UMA EXPERIÊNCIA EM PORTABILIDADE DE INTERFACES GRÁFICAS

AN EXERCISE IN GRAPHICAL USER INTERFACES PORTABILITY

Miguel ARGOLLO JR.*
Simone SCARPELINI**

ABSTRACT

During the last few years Graphical User Interfaces (GUI for short) became a key factor for the success of a software system in almost any application area. Software portability has always been a great concern for developers, and GUI portability became a reality with multi-platform GUI design tools; however, these tools are quite new for the Brasilian software industry. This work describes an experience in developing portable GUI with such a tool, XVT.

KEY WORDS: Graphical User Interfaces (GUI), portability.

RESUMO

Nos últimos anos interfaces gráficas (GUI) tornaram-se um fator chave para o sucesso de um sistema de software em praticamente qualquer domínio de aplicação. Portabilidade de software sempre foi uma preocupação para os desenvolvedores, e a portabilidade de interfaces gráficas tornou-se uma realidade com ferramentas para o projeto de interfaces gráficas multi-plataformas; entretanto, a utilização das mesmas é bastante recente para a indústria nacional de software. Este trabalho descreve uma experiência no desenvolvimento de interfaces gráficas portáteis com uma ferramenta desse tipo, XVT.

PALAVRAS-CHAVE: Interfaces Gráficas (GUI), portabilidade.

1 - INTRODUÇÃO

Desde o final da década de 80 o mercado de software passou a fazer uma forte pressão para que os aplicativos fossem desenvolvidos com interfaces gráficas. Esta pressão foi motivada tanto pela disponibilidade de computadores pessoais e estações de trabalho com capacidade gráfica a um preço acessível quanto pelo surgimento de poderosos ambientes gráficos.

Uma das principais vantagens de aplicativos com interfaces gráficas sob o ponto de vista de seus usuários é o fato de apresentarem aparência e funcionamento (**look-and-feel**) consistentes. Essa consistência não é obtida automaticamente, e sim através do uso de padrões, que podem ser públicos ou proprietários e que definem tanto os elementos gráficos das interfaces quanto o funcionamento dos mesmos.

Um padrão proprietário depende unicamente da companhia que o desenvolveu, e normalmente se encontra associado a um único ambiente operacional. Os sistemas MS-WINDOWS, IBM Presentation Manager e Macintosh são os exemplos de padrões proprietários mais utilizados atualmente.

Um padrão público não se encontra sobre o controle de uma única companhia, sendo sua evolução normalmente controlada por um conjunto de empresas ou um grupo de usuários. O sistema OSF/MOTIF, o padrão público de maior aceitação nesta área, foi definido pela Open Software Foundation, um consórcio formado pelos principais fornecedores de sistemas Unix.

Embora o sistema MS-WINDOWS domine o cenário nacional, em nível internacional outros ambientes possuem uma fatia considerável do mercado. A dinâmica da área de informática, com o surgimento de novas

(*) Matemático (mod. Informática) pela UFRJ (1980) e mestre em Engenharia de Sistemas e Computação pela COPPE/UFRJ (1984). Atualmente chefe da Divisão de Engenharia de Software do Instituto de Computação da Fundação Centro Tecnológico para Informática - CTI.
(**) Engenheira de Computação pela Unicamp (1992). Atualmente Analista de Software prestando serviços de reengenharia de sistemas no Bamerindus.

plataformas de software e hardware, bem como a possibilidade dos produtores nacionais passarem a disputar o mercado internacional, transforma o desenvolvimento de aplicações multi-plataformas em um mecanismo gerador de novos negócios. Um produtor de software que domine a tecnologia necessária para oferecer seus produtos em várias plataformas sem onerar substancialmente seus custos de desenvolvimento possui uma importante vantagem competitiva em suas mãos.

As áreas de linguagens de programação, sistemas de bancos de dados e sistemas de comunicações vêm sendo objetos de esforços de padronização há algum tempo e os padrões obtidos, embora com um uso ainda incipiente, são uma ferramenta poderosa para o desenvolvimento de software multi-plataforma [IEE93a]. Entretanto, o desenvolvimento de interfaces gráficas portáteis ainda é considerado, por boa parte dos produtores de software, como um desafio tecnológico a ser superado. Por exemplo, uma pesquisa sobre experiências práticas no desenvolvimento de interfaces gráficas publicada em 1992 mostrou que somente 3 dos 74 projetos que responderam à pesquisa tinham sido planejados para serem transportados para múltiplas plataformas [MR92].

Este artigo reflete a experiência do CTI na utilização de uma ferramenta empregada para produção de interfaces gráficas portáteis e está organizado na seguinte forma: a próxima seção apresenta o escopo no qual o trabalho foi realizado; a 4ª seção apresenta a arquitetura da ferramenta utilizada e o fluxo de desenvolvimento de um programa de computador através de seu uso; as duas seções seguintes apresentam os dois módulos principais da ferramenta, o editor de interfaces gráficas e a biblioteca multi-plataforma; o artigo termina com 2 seções que apresentam os resultados dos testes realizados e as conclusões obtidas.

Embora [BA] e [HS91] apresentem métodos de avaliação de ferramentas de suporte ao desenvolvimento de interfaces gráficas, estas referências não tratam do problema de portabilidade das interfaces obtidas, de modo que o enfoque adotado para o desenvolvimento do trabalho reportado aqui é mais pragmático e orientado para as necessidades de produção de software portátil do Programa de Qualidade e Produtividade em Software - PQPS.

2 - PROGRAMA DE QUALIDADE E PRODUTIVIDADE EM SOFTWARE - PQPS

O Programa de **Qualidade e Produtividade em Software do CTI - PQPS** - é composto por um conjunto coordenado de ações que visa a aquisição, o domínio e a disseminação de métodos, técnicas, critérios e ferramentas integrados em ambientes de suporte à produção de software. Essas ações cