAÇÃO

Uma das capacidades essenciais de um ambiente onde se desenvolvem interfaces é permitir ao usuário visualizar como está a interface em um determinado momento, e como ela funcionará [MYER86]. Uma maneira de se alcançar esta capacidade é a utilização de técnicas de manipulação direta associadas a algumas técnicas de controle e auxílio ao usuário, por exemplo, através de regras e diagramas de estado.

Portanto, é muito importante que o ambiente possa demonstrar o funcionamento da interface que ele está gerando, para que o desenvolvedor saiba se o seu projeto está no caminho certo, ou seja, analisando previamente qualquer dificuldade que o usuário final poderá vir a ter, diminuindo portanto, o índice de rejeição de protótipos de interface pelo usuário final.

## 1.6 PROTOTIPAÇÃO RÁPIDA

Métodos para desenvolvimento e avaliação de interfaces são muito mais analíticos do que sintéticos por natureza, isto é, algo deve ser construído primeiro, analisado e iterativamente refinado. O presente estado do conhecimento sobre fatores humanos não permite a síntese de uma interface que fique pronta da primeira vez [HART89]. A "fatorabilidade humana", requer ferramentas de desenvolvimento de diálogos que produzam protótipos rapidamente para permitir uma prévia observação do comportamento da interface e permitir modificações fáceis de projeto.

Prototipação, no âmbito do desenvolvimento de sistemas, significa a produção de, pelo menos, uma versão prévia do sistema que demonstre suas capacidades essenciais. É uma técnica e não uma ferramenta. Protótipos reduzem as chances de surpresas para o usuário final e reduzem também o problema da inabilidade do usuário final em fornecer especificações completas sobre o sistema ao projetista.

Um ambiente de suporte deve manter o protótipo funcionando apesar dele ser, a princípio, incompleto, ambíguo e repleto de erros e tentativas iniciais.

## 1.7 MODELOS ESTRUTURAIS DE INTERFACES

Os três modelos mais proeminentes de interação homem-computador são: análise de tarefas, descrição estrutural e representação de interface. O primeiro é utilizado para analisar e descrever detalhes de uma tarefa particular do usuário, geralmente por decomposição hierárquica em níveis de sub-tarefas. O segundo trata de descrições do processo geral de comunicação homem-computador, descrevendo a estrutura das trocas entre o usuário final e o computador. O último se baseia em esquemas para representar instâncias particulares da interação homem-computador. Estes modelos podem ser classificados como linguísticos e não-linguísticos [HART89].

## 1.8 REPRESENTAÇÃO DA INTERFACE HOMEM-COMPUTADOR

Desenvolvedores de diálogos requerem um mecanismo para expressar e guardar seus projetos. Recentemente, sistemas para desenvolvimento de diálogo tem começado a fornecer ferramentas automatizadas para produção interativa de representações de interfaces. Elas se baseiam desde metalinguagens até representações completas das tarefas do desenvolvedor.

Essas técnicas podem ser classificadas de acordo com o tipo de diálogo a ser tratado. Para os diálogos sequenciais podemos citar:

- BNF [MAUR63];
- Diagramas de Estado [JACO86];
- Representação por Linguagem de Diálogo [HART89].

Já para os diálogos assíncronos, temos:

- Representação Baseada em Eventos [GREE86].

## 1.9 FERRAMENTAS PARA O DESENVOLVIMENTO DE INTERFACES

Este tipo de ferramenta é frequentemente utilizado para automatizar o processo de representação da interface. Desenvolvedores de diálogos usam essas ferramentas para outras atividades, incluindo, prototipação, avaliação, análise e implementação.

A terminologia para ferramentas de desenvolvimento de interfaces ainda não foi estabelecida. Podemos usar o termo genérico "ferramenta" para nos referirmos desde ambientes de desenvolvimento de interfaces completos até o simples recurso de uma interface pequena.

Um UIMS (User Interface Management System) é um conjunto de programas interativos de alto nível para projetar, prototipar, executar, avaliar e manter interfaces com o usuário, tudo integrado sob uma interface de desenvolvimento de diálogo simples.

Podemos destacar alguns requerimentos para estes ambientes [CABR90] [HART89]:

- deve estar bem claro o que a ferramenta pode fazer;
- ferramentas devem aumentar a produtividade;
- não deve ser necessário o entendimento de todo o sistema para que se possa realizar uma tarefa;
- a ferramenta deve permitir reversibilidade de ações;
- além de outros.

Ambientes de programação orientados a objetos tem atraído a atenção de implementadores de ferramentas [SIBE86]. Estes oferecem vantagens por permitir esconder a informação necessária para representar objetos independentemente de suas implementações. Por sua natureza baseada em eventos, a orientação a objetos é efetiva na representação de diálogos assíncronos. A capacidade para ajuste dinâmico, a definição hierárquica com herança de atributos e procedimentos e a comunicação por passagem de mensagens trabalham juntos para dar suporte ao compartilhamento, a reutilização, a consistência, a flexibilidade e outros, atributos de interesse dentro das implementações das interfaces.

Ainda pode-se falar das UIDS (User Interface Design Systems) [CABR90], onde o ambiente participa do projeto da interface. O meio de se obter este tipo de capacidade é a integração de uma assistência baseada em conhecimentos à técnicas de projeto de interfaces. O ambiente conteria conhecimento sobre projeto de interface e princípios sobre o projeto do software da aplicação, podendo reconhecer as ações do usuário através de um estudo dos planos de desenvolvimento do usuário [GUAR89].

# 2 CARACTERÍSTICAS GERAIS DO AMBIENTE

A partir dos estudos feitos, foram determinados os requerimentos do nosso ambiente para criação de interfaces gráficas. Ficaram claros diversos pontos que são explicados detalhadamente a seguir.

Como ponto principal do estudo realizado ficou claro que o **ambiente deveria ser orientado às necessidades cognitivas do desenvolvedor da interface**, já que qualidade se refere à satisfação de pessoas sobre determinado produto.

Para se criar uma interface gráfica é necessário que se trabalhe o mais perto possível das representações visuais dos elementos que vão compor esta interface. A melhor maneira de se fazer isto é manipular diretamente essas representações visuais. Por isso, um ponto chave do ambiente, foi **permitir a criação das janelas que comporão a interface através da manipulação direta das representações visuais dos elementos de interfaces gráficas que farão parte do ambiente.**

Tendo em vista a escolha acima tornou-se necessário fazer com que os **elementos que comporiam a interface pudessem ser combinados dinamicamente**, isto é, que na tela, durante a criação da interface, o usuário pudesse ver como os elementos se combinam.

Outro ponto chave que surgiu dos estudos realizados, foi dar ao usuário o poder de criar várias alternativas da sua interface através da capacidade de avançar por um determinado caminho e poder desfaze-lo da maneira mais simples possível. Com isso chegou-se à conclusão que o **ambiente deveria permitir a criação da interface segundo um esquema de prototipação rápida evolutiva e intermitente.**

A partir do requerimento anterior ficou claro que **o ambiente deveria permitir a demonstração do funcionamento da interface a qualquer ponto de sua criação.** É fácil justificar esse requerimento pois uma das partes fundamentais de uma interface gráfica é o funcionamento de sua parte visual, isto é, alteração de forma, aparecimento de menus e outros. Portanto foi fundamental permitir que o desenvolvedor da interface pudesse, a qualquer momento, ver como atuaria visualmente a interface quando pronta.

Uma interface gráfica não existe sozinha: existe para uma aplicação. A interface, por meio das associações entre ações sofridas na sua parte visual com procedimentos da aplicação, permite o funcionamento da aplicação. Sob este ponto de vista tornou-se claro que **o ambiente deveria fornecer um meio para associar ações sofridas pela interface com procedimentos da aplicação.**

Como sabemos que quem aciona a interface gráfica, mesmo que essa tenha controle, é a aplicação, e que são exigidos, pelas ferramentas de criação de interfaces, alguns procedimentos para este acionamento, ficou claro que **o ambiente deveria fornecer um padrão de programa onde seria incorporada a aplicação, sendo que este padrão já conteria todos os procedimentos para o acionamento da interface e seu perfeito funcionamento, além de fornecer exemplos práticos da associação de ações da interface com procedimentos da aplicação. Por fim, este padrão, como está sob a responsabilidade do ambiente, deve poder ser executado a qualquer momento.**

Levando-se em conta os requisitos gerais descritos anteriormente, foram estabelecidos os requisitos internos do ambiente:

- manter um diálogo assíncrono;
- ter especificação centrada numa coleção de elementos individuais da interface gráfica, cada um com especificação separada;
- elementos devem ser compostos dinamicamente e poder ter sua parte visual ativada a qualquer momento;
- reconhecer determinadas entradas e saídas em sequência que possa ser interrompida

a qualquer ponto e retomada posteriormente no mesmo ponto, mantendo a independência das entradas e saídas do "layout" e das representações visuais;

- ter cada diálogo da interface especificado como uma interação entre um dos elementos do ambiente e as descrições dos diálogos independentes;
- diálogos relacionados num esquema semelhante a corotinas, isto é, podendo ser interrompidos a qualquer ponto e retomados sem perda de estado;
- ter controle do diálogo ativo por manipuladores de eventos;

O ambiente para geração de interfaces através de manipulação direta foi representado por uma coleção de elementos que interagem entre si, organizados em torno de elementos manipuláveis e com o estado do diálogo constantemente "lembrado".

Seguiram-se os sequintes procedimentos para se obter o resultado desejado: definiu-se a coleção de elementos de interface gráfica; especificou-se o comportamento desses elementos; e providenciou-se um mecanismo para combiná-los e para controlar os respectivos diálogos, no esquema de corotinas.

Para que a interface possa ser dividida em elementos individuais é necessário que se decida qual será a menor unidade significativa com a qual o usuário poderá conduzir um diálogo relevante e que, também se permita que o respectivo diálogo possa ser lembrado tanto quando o diálogo é interrompido ou reassumido. Também para manter a divisão em elementos deve-se especificar um manipulador de eventos (entrada e saída) para cada elemento. Com isso, cada elemento que recebe entradas sequenciadas pode ter seu diálogo lembrado quando for interrompido (corotinas).

Como em [SIBE86], a melhor maneira de se especificar elementos de interfaces gráficas é estruturá-los em classes com heranças e outras características, o que leva a uma orientação a objetos. Neste ambiente a coleção de elementos gráficos será limitada pelo conjunto mínimo especificado e o desenvolvedor só poderá desenvolver novos elementos compondo elementos pré-existentes.

No âmbito do controle do ambiente, para que a especificação dos elementos possa ser combinada no sentido global da interface, é necessário que haja um "executivo" (manipulador geral) que controla entradas e saídas (eventos) e ativa ou interrompe o diálogo individual entre o elemento e o usuário, lembrando o estado deste num esquema de corotinas.

## 2.1 A ESCOLHA DO MOTIF

Tendo em vista as decisões de que cada elemento deveria poder ser composto dinamicamente, ter "layout" pré-definido que pudesse ser alterado e que possibilitasse a demonstração do funcionamento de sua parte visual a qualquer momento, isto levou a escolha do Motif [OSFM90] que é um ambiente de criação de interfaces gráficas no qual os elementos são estruturados em classes, onde cada classe tem uma lista de atributos "default", que podem ser compostos dinamicamente, e que tem ações visuais ligadas a cada elemento.

O Motif contém a UIL (User Interface Language) que é uma linguagem de especificação de interfaces gráficas nos seus aspectos estruturais, não fornecendo meios para especificar o modo de agir da interface.

Neste ponto notamos que **o ambiente deveria guardar o trabalho feito pelo desenvolvedor de interfaces de uma maneira simples e bastante descritiva, que permita a atuação do desenvolvedor diretamente neste meio de armazenamento caso ele o deseje.**

Outro ponto importante do Motif é que ao permitir a composição de seus elementos dinamicamente e trabalhar sobre o X Window [SCHE86], que por sua vez trabalha num esquema de eventos, ele nos permite manter o diálogo no esquema de corotinas. Qualquer aplicativo que trabalhe sobre o X Window, é cliente de um processo chamado servidor que controla todas as ações dos usuários na tela, transformando-as em eventos e transmitindo-as para os seus respectivos manipuladores através de mensagens passadas através de uma rede.

O Motif por manter a especificação da interface num módulo separado da aplicação fornece a independência do diálogo [EHRI81].

## 2.2 ESPECIFICAÇÃO DO AMBIENTE

A primeira atividade realizada foi a especificação da coleção de elementos. Considerou-se as características dos elementos já fornecidas pelo Motif e acrescentaram-se outras impostas pelo nosso ambiente para produzir-se uma descrição de cada elemento. Esta descrição diz: como o elemento é especificado pelo Motif; com quais elementos podem ser compostos outros elementos e que argumentos são herdados para cada composição; quais as listas dos valores iniciais dos argumentos de cada elemento, especificando os que podem ser alterados pelo usuário; quais são as estruturas pré-definidas para cada elemento; que tipos de manipulações o elemento pode sofrer e que tipos de saídas ele vai apresentar.

Os elementos que compõem o ambiente aqui descrito são: JANELA BASE, BOTÃO DE AÇÃO, BOTÃO DE MENU, MENU DE BARRA, MENU, LISTA E TEXTO. Podem ser vistos na figura 2.1.

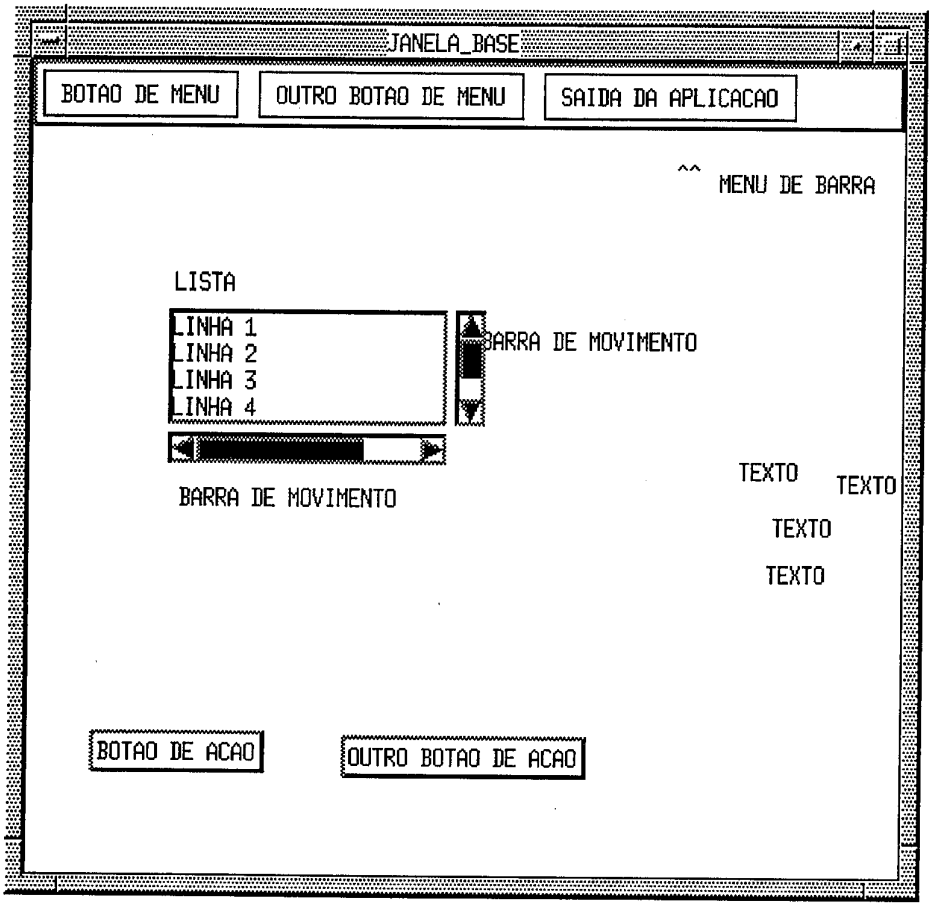


Figura 2.1 - Elementos de Interface Gráfica

Para que o ambiente possa ter o controle da criação da interface, existe uma estrutura interna relativa a cada elemento que permite a criação da especificação da interface e fornece subsídios à geração da aplicação que vai ativar a interface. Esta estrutura é demonstrada a seguir.

```
ESTRUTURA {
        NOME:           nome dado ao elemento (exigência UIL)
        WIDGET:         identificador MOTIF do elemento
        TIPO:           tipo do elemento (se botão, se lista, ...)
        DELETADO:       se elemento foi "deletado" ou não (UNDO)
        FILHOS:         lista de elemento contidos
        PROCEDIMENTO:   estrutura do procedimento associado
            }
```

Com a definição da coleção dos elementos que fizeram parte do ambiente, necessitou-se estipular quais seriam as manipulações que estes elementos poderiam sofrer, baseadas no ponto de vista do usuário:

| | |
|---|---|
| JANELA BASE: | Pode ser movimentada através da seleção do título que a compõe e sofrer alteração do seu tamanho através dos cantos da janela. Para ser acessada para trabalho deve ser selecionada através da representação que o ambiente lhe atribui. |
| MENU DE BARRA: | Pode ser posicionado dentro de uma JANELA BASE. Depois de selecionado, o acionamento do "mouse" dentro da JANELA BASE determinará o posicionamento deste. Se o acionamento do "mouse" for perto das bordas superiores ou inferiores o MENU DE BARRA será posicionado horizontalmente e será posicionado verticalmente se a ação do "mouse" for perto de uma das laterais da JANELA BASE. |
| BOTÃO DE AÇÃO: | Depois de selecionado pode ser posicionado em uma JANELA BASE através do acionamento do "mouse" em cima deste. |
| BOTÃO DE MENU: | Depois de selecionado pode ser posicionado em uma JANELA BASE ou em um MENU DE BARRA através do acionamento do "mouse" dentro destes. No caso do MENU DE BARRA será posicionado próximo ao ponto de acionamento do "mouse", pois o MENU DE BARRA controla o posicionamento. |
| MENU: | Depois de selecionado deve ser posicionado em um BOTÃO DE MENU através do acionamento do "mouse" em cima deste. |
| LISTA: | Depois de selecionada pode ser posicionada dentro da JANELA BASE. |
| TEXTO: | Depois de selecionado deve ser posicionado dentro da JANELA BASE através do acionamento do "mouse" em determinada posição desta. |

Já foi discutido anteriormente o que se entende por diálogo, os tipos de diálogo e como modelá-los. A escolha recaiu, tendo em vista as necessidades da manipulação direta e a orientação a usuários seguida, sobre diálogos assíncronos. Isto é, o usuário pode interromper seu diálogo com o sistema a qualquer tempo, iniciando outro diálogo ou continuando outro que já tenha sido interrompido, depois voltando ao diálogo anterior no mesmo estado em que estava quando foi interrompido. Este foi o requisito mais importante para a escolha do meio de se especificar o diálogo.

Para que sejam melhor tratados, divide-se o diálogo global, entre o usuário e o sistema em diálogos entre o usuário e cada elemento da coleção que compõe o ambiente, sendo cada um desses diálogos independentes um dos outros.

Focalizando a atenção no Motif, mais diretamente no X Window, vemos que as ações do "mouse" na tela, que é o modo do usuário interagir com interfaces gráficas, são interpretadas como eventos, isto é, fatos relevantes ao sistema, como o evento de se acionar um dos botões do "mouse" numa região da tela, o evento de se entrar com o cursor (representação visual do "mouse") dentro de uma janela aberta na tela, e outros que são classificados e bem especificados pelo X Window, fornecem um modo de capturar as ações do usuário.

Esses eventos são capturados no esquema de "loop infinito", isto é, um loop de eventos, onde se lê um evento, trata-se o evento e se espera por outro. Como o evento pode ocorrer em qualquer momento e lugar (interrupção provocada pelo "mouse"), teremos, então, um diálogo assíncrono.

Para capturar e tratar as ações do usuário, através do reconhecimento dos eventos que estão acontecendo, necessita-se de manipuladores de eventos, isto é, procedimentos que

reconheçam basicamente com qual elemento o usuário quer iniciar um diálogo e acionem outros procedimentos (também manipuladores de eventos) que vão reconhecer e tratar os eventos deste determinado diálogo.

Um fato importante não pode ser esquecido que é a participação do manipulador de eventos no diálogo. Para que ele não interfira no esquema de loop infinito é necessário que ele seja capaz de realizar suas tarefas completamente quando recebe o controle do diálogo. A maneira encontrada para isso foi especificar os manipuladores orientados a uma ação simples do usuário, isto é, a um tipo de evento. O tipo de evento escolhido foi o de pressionamento do botão mais a esquerda do mouse sobre o elemento.

Na criação de um elemento para compor a interface é associado ao acionamento do botão mais a esquerda do "mouse" sobre este elemento, um procedimento que indentificando o estado do sistema chamará o manipulador de eventos respectivo, que por sua vez, realizará sua tarefa e retornará o controle ao loop infinito do Motif e X Window (Figura 2.2).

Resumindo, o sistema se constitui de uma coleção de elementos que podem ser manipulados diretamente pelo usuário. Há um manipulador de eventos global (o executivo) que age como um controlador de diálogos, acionando determinados manipuladores de eventos, que foram especificados para o tratamento do diálogo entre o usuário e as instâncias do elemento para o qual foi preparado.

Até agora descrevemos as preocupações com a composição visual da interface. Entretanto, uma interface não se limita a sua parte visual, isto é, a ela está associado um modo de agir. Fala-se dos efeitos que determinada ação na interface desencadeiam, não só na aplicação mas também na interface. O fato de um botão ser pressionado pode implicar no aparecimento de uma janela, na alteração de estado de determinados elementos da interface e outras ações.

A associação de ações sofridas pelos elementos da interface gráfica com procedimentos na aplicação vai permitir a construção completa da interface, de sua parte visual e seu modo de operação. Existem dois tipos básicos de ações que os elementos da interface podem desencadear: o primeiro é aquele tipo de ação relativo diretamente a aplicação e que pode ter alguma influência na interface, e o segundo são as ações ligadas diretamente a alteração da parte visual da interface, como por exemplo, o aparecimento de outras janelas. O primeiro tipo é de responsabilidade do desenvolvedor da aplicação, já o segundo é de responsabilidade do ambiente.

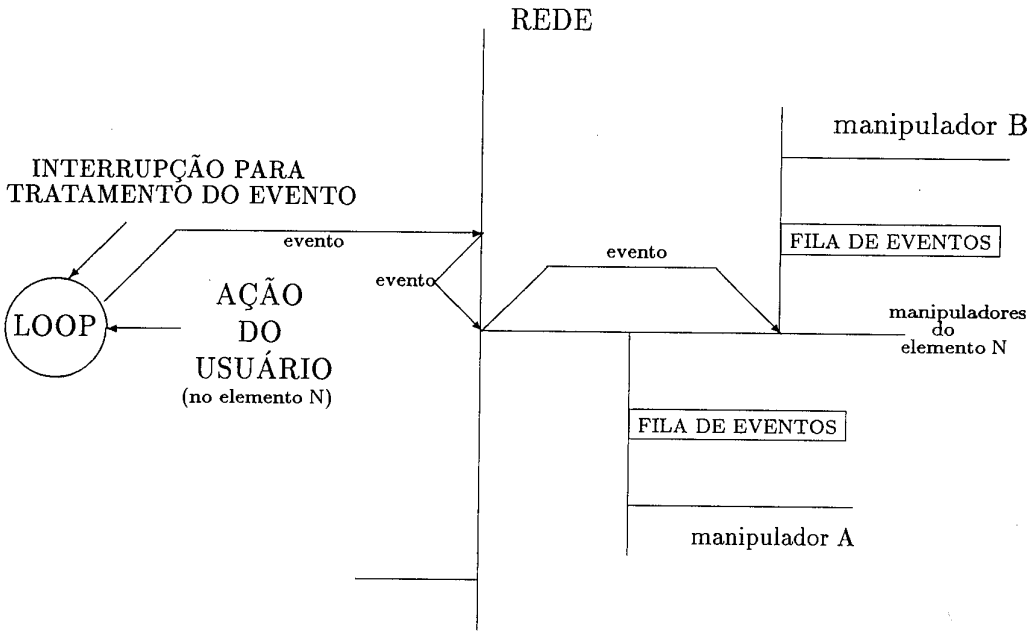


Figura 2.2 - Modelo Baseado em Eventos

Cada elemento tem um elenco de ações relacionadas a interface gráfica que podem ser associadas a ações na aplicação. O Motif, através da UIL, permite associação dessas ações a cada elemento.

Para controlar essa associação de ações utiliza-se uma estrutura simples, abaixo demonstrada, que permite informar à interface, via UIL, a associação, e permite gerar na aplicação o código necessário para a realização da tarefa desejada.

```
ESTRUTURA {
                NOME:    nome do procedimento (aplicação)
                RAZÃO:   tipo de ação sofrida pelo elemento
                TIPO:    tipo de ação a ser implementada
                VALOR:   parâmetro fornecido pelo usuário
          }
```

O ponto de vista do ambiente é o de fornecer um programa padrão em C no qual a aplicação possa ser incorporada posteriormente. Esse padrão de programa será responsável por todos os procedimentos que a aplicação deve possuir para acionar a interface, e, dependendo das associações feitas, fornecerá o esqueleto do procedimento dando "dicas" sobre como ter acesso aos dados passados pela interface.

O objetivo do programa padrão em C é resumir ainda mais a necessidade do desenvolvedor de se envolver em assuntos relativos à interface gráfica, resumindo seu trabalho à aplicação. Este modo de agir do ambiente, dependendo de quantos procedimentos e funções da ferramenta de criação da interface (nosso caso Motif) ele sabe tratar, reduz muito o trabalho que qualquer desenvolvedor de interfaces venha a ter na aplicação.

Uma das questões mais importantes é que esse programa padrão em C, enquanto for de

responsabilidade do ambiente, seja a qualquer momento executável, isto é, o desenvolvedor deve poder compilá-lo e ver a interface funcionando. Também, através do programa padrão em C, deve receber a confirmação de que as ações sofridas pela interface, quando associadas a ações na aplicação, estejam sendo identificadas. Por exemplo, no caso de uma LISTA que o desenvolvedor possa acionar um item da lista e receber uma resposta, que, no caso, seria a impressão do item escolhido.

A cada chamada de procedimento, são passados três parâmetros, o primeiro é o identificador do elemento, o segundo é um ponteiro para um dado do usuário e o terceiro um ponteiro para uma estrutura que fornece informações sobre o elemento, e dicas sobre acesso aos dados dessa estrutura devem ser passadas nos corpos desses procedimentos.

Pelos motivos até aqui estudados, foram criadas as JANELAS DE AÇÃO que apresentam meios, por manipulação direta, para se associar um procedimento da aplicação a um elemento da interface.

Com esta estratégia o tratamento de eventos é feito pelo ambiente quando estes estão associados a um tipo de ação. A interdenpendência entre os elementos da interface gráfica é controlada pelo ambiente quando esta é relativa à um tipo de ação implementada pelo ambiente. Portanto, o desenvolvedor não precisa mexer no código da interface, só no da aplicação, mesmo quando esteja trabalhando sobre o modo de agir da interface.

Cada elemento tem um determinado conjunto de características (valores de seus argumentos) que podem ser alteradas. Por isso, aliada à chamada do manipulador de eventos quando se está criando ou alterando um elemento, associa-se o aparecimento de uma JANELA DE CARACTERÍSTICAS que fornecerá meios para a alteração dessas características. Excetuando a digitação de textos estes meios serão por manipulação direta. Estas JANELAS DE CARACTERÍSTICAS só precisam ser acessadas quando e se o desenvolvedor desejar, pois forçar que o usuário tenha que atuar nesta janela, interfere com o diálogo assíncrono desejado e prejudica a performance. As JANELAS DE CARACTERÍSTICAS são fundamentais mas não devem interferir com o diálogo.

Quanto mais elaboradas forem as JANELAS DE CARACTERÍSTICAS e as JANELAS DE AÇÕES, mais completo o ambiente será em relação à ferramenta que usa para criação da interface, gerando uma melhor interface e resumindo ao máximo possível o trabalho feito na aplicação com relação à interface gráfica.

Neste ponto, torna-se necessário saber o que o usuário pretende ao acionar o "mouse" em determinado local. Para se identificar o que o usuário pretende fazer só são necessárias duas informações: se o usuário quer criar, alterar, destruir ou associar ação a um elemento, resumindo o ESTADO em que o ambiente se encontra e qual é esse ELEMENTO.

Com isso conlui-se que o ambiente deve compreender uma área de escolha de ESTADO: [CRIAR], [ALTERAR], [DESTRUIR], [AÇÃO] e uma área de escolha do tipo de elemento a ser trabalhado, e como estamos trabalhando com manipulação direta, esta área de escolha do tipo de elemento deve ser composta de representações visuais dos elementos que compõem o ambiente.

Com o acionamento de manipuladores de eventos associados ao pressionamento do botão mais a esquerda, com a estipulação do ESTADO e do ELEMENTO a ser trabalhado e com os manipuladores especificados para tratar o pressionamento do "mouse", sem prender o controle, conseguimos especificar o diálogo do ambiente, como um diálogo assíncrono, por manipulação direta.

A seguir temos uma descrição dos modos de ação do manipulador de eventos principal, chamado de agora em diante de EXECUTIVO e que aparece na Figura 2.3, seguido do manipulador de um dos elementos do ambiente. Faz-se abaixo uma especificação em linguagem natural semelhante às linguagens de programação estruturadas:

EXECUTIVO

-SE nem ESTADO nem ELEMENTO escolhido
ENTÃO ERRO ("NEM ESTADO NEM ELEMENTO FORAM DEFINIDOS")
-SE só ESTADO não foi escolhido
ENTÃO ERRO ("ESTADO DO SISTEMA NÃO FOI DEFINIDO")
-SE só ELEMENTO não foi escolhido
ENTÃO ERRO ("ELEMENTO A SER TRABALHADO NÃO FOI DEFINIDO")
-identifica tipo de elemento com qual se está trabalhando
-CASO tipo de elemento FAÇA

| | |
|---|---|
| -BOTÃO DE AÇÃO : | chamar manipulador de botão de ação; |
| -BOTÃO DE MENU : | chamar manipulador de botão de menu; |
| -MENU DE BARRA: | chamar manipulador de menu de barra |
| -MENU : | chamar manipulador de menu |
| -LISTA : | chamar manipulador de lista |
| -TEXTO : | chamar manipulador de texto |

O ESTADO do sistema ([CRIAR], [ALTERAR], [DESTRUIR], [AÇÃO]) é reconhecido por cada manipulador (Figura 2.3).

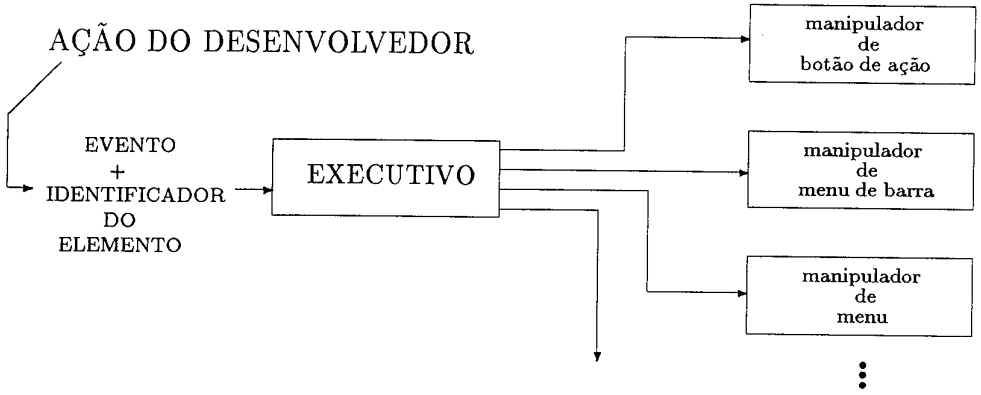


Figura 2.3 - Executivo e Manipuladores

## MANIPULADOR DE BOTÃO DE AÇÃO

Reconhece as seguintes intenções do usuário sobre os BOTÕES DE AÇÃO:

- colocá-lo em determinada posição na JANELA BASE, através da seleção no menu de representações visuais e estado = [CRIAR]. O posicionamento do botão é feito através do pressionamento do botão mais a esquerda do "mouse" na posição desejada.
- alterar a posição através da seleção do mesmo na JANELA BASE e estado = [ALTERAR]. Quando ocorrer o segundo pressionamento do "mouse", que deve ser na nova posição, o botão será retirado da posição antiga e posto na nova.
- retirá-lo da JANELA BASE através da seleção do mesmo na JANELA BASE e estado = [DESTRUIR].
- associar uma ação através da seleção e estado = [AÇÃO]. Neste caso específico a ação pode ser associada ao aparecimento ou à ativação do botão.

## DESCRIÇÃO INFORMAL DO MANIPULADOR DE BOTÃO DE AÇÃO

```
- recebe IDENTIFICADOR do elemento que sofreu ação do "mouse"
- recebe EVENTO
- caso ESTADO faça
  [CRIAR]      : { SE tipo do elemento é JANELA BASE
                   ENTÃO - criar BOTÃO DE AÇÃO na posição (EVENTO)
                           - escolher nome para o BOTÃO DE AÇÃO
                           - acionar JANELA DE CARACTERÍSTICAS
                           - relacionar BOTÃO DE AÇÃO à JANELA BASE
                   SENÃO - erro("POSIÇÃO INVALIDA !!") }
  [DESTRUIR] : { SE tipo do elemento é BOTÃO DE AÇÃO
                   ENTÃO - retira BOTÃO DE AÇÃO
                   SENÃO - erro("ELEMENTO NÃO É BOTÃO DE AÇÃO") }
  [ALTERAR]  : { SE TIPO DO ELEMENTO é JANELA BASE && ALTERANDO
                   ENTÃO - reposicionar BOTÃO DE AÇÃO
                           - alterando = FALSE;
                   SE tipo do ELEMENTO é BOTÃO DE AÇÃO && não alterando
                   ENTÃO - identificar BOTÃO DE AÇÃO a ser alterado
                           - alterando = TRUE
                           - acionar JANELA DE CARACTERÍSTICAS
                   SE tipo do elemento não é BOTÃO DE AÇÃO && não alterando
                   ENTÃO - erro("ELEMENTO NÃO É BOTÃO DE AÇÃO")
                   SE tipo do elemento não é JANELA BASE && alterando
                   ENTÃO - erro("NÃO POSSO DESLOCAR BOTÃO") }
  [AÇÃO]       : { SE tipo do elemento é BOTÃO DE AÇÃO
                   ENTÃO - aciona JANELA DE AÇÃO para BOTÃO DE AÇÃO
                   SENÃO - erro("ELEMENTO NÃO É BOTÃO DE AÇÃO") }
```

## JANELAS DE CARACTERÍSTICAS E DE AÇÃO DO BOTÃO DE AÇÃO

Como podemos ver na Figura 2.4, a JANELA DE CARACTERÍSTICAS do BOTÃO DE AÇÃO, permite a alteração do nome associado ao BOTÃO e da largura de sua borda. Já na Figura 2.5, temos um exemplo de associação de um BOTÃO DE AÇÃO com procedimentos.

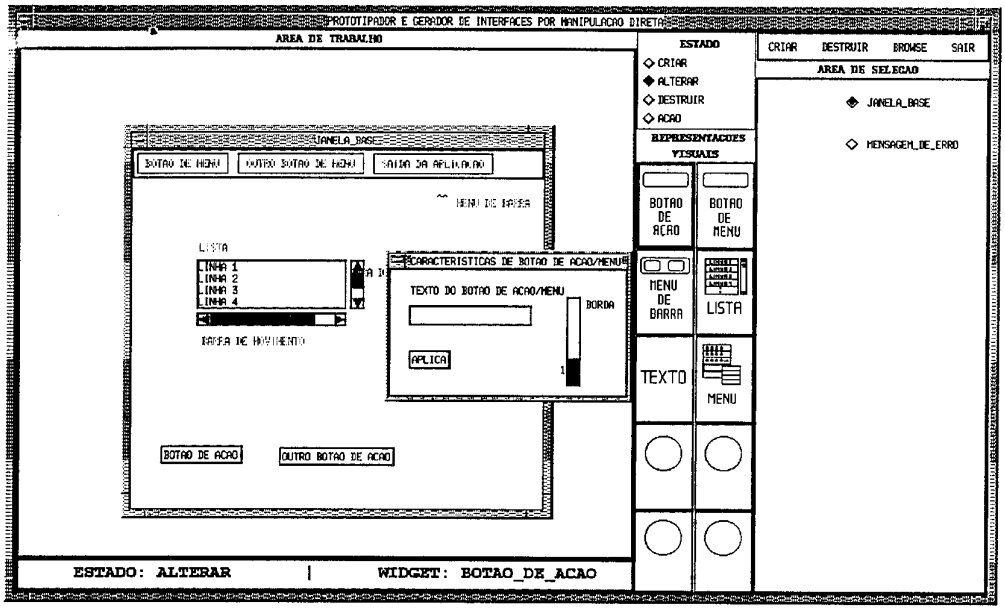


Figura 2.4 - Janela de Características de Botão de Ação

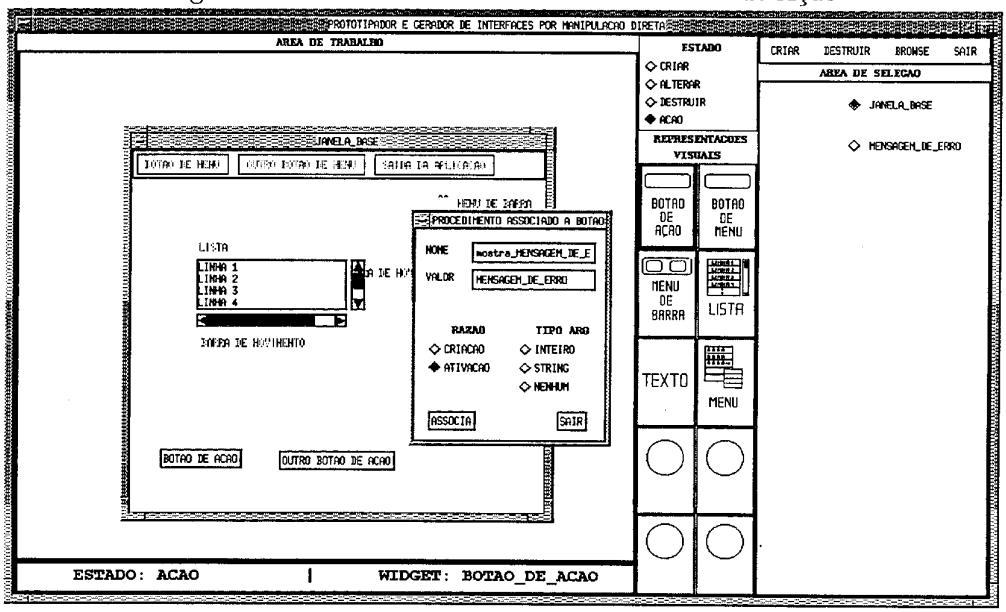


Figura 2.5 - Procedimento Associado à Botão de Ação

# 3 MODELO DE IMPLEMENTAÇÃO

Para materializar todo o trabalho explicado anteriormente, decidiu-se desenvolver um protótipo que utiliza as teorias estudadas e que de agora em diante, será chamado de INTERFÁCIL. Resolveu-se desenvolvê-lo em C e Motif (X Window), visto que, como já foi explicado, conseguimos atender a todos os requisitos do ambiente proposto com essas ferramentas.

O INTERFÁCIL tem como interface gráfica uma janela dividida em várias áreas onde o usuário pode criar janelas, estipular o estado (criação, alteração, destruição e associação de ações), escolher o elemento a ser trabalhado, dentre outros recursos. Isto é feito com o uso de manipulação direta e com a possibilidade de manutenção de um diálogo assíncrono com o ambiente.

A principal área é chamada de área de trabalho se destina ao desenvolvimento de uma determinada janela de uma interface, isto é, à construção de determinada janela de uma interface que pode ser composta com elementos de interfaces gráficas disponíveis no INTERFÁCIL.

Outra área serve para a escolha do elemento com o qual o usuário deseja trabalhar, é chamada área das representações visuais, pois é composta por vários botões de ação com representações visuais dos elementos que compõem o INTERFÁCIL. Esta maneira de representação é necessária já que trabalhamos com manipulação direta.

Outra área é a de estados, onde o usuário pode escolher o tipo de operação que deseja realizar com o elemento escolhido, isto é, compor este elemento com outro, alterar posição ou características desse elemento, retirá-lo de um determinado elemento e, por fim, associar o elemento a uma determinada ação da aplicação para qual a interface está sendo desenvolvida.

Além das áreas anteriores, existe a área de seleção de janelas, que é a área onde o usuário pode escolher determinada janela que compõe a interface para poder trabalhar sobre ela. O usuário pode manter diversas versões de uma mesma janela até escolher a que melhor atende suas necessidades.

Agregada a esta área existe um menu de barra com as opções de criação, destruição e demonstração destas janelas. Além dessas opções, existe a opção de saída do INTERFÁCIL, para a qual existem duas maneiras possíveis: uma chamada ARMAZENAR, que serve para guardar o trabalho do desenvolvedor e outra chamada FINALIZAR que além de guardar o trabalho desenvolvido, gera a interface da aplicação já compilada e fornece o programa padrão C para ser compilado. Será considerada como janela principal da aplicação desenvolvida aquela que estiver na área de trabalho no momento da finalização.

Entende-se por demonstração a possibilidade que o sistema fornece ao usuário de ver o funcionamento da interface simulando as ações que vão ocorrer depois que a interface estiver pronta. É importante frisar que essa demonstração se limita à parte visual da interface; nenhuma ação ligada à execução de procedimentos da aplicação a ser construída a partir da interface pode ser vista. Entretanto, é possível ver as relações entre as janelas da interface, isto é, o aparecimento ou desaparecimento de determinada janela da interface com uma ação do usuário sobre determinado elemento da interface, pois esta ação é implementada pela interface e não pela aplicação.

O modo de agir do usuário do INTERFÁCIL pretende ser o mais simples possível. Exige o conhecimento prévio dos elementos de interfaces gráficas que o compõem e baseia-se, predominantemente, na manipulação dos elementos pelo usuário através da utilização do "mouse".

Para construir a interface o usuário vai ter que começar pela criação de uma determinada janela indo ao menu de barra agregado à área de seleção de janelas. Com isso, uma representação desta janela aparecerá, e o usuário poderá colocar esta janela em posição na área de trabalho. A partir deste ponto o usuário poderá compor esta janela com os elementos que deseja. Para isto, ele primeiro deve escolher o tipo de elemento com o qual deseja trabalhar através de uma seleção simples sobre a representação visual deste. A indentificação do tipo do elemento a ser trabalhado aparecerá na parte inferior da área de trabalho. A seguir deve escolher o tipo de operação que deseja executar sobre este elemento. As operações, podem ser as seguintes (Figura 3.1):

- CRIAR : compor um novo elemento do tipo escolhido em outro. Para fazer isso o usuário só precisa entrar na área de trabalho e selecionar com o "mouse" a posição onde deseja colocar o elemento, o sistema fará a verificação da validade da posição e a validade da composição feita.
- ALTERAR : alterar a posição de um determinado elemento, e por conseguinte sua composição. O usuário tem que entrar na área de trabalho e selecionar o elemento a ser alterado através de uma seleção simples, e logo a seguir, através de outra seleção simples, posicioná-lo em outra posição. A alteração da posição de elementos compostos com outros, implicará na alteração dos elementos que o compõe. Neste estado o usuário pode alterar as características do elemento, como será explicado a seguir.
- DESTRUIR : para retirar determinado elemento da janela, basta o usuário selecionar o elemento a ser retirado na área de trabalho. A retirada de um elemento composto com outros elementos implicará na retirada destes outros elemento.
- AÇÃO : associar uma ação a determinado elemento a ser escolhido na área de trabalho através de seleção simples. Depois da seleção do elemento o usuário poderá escolher o tipo de ação que deseja associar ao elemento, podendo ser uma associação a um procedimento de uma aplicação, associação ao aparecimento de uma outra janela da interface, e outros tipos de ações que dependem do tipo de elemento selecionado.

Como já foi citado acima, quando um elemento é selecionado na área de trabalho quando o estado é [CRIAR] ou [ALTERAR], aparece na parte superior da tela uma JANELA DE CARACTERÍSTICAS associada a este elemento, onde o usuário poderá alterar, através de manipulação direta, as características desse elemento.

Uma capacidade importante do ambiente é que qualquer ação pode ser desfeita ("undo") na ordem contrária à que foi executada. O usuário pode desfazer seu trabalho do ponto onde se encontra até o início do seu trabalho.

Como foi dito anteriormente, o uso do INTERFÁCIL requer um conhecimento prévio pelo usuário dos elementos de interfaces gráficas que compõem o sistema. Para ampliar o espectro de usuários, foi desenvolvido paralelamente a esse trabalho um estudo sobre um sistema de ajuda para usuários do ambiente [BARC91] [BORE85] [BAUE87]. Esse sistema de ajuda compreende um sistema passivo que é ativado toda vez que o usuário comete um erro ou tem um mal entendimento sobre a funcionalidade do objeto ("misconception") [CARV88]. Baseia-se no aparecimento de uma janela que apresenta uma interpretação sobre o "erro" que ocorreu e suas possíveis soluções. Também compõe este sistema de ajuda, um modo de ajuda ativa, que compreende o aparecimento de uma janela de apoio que através da interpretação do estado do sistema, apresenta, além das informações sobre o elemento escolhido, as ações que o usuário pode tomar. Por exemplo: por que tipo de elementos o elemento escolhido pode ser composto.

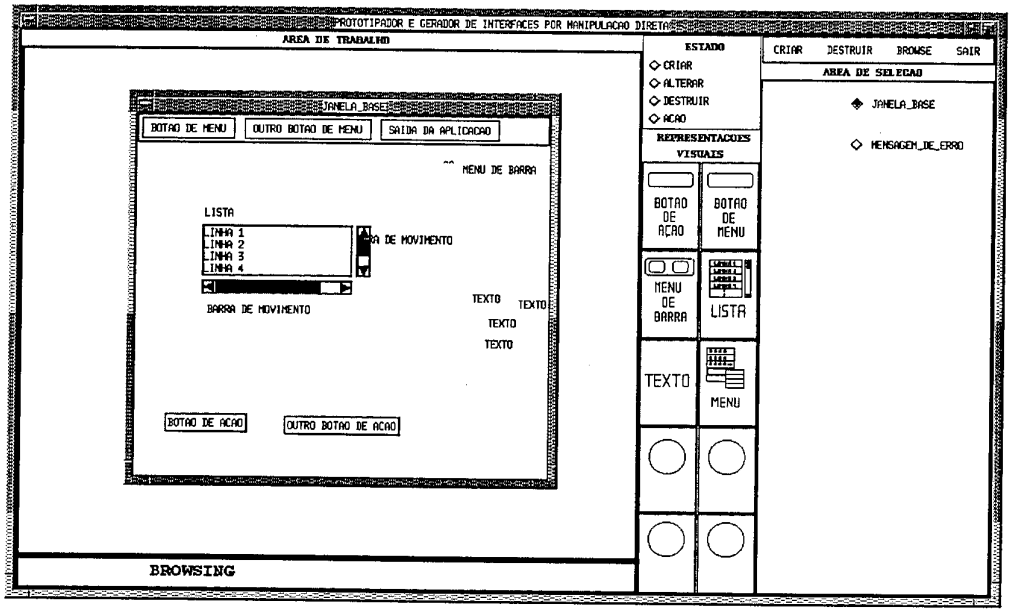


Figura 3.1 - Construindo a Interface

As manipulações que estes elementos vão sofrer, não são inerentes a eles e sim definidas pelo ambiente, já que, na realidade, manipula-se uma representação destes. Não se "caminha" com o elemento no ambiente na hora da criação; apenas indica-se "tocando" em uma representação deste como se quer trabalhar. Já na alteração, destruição e associação de ações, "toca-se" no elemento já posicionado em uma janela da interface.

No estudo das características destes elementos, algumas se destacaram como necessárias para o posicionamento dos elementos e de seus formatos, sendo estas usadas internamente pelo INTERFÁCIL e alteradas indiretamente pelo desenvolvedor. Outras, que definem principalmente as características visuais desses elementos, quando consideradas relevantes, foram elencadas para fazer parte da JANELA DE CARACTERÍSTICAS relativa a esse elemento (Figura 2.4).

Para satisfazer o requisito de diálogo assíncrono entre a interface do INTERFÁCIL e o desenvolvedor da interface, utilizou-se o conhecimento sobre o modo de ser do X Window, que serve de base para o Motif. O que nos interessa no X Window é o fato deste trabalhar

com um "loop infinito" que lê e despacha eventos.

Todas as ações sobre uma janela em um ambiente que roda sobre o X Window são consideradas eventos, e são despachados por meio de uma rede para procedimentos que são responsáveis pelo tratamento dessa ação.

Para informar ao Motif que se quer receber o controle quando um evento acontece, sabendo-se que o Motif guarda uma lista de procedimentos que devem receber um determinado evento, basta adicionar um procedimento a essa lista. Estes procedimentos são chamados de manipuladores de eventos. Como a especificação se baseia em um tipo de evento, que é o acionamento do "mouse" sobre determinado elemento, só precisamos informar ao Motif que queremos adicionar um procedimento à lista de procedimentos do evento "ButtonPress", sobre determinado elemento.

Seguindo o que foi especificado os manipuladores de eventos foram preparados para realizar tarefas completas e devolver o controle para o "loop infinito".

Cada manipulador, quando chamado, recebe como parâmetros o identificador do elemento onde o evento ocorreu e a estrutura Motif que representa este evento. Através de uma estrutura gerada na hora da criação de cada elemento, o manipulador, através do identificador do elemento, consegue saber seu tipo, se é composto por outros, se tem procedimentos associados, seu nome e outras informações. Já com o evento, consegue saber a posição na qual o mouse foi acionado.

A alteração de características no protótipo atual só é feita quando o desenvolvedor informa o fim das alterações, mas poderiam acontecer dinamicamente, isto é, um desenvolvedor ao alterar a largura de um elemento atuando sobre uma "escala", veria a alteração dinâmica da largura do elemento e no fim da alteração decidiria se quer manter ou não a alteração. Existe um outro meio de apresentar as alterações que um elemento vai sofrer quando tem suas características alteradas. Esta maneira é a representação desse elemento na própria JANELA DE CARACTERÍSTICAS, o que, no protótipo, acontece na janela de LISTAS onde a inclusão e deleção de elementos da lista que se quer alterar aparece numa lista na JANELA DE CARACTERÍSTICAS que, inicialmente, é igual à lista a ser alterada.

Na associação de ações, foram criadas JANELAS DE AÇÃO (Figura 2.5), que permitem ao desenvolvedor fazer a associação. Como essas janelas são acionadas pelos manipuladores de eventos, estes já conhecem o tipo de elemento, gerando, assim, uma janela intermediária que apresenta um elenco de tipos de associações que podem ser feitas com determinado elemento. Depois que a escolha é feita aparece a janela associada ao tipo de associação.

Para o tratamento do erro de composição de elementos foram criadas as JANELAS DE ERRO, onde se tenta dar maiores informações sobre os erros. Foi criado um procedimento que, de acordo com o elemento que sofreu o erro e o estado do sistema, apresenta um texto sobre o erro. Este texto, como já foi explicado, considera que os usuários tem conhecimento uniforme.

Para completar o INTERFÁCIL, foi necessário poder guardar o estado de uma interface em qualquer momento. Para tanto, tornou-se necessária a obtenção de um meio para fazer esse armazenamento. Para aproveitar o fato que como saída do ambiente teremos um arquivo com especificação UIL, decidiu-se criar um arquivo auxiliar em UIL que se diferencia em alguns pontos do arquivo que cria a interface quando pronta.

Este método de armazenamento nos permite utilizar o compilador UIL, chamado de dentro da aplicação, para recuperar o estado do trabalho anterior. A diferença básica entre o arquivo temporário e o final, são os procedimentos associados aos elementos, que, no arquivo temporário, são associaddos a criação e que, no final, dependem do tipo de associação.

Para gerar o programa padrão em C, existem procedimentos que criam as declarações básicas de toda a aplicação desenvolvida sobre o Motif e outros, que dependendo dos procedimentos associados aos elementos, faz a declaração destes (como exigência do Motif), criando em seguida, o corpo destes procedimentos. Esse corpo inclui algumas funções do Motif, que servem como "dicas" para o desenvolvedor na hora de incorporar a aplicação.

Deve-se frisar que, como em toda interface desenvolvida no Motif, a princípio, o controle é feito pela interface e a aplicação pode ser considerada um repositório de procedimentos, que são acionados pela interface. Ela realiza suas tarefas e devolve o controle para a interface. Então, interfaces desenvolvidas no ambiente, devem, a princípio, atuar dessa maneira. Entretanto, o Motif permite que a aplicação passe a controlar, acarretando a transferência do controle para a aplicação, implicando em que o INTERFÁCIL não terá meios de evitar que o desenvolvedor tenha o trabalho de atuar no controle da interface. Portanto, por mais completo que seja o ambiente, ele não será capaz de associar ações que não tenham nenhuma relação direta com ações da interface.

Neste ponto, vale a pena dar uma idéia de como ficou a estrutura do ambiente proposto, que também é a do protótipo, fazemos isso através da Figura 3.2, abaixo.

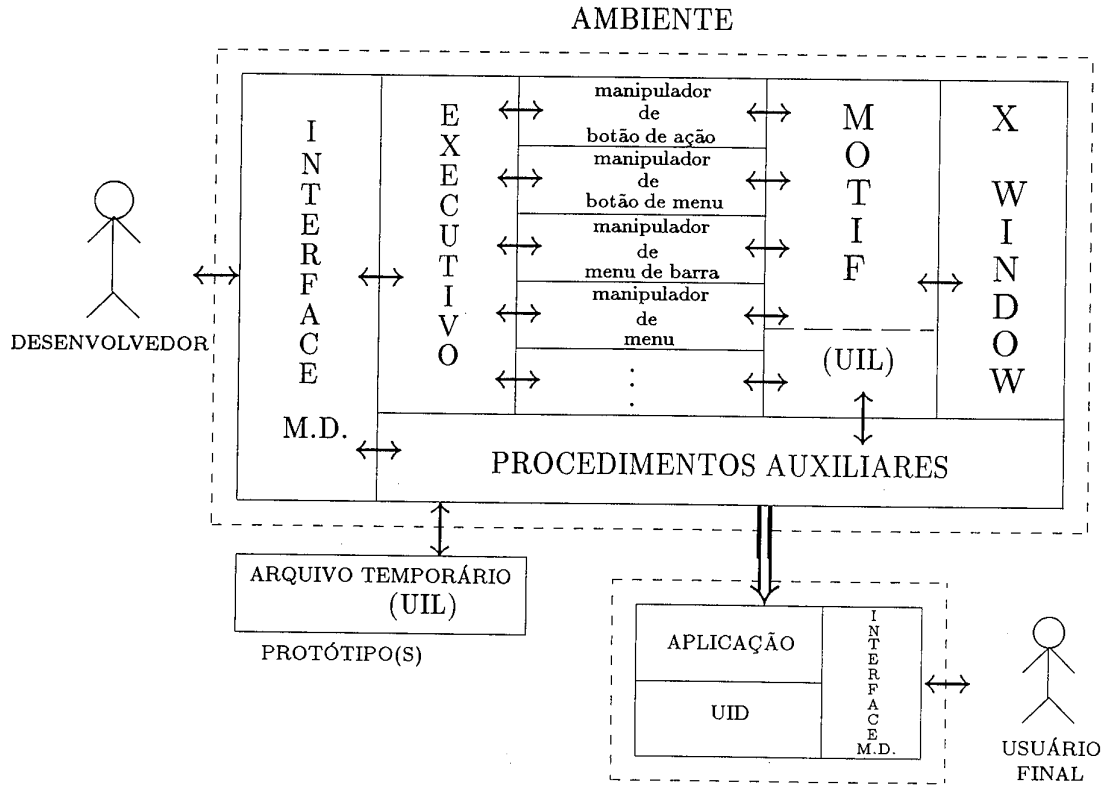


Figura 3.2 - Estrutura do Ambiente

# 4 ESTUDO DE CASO

Afim de analisar a eficiência do INTERFÁCIL, decidiu-se pelo desenvolvimento da interface de uma aplicação exemplo.

A análise foi feita sob o ponto de vista de um desenvolvedor que recebeu a tarefa de criação de uma interface para uma aplicação, da qual não se tem a especificação completa. Demonstraremos, passo a passo, as decisões do desenvolvedor sobre a interface e os efeitos destas na aplicação.

Depois que a interface estiver construída far-se-ão algumas considerações sobre o tempo que esta levou para ser concluída em comparação ao desenvolvimento da mesma interface diretamente no Motif, sem o auxílio do INTERFÁCIL.

## 4.1 APLICAÇÃO EXEMPLO

Para se ter parâmetros para comparações, escolheu-se desenvolver uma interface para um ambiente de desenvolvimento de sistemas especialistas baseado em regras de produção [BARB90], que já teve uma interface gráfica desenvolvida em Motif.

O ambiente, a partir da leitura de um arquivo onde estão as declarações de classes, atributos, valores e regras de produção (em uma gramática própria) permite testar o sistema especialista.

A tela inicial do ambiente deve poder acionar a leitura de uma base de conhecimento e permitir que esta seja utilizada. Outro requisito é que o ambiente possa demonstrar as classes que possui, os atributos de classes e os valores válidos para determinado atributo. Por isso, resolveu-se criar uma JANELA BASE com um MENU DE BARRA com os BOTÕES DE MENU com títulos "LEITURA", para a leitura de uma base; "TESTA", para executar a base; "CONSULTA" para a consulta de classes, atributos e valores; e "FIM" para sair da aplicação (Figura 4.1).

Ao se ler uma base com gramática própria, espera-se encontrar erros. Para isso, o sistema deve ter janelas de erro, Figura 4.2, que, neste caso, constarão de um TEXTO explicando o tipo de erro e um BOTÃO para continuar (identificando que o usuário leu a mensagem).

Ao se executar a base composta de regras de produção, surgirão pontos onde há a necessidade de se responder a uma pergunta, isto é, dar um valor a um determinado atributo. Sendo assim, pensamos em uma JANELA BASE onde, na parte superior, fica a pergunta acompanhada por uma LISTA para escolha do valor (já que existe um número restrito de valores).

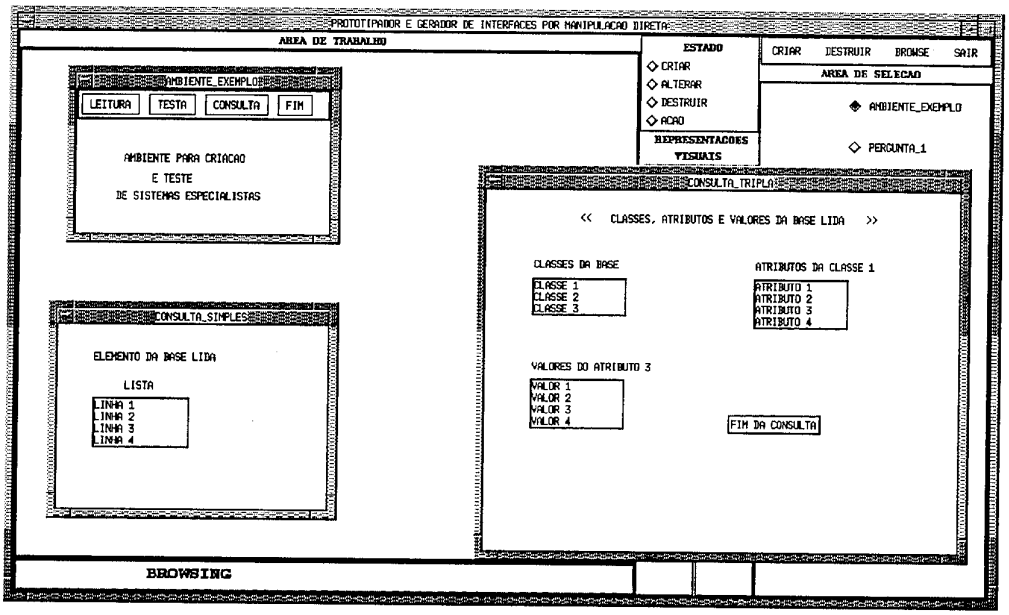


Figura 4.1 - Demonstrando a Interface da Aplicação Exemplo

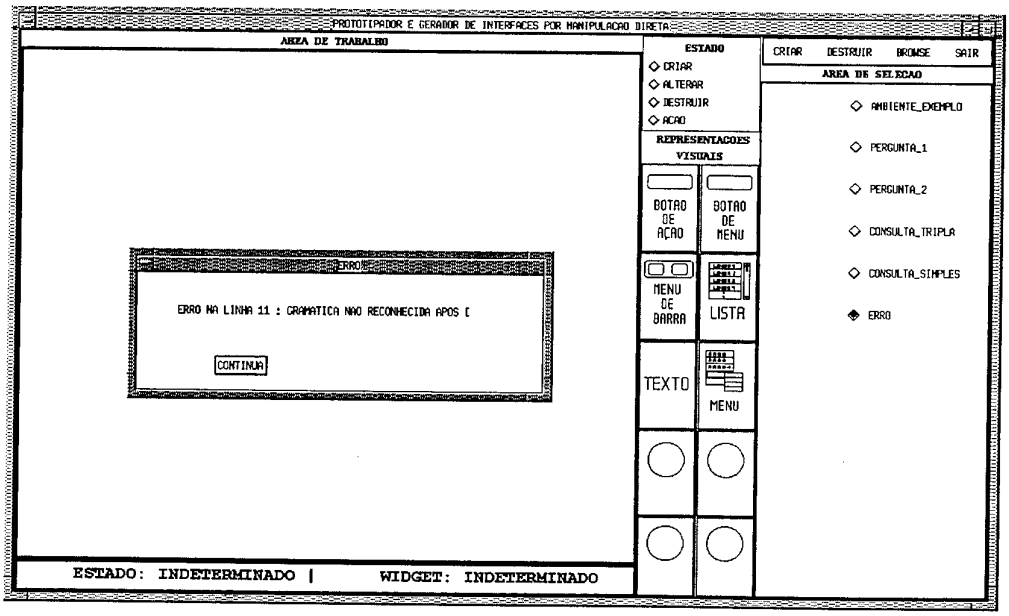


Figura 4.2 - Erro na Leitura da Base

Aqui temos um ponto a destacar. A escolha da resposta à pergunta ser feita diretamente na LISTA, implicou num maior número de enganos, isto é, o usuário mal acostumado com o esquema de listas, seleciona sem querer valores diferentes do que gostaria. Como a máquina de inferência que vai acionando as perguntas não pode voltar atrás, este perde tempo, tendo que voltar ao início para responder corretamente aquela pergunta. Para solucionar esta questão foi adicionado um BOTÃO DE AÇÃO com o título "RESPOSTA", que serve como meio de confirmação da escolha feita na LISTA. Na Figura 4.3 vemos o resultado.

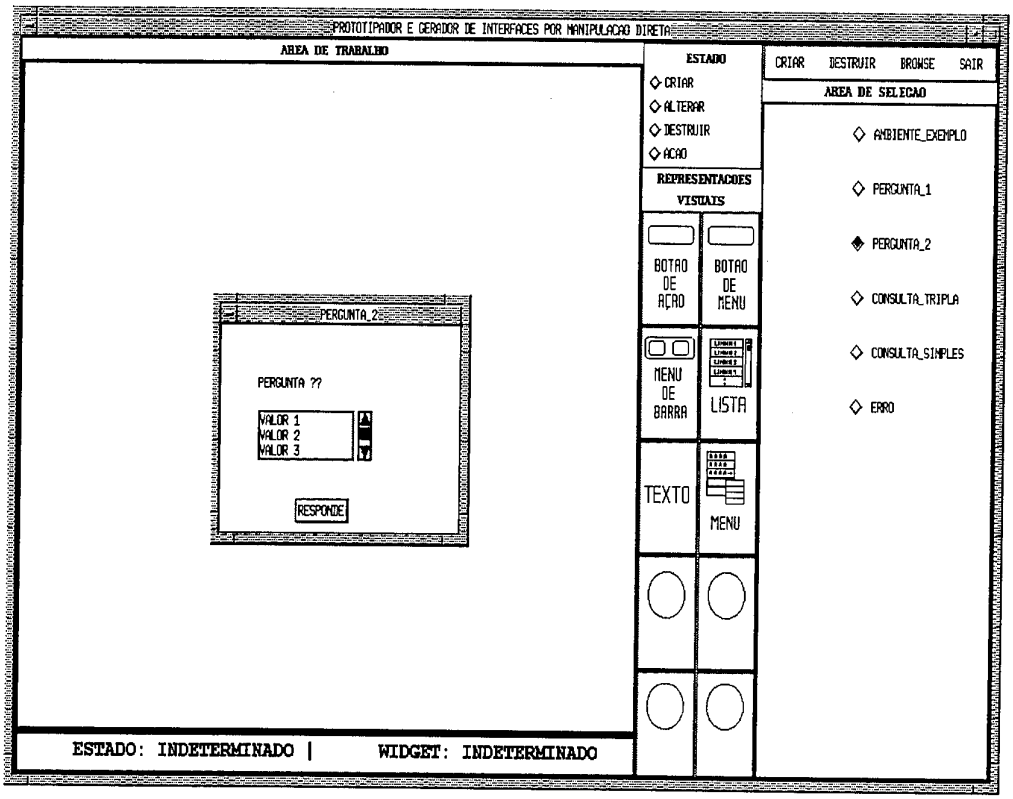


Figura 4.3 - Pergunta durante a Inferência

Já que a base do sistema especialista, lida pelo ambiente, é composta de classes, atributos e valores, torna-se importante um meio de se poder mostrar esses elementos. A princípio pensou-se numa JANELA BASE composta de três LISTAS onde a primeira conteria as classes declaradas na base, a segunda conteria os atributos da classe escolhida e a terceira os valores do atributo escolhido. A princípio as duas últimas listas estariam vazias, dependendo de escolha prévia.

Verificou-se que a criação de uma única JANELA BASE, com três listas, implicava em que só uma classe poderia ser avaliada por vez. Resolveu-se então criar um padrão de JANELA BASE, como na Figura 4.4, que conteria na parte superior, um TEXTO (por exemplo, "CLASSES DA BASE") e uma LISTA, que poderia conter nomes de classes,

atributos ou valores. Então, o simples fato de se escolher uma classe na LISTA de classes implica no aparecimento de outra JANELA BASE, deste mesmo tipo, contendo os atributos desta classe, e assim por diante. Pode-se assim visualizar várias classes ao mesmo tempo.

Outra opção avaliada, foi transformar o BOTÃO DE MENU de título "CONSULTA" em um MENU com três BOTÕES DE MENU que teriam os títulos, "CLASSES", "ATRIBUTOS" e "VALORES", como na Figura 4.5. A escolha de classe, neste caso, implica em trabalhar no esquema de JANELAS BASE padrão descrito anteriormente, e a escolha de atributos e valores acionam JANELAS BASE com LISTAS onde se pode ver todos atributos da base, independentes de classes, e todos os valores independentes dos atributos.

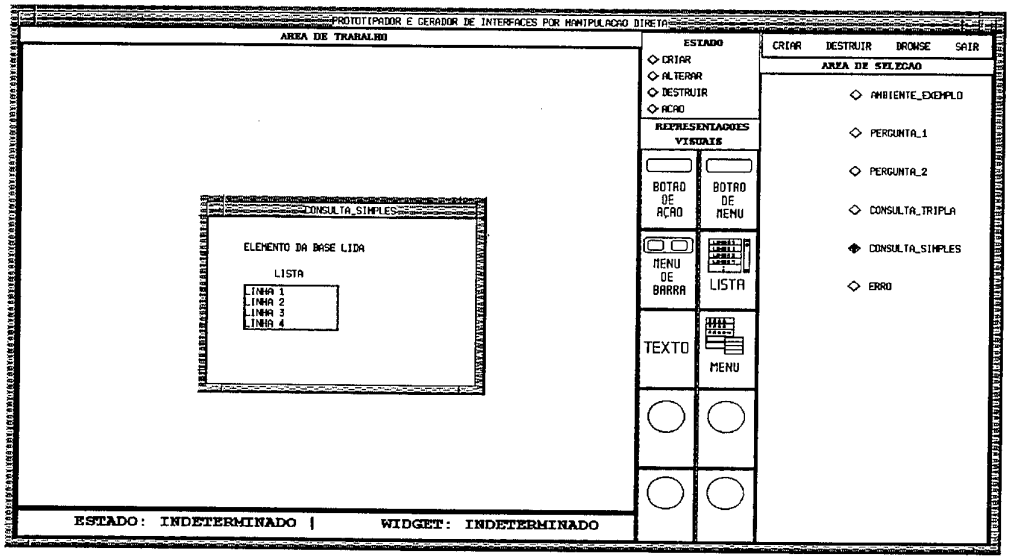


Figura 4.4 - Janela de Consulta

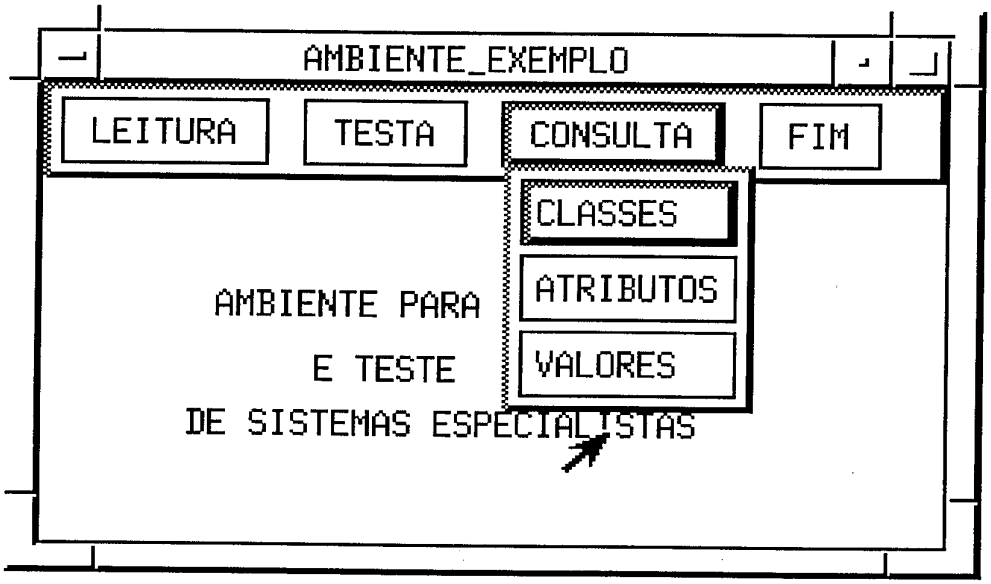


Figura 4.5 - Tela Principal da Aplicação com Menu

## 4.2 CONCLUSÕES SOBRE O ESTUDO DE CASO

Fica claro (como no caso das lista nas perguntas) que não se precisa ter toda a especificação da aplicação em mente para construir a interface, já que novos fatos e condições são facilmente incorporados à interface.

O tempo de desenvolvimento da interface acima descrita não ultrapassou 20 minutos, tempo muitíssimo inferior ao do desenvolvimento da interface real do ambiente. É importante frisar que as comparações de tempo entre a interface exemplo e a interface real, são feitas com base nas composições de elementos feitas para a construção da interface, ou seja, sem contar o tempo consumido pelo pouco conhecimento do Motif, na época, para o desenvolvimento da interface real.

O fato de se poder visualizar a colocação de TEXTOS e LISTAS nas JANELAS BASE economizou um tempo enorme de compilações que foi gasto no desenvolvimento da interface real para o posicionamento desejado dos elementos. Isto sem contar os erros de digitação que implicaram em novas compilações, só para mover uma LISTA alguns "pixels" na tela.

Outra economia de tempo e de trabalho que se pode destacar está na criação de várias versões para determinadas telas. O tempo que se levava na criação da interface real para produzir os elmentos na JANELA BASE já existente (que implicava na inclusão do elemento na especificação de outro sem erros de digitação) e no remanejamento de outros elementos com base em uma escala de "pixels", retirava o ânimo de se experimentar novas opções. Alie-se a isto o fato que o mais importante era a aplicação. Já neste ambiente, a interface real poderia ter sido muito mais elaborada sem a perda do tempo adicional.

Com relação ao aspecto visual do funcionamento da interface, temos que constatar que no desenvolvimento da interface real, que foi desenvolvida para uma aplicação já pronta, não se podia ver como as janelas do sistema apareceriam a não ser que a aplicação estivesse fechada, isto é, funcionando naquela determinada parte. Já com os meios que o ambiente nos fornece para a demonstração da interface, uma decisão que na interface real, foi tomada depois da aplicação e interface estarem prontas (alteração das larguras das listas e do número de itens visíveis para que estas pudessem existir em um maior número possível sem "engarrafar" a tela) pode ser avaliada e testada quase que imediatamente, posteriormente à criação das janelas.

Pontos como os descritos acima são importantes, pois poderiam implicar no remanejamento de vários elementos, na recompilação da especificação em UIL, e até alteração da aplicação. Isto não acontece se a interface pode ser testada, como no INTERFÁCIL, a qualquer momento, independente da aplicação.

Apesar de se ter dito que as comparações de tempo se limitariam ao desenvolvimento da interface, não se pode deixar de considerar o tempo gasto, no caso da interface real com o estudo do acionamento do "toolkit" do Motif, das declarações dos procedimentos associados à interface, da descoberta de funções para o tratamento de LISTA, dentre outros. Este tempo seria reduzido a zero, se a interface tivesse sido desenvolvida no INTERFÁCIL, pois o programa padrão em C contém todas as declarações de procedimentos, todas as inicializações e ainda fornce "dicas" para o uso dos elementos nos corpos dos procedimentos a eles associados.

# 5 CONCLUSÕES

De acordo com o nosso objetivo principal, que era o de mostrar a viabilidade da especificação de uma aplicação a partir da especificação de sua interface, verificou-se neste trabalho que essa hipótese é muito válida. Ao se usar o protótipo desenvolvido (INTERFÁCIL) para criar a interface da aplicação exemplo, verifica-se que, apesar do conhecimento prévio que um determinado desenvolvedor possa ter sobre a aplicação a ser desenvolvida, vários "caminhos novos" aparecem durante a criação da interface.

Pode-se afirmar, sem dúvidas, que o desenvolvimento de interfaces, principalmente gráficas, é uma etapa fundamental do desenvolvimento de software e que ambientes como o proposto devem constar em ambientes de desenvolvimento de software.

O uso de interfaces gráficas, como as adotadas no ambiente, permite uma comunicação maior e uma melhor apresentação de uma aplicação. Entretanto, deve-se notar que o uso de interfaces gráficas exige do usuário a capacidade de interpretar as representações visuais utilizadas, e que, dependendo do domínio da aplicação, estas são muito orientadas para um tipo de usuário específico, tornando-se imcompreensíveis para usuários não acostumados com o domínio. Neste caso, uma interface mais simples, que não permita tanta liberdade, pode ser mais conveniente.

Com as características que o sistema apresenta, o usuário pode, facilmente, testar todos os caminhos que desejar, visualizando-os imediatamente de uma maneira fácil, pois além de estar usando a manipulação direta, tem a vantagem de poder desfazer e depois refazer todo um caminho desenvolvido, passo a passo.

Outro fato que pode ser identificado é que o usuário não precisa ter um conhecimento profundo sobre o que pretende desenvolver, pois ao trabalhar na interface associando a cada elemento gráfico desta uma tarefa a ser realizada por sua aplicação, identifica praticamente toda a funcionalidade da aplicação.

O ambiente, por apresentar resultados imediatos, permite ao desenvolvedor sentar ao lado de usuários que vão utilizar a interface gráfica que se está projetando, e, junto, melhorar a interface para atender melhor às necessidades cognitivas dos usuários.

Apesar do INTERFÁCIL conter um esquema para reverter as ações ("undo"), que, dependendo do seu nível de desenvolvimento, pode se tornar muito complexo, descobriu-se que por se estar trabalhando com manipulação direta, o mesmo trabalho feito pelo sistema de "undo" pode ser realizado diretamente pelo usuário sem muito trabalho. Conclui-se então que esse esquema de reversibilidade de ações pode estar limitado a inversão de ações como criação e deleção de elementos, sem perda de funcionalidades.

Ao se limitar o trabalho do desenvolvedor a um simples acionamento de "mouse" para a composição de elementos, torna-se a tarefa de criação muito simples e rápida. Ao se separar a alteração de características e a associação de ações, tratando-as em janelas separadas que não precisam ser acessadas pelo usuário, dá-se uma liberdade muito maior ao desenvolvedor, pois este decide se vai alterar as características de um elemento ou se vai lhe associar uma ação. Se tal não ocorresse, o desenvolvedor, para ter diversas visões de sua interface, teria um trabalho muito maior, mesmo quando só quer ver como os elementos ficariam compostos, sem preocupação de nomes e outras coisas.

A associação de janelas de erro, acionadas quando o usuário comete um erro, torna o ambiente muito mais comunicativo, isto é, permite ao usuário conhecer muito mais

sobre o tipo de interface que está criando e amplia rapidamente sua visão, aumentando as alternativas de interface que ele tenha na mente.

A utilização de linguagens de especificação de interfaces gráficas, como a UIL, permite uma flexibilidade muito maior na criação de ambientes como o proposto, pois fornece um meio de armazenar a especificação temporária da interface, que é facilmente recuperável pelo sistema. Caso o ambiente fosse desenvolvido com base no "toolkit" do Motif, teria de haver a criação de uma linguagem de especificação e um compilador para poder transformar essa especificação em uma interface, sendo o trabalho muito maior.

Toda a riqueza visual do ambiente está na capacidade da composição dos elementos dinamicamente, isto é, ao se estabelecer uma composição, consegue-se visualizar, imediatamente, o elemento novo. Por isso, conclui-se que é fundamental a utilização de uma coleção de elementos de interfaces gráficas, como os do Motif, que possam ser inseridos em outros com visualização imediata. Também, torna-se relevante a estruturação desses elementos em classes, pois o simples conhecimento da hierarquia utilizada facilita a construção de interfaces.

## 6 COMPARAÇÕES COM OUTROS AMBIENTES

Com o objetivo de melhor situar o ambiente proposto, faz-se comparações com outros ambientes: Jacob [JACO86], Peridot [MYER86], BLOX Graphics Builder [HART89] e o Guide [SUNM90]. Como vemos adiante o sistema com maiores semelhanças com o nosso trabalho é o GUIDE:

- Jacob, apresenta um aplicativo desenvolvido na sua teoria. Apesar desta teoria, baseada em diagramas de estado "ampliados", servir para a construção de interfaces por manipulação direta, Jacob não se preocupa com a possibilidade de se poder construir o mesmo aplicativo por manipulação direta. Não se preocupa com as necessidades do usuário, limitando-se a proposição de uma linguagem de especificação.

- Peridot é um sistema desenvolvido para criação de interfaces gráficas, baseadas em menus, por manipulação direta. Peridot é uma proposta de ambiente que auxilia a criação dos menus, analisando as escolhas do desenvolvedor feitas através de comandos em uma janela, e através da aplicação de regras internas que permitem a continuação do trabalho pelo desenvolvedor. Mantém um diálogo sequencial, pois não permite que o desenvolvedor abandone a criação de um menu. Gasta tempo pedindo autorização para a aplicação de regras, para reunir trabalhos que por manipulação direta, seriam mínimos. O uso de conhecimento de projeto de interfaces em um ambiente é muito válido, mas não a esse ponto.

- BLOX Graphics Builder apresenta um ambiente bastante similar ao nosso, além de fornecer um esquema de ajuda na utilização dos elementos que compõem a interface, que pode ser incorporada ao sistema que se está desenvolvendo. Apesar de permitir a demonstração da interface independente da aplicação, e fornecer rotinas a serem adicionadas à aplicação, exige uma ligação direta da aplicação à interface, isto é, a alteração de qualquer coisa na interface implica no "relink" à aplicação. Com a

utilização de um módulo separado, como no nosso ambiente, pode-se fazer alterações na interface independente da aplicação.

- GUIDE é um ambiente de criação de interfaces gráficas do tipo OpenWindows. Utiliza manipulação direta e para cada grupo de elementos do mesmo tipo apresenta uma janela onde as características podem ser alteradas e procedimentos podem ser associados. Nesta janela apresenta também uma lista dos elementos do tipo escolhido. Depois que o usuário acaba de compor a interface, um arquivo .G é criado, contendo a descrição da interface em GIL (Graphical Intermediate Language Syntax). Esta não é orientada a objetos, e associa uma lista fixa de atributos por elemento, gerando arquivos grandes.
  Depois que o arquivo .G é gerado o usuário deve compilá-lo com o aplicativo "gxv", que gera como saída 4 arquivos: um com o código fonte primário, inicialização e funções de criação para os elementos da interface (em C e XView); outro com as declarações e definições de tipo do arquivo anterior; outro com os padrões das "callbacks procedures" que foram associadas aos elementos da interface e por fim um arquivo que contém textos de ajuda sobre os elementos da interface.
  Como foi dito acima o GUIDE permite a associação de um texto de ajuda à cada elemento da interface. Permite também a associação de ações da aplicação à elementos da interface através de "callbacks procedures", mas não apresenta meios de associar ações de interface (como o aparecimento de outras janelas e outras já explicadas neste trabalho), impossibilitando assim a simulação do funcionamento da interface. Quando se executa a primeira versão do programa gerado depois que o "gxv" trabalha sobre o arquivo .G, todas as janelas aparecem ao mesmo tempo não possibilitando a demonstração de um fator importante da interface, que é o relacionamento entre as janelas. Isto transfere o trabalho para o usuário dentro dos arquivos em C gerados, aumentando o conhecimento necessário do usuário sobre o "toolkit" no qual o GUIDE se baseia.
  O GUIDE na parte da composição das janelas da interface não trata da criação de menus por manipulação direta, forçando o usuário a digitar os itens que vão compor os menus e seus subitens, para, na simulação, apresentar o menu. No arquivo em C que gera não apresenta nenhuma "dica" sobre os elementos da interface em termos dos procedimentos associados.

## 7 TRABALHOS FUTUROS

Como já foi discutido, o caminho futuro a ser seguido para a evolução desse ambiente é prepará-lo para fazer parte de um ambiente maior, um ambiente de desenvolvimento de software, onde o desenvolvimento da interface de uma aplicação possa ser ou a etapa inicial da especificação de um software ou mais uma das etapas da especificação.

Para tornar o ambiente atual mais completo, deve-se aumentar a coleção de elementos usada no estudo de caso, ampliar todas as janelas de características e de associação de ações, e transformar o esquema de ajuda da forma atual para a forma de hipertexto.

Para incorporar o ambiente a um ambiente maior deve-se prepará-lo para a possibilidade de que partes da aplicação já estejam desenvolvidas e que o desenvolvedor queira

incorporá-la ao seu trabalho.

Outra melhoria que pode ser acrescentada no ambiente seria a inclusão de um esquema de armazenamento de conhecimento de projetos de interfaces gráficas aliado a um meio de ajuda muito superior, que poderia até tomar decisões pelo usuário. Seria possível mesmo gerar automaticamente a interface do usuário, partindo-se do conhecimento das expectativas do usuário, obtidas através de alguma forma de diálogo.

Outro ponto seria a adaptação do ambiente para trabalhar em ambientes cooperativos, onde vários desenvolvedores estariam trabalhando no projeto de um mesmo software, e, no caso específico do ambiente, queiram criar versões dessa interface e compartilha-las com os outros desenvolvedores. O resultado final pode exigir que a interface definitiva seja construída de partes desenvolvidas por usuários diferentes, resolvendo-se todas as pendências que surgirão neste processo.

Também seria válida a alteração do ambiente para permitir a criação de interfaces para mais de um tipo de ferramenta de criação de interfaces gráficas, como é o caso atual (Motif). Isto implicaria no desenvolvimento de uma estratégia de nível superior, pois o ambiente teria que se adaptar, alterando suas representações visuais e seu modo de ser, em função do tipo de ferramenta escolhida. Neste caso, para não se tornar um verdadeiro "monstro" em tamanho, ele teria que trabalhar com uma representação única da interface. Esta representação teria de ser suficiente para a geração da interface de acordo com a ferramenta escolhida, e permitir ao ambiente atuar como se estivesse preparado especificamente para determinada ferramenta.

Um ambiente reunindo tudo o que foi feito, com o que foi proposto como direção futura de desenvolvimento, seria muito poderoso, pois, além de praticamente criar a interface para uma determinada aplicação, prepararia esta aplicação para rodar em tantos ambientes quantas fossem as representações de ferramentas que possuísse. Isto supõe que a aplicação é portável para os ambientes das ferramentas de interfaces com que se trabalharia.

## BIBLIOGRAFIA


[BARB90] - Barbosa, Luís Fernando D. J.; "EXPERT: Um Ambiente para Sistemas Especialistas"; trabalho desenvolvido como Projeto Final de Programação do Mestrado, Departamento de Informática, PUC/RJ, 1990.2.

[BARC91] - Barcellos, Marta R.; "Um Sistema de Ajuda para um Gerador de Interfaces Gráficas", trabalho de Iniciação Científica orientado pelo Prof. Carlos José P. de Lucena, Depto de Informática, PUC/RJ, Julho de 1991.

[BAUE87] - Bauer, Joachim; Schwab, Thomas; "Propositions on Help-Systems", friedr. Vieweg & Sohn Verlagsgesellschaft mbH, Maio de 1987.

[BORE85] - Borenstein, Nathaniel S.; "The Design and Evaluation of On-line Help Systems", Tese de Doutorado do Departamento de Ciência da Computação da Universidade Carnegie-Mellon, Abril 1990, pg 55.

[CABR90] - Cabral, R. H. B.; Campos, I. M.; Cowan, D. D.; Lucena, C. J. P.; "Interface as Specifications in the MIDAS User Interface Development System", ACM SIGSOFT, vol. 15, no 2, Abril 1990, pg 55.

[CARV88] - Carvalho, Eliana B. S. de; "Tópicos em Inteligência Artificial II", Departamento de Informática, PUC/RJ, 1988.

[EHRI81] - Ehrich, R. W.; Hartson, H. R.; "DMS: An Environment for Dialogue Management", In proceedings of COMPCON81, IEEE, Setembro de 1981, pg 121.

[GREE86] - Green, M.; "A Survey of three dialog models", ACM Trans. Graph. 5, 3 (Julho 1986), 244-275

[GUAR89] - Guaranys, P. Y.; Lucena, C. J. P.; "PUC: A Knowledge Based Environment for Planned User Communication"; Proceedings of the Computer Software and Applications Conference, IEEE, COMPSAC89, 1989.

[JACO86] - Jacob, R. J. K.; "A Specification Language for Direct Manipulation User Interfaces", ACM Transactions Graphics, vol.5, no 4, Outubro 1986, pg 283-313.

[HART89] - Hartson, H. R.; Hix, D.; "Human-Computer Interface Development Concepts and Systems for its Management", ACM Computing Surveys, vol. 21, no 1, Março 1989.

[HAYE89] - Hayes, F.; Baran, N.; "A Guide to GUIs", BYTE, Julho 1989.

[LEIT91] - Leite, Jair C.; "A Interação entre usuários e Sistemas de Computadores em Linguagem Natural Orientada por Menus", Dissertação de Mestrado apresentada ao Departamento de Informática da PUC-RJ, Setembro 1991.

[MYER86] - Myers, B. A.; Buxton, W.; "Creating Highly-Interactive and Graphical User Interfaces by Demonstration", ACM SIGGRAPH 86, vol. 20, no 24, Agosto 1986.

[NAUR63] - Naur P.; "Revised report on the algorithmic language ALGOL 60", Commun. ACM 6, Janeiro 1963.

[OSFM90] - Open Software Foundation; "OSF/MOTIF Programmer's Guide", Revision 1.0, Prentice Hall, 1990.

[SCHE86] - Scheifer, R. W.; Gettys, J.; "The X Window System", ACM Transactions Graphics, vol.25, no 2, Abril 1986, pg 79-109.

[SHNE83] - Shneiderman, B.; "Direct Manipulation: A Step Beyond Programming Languages", IEEE Computer 16, Agosto 1983, pg 55-69.

[SIBE86] - Sibert, J. H.; Hurley, W. D.; Blesser, T. W.; "An Object-Oriented User Interface Management System", Computer Graphics SIGGRAPH 86 Conference Proceedings, vol. 20, no 4, Agosto 1986.

[SUNM90] - OpenWindows Developer's Guide - 1.1, Sun Microsystems Inc., 1990.

[OLSE86] - Olsen, D. R. JR.; "Mike: The Menu Interaction Kontrol Environment", ACM Transaction GRAPHICS, vol. 5, no, Outubro 1986, pg 318-344.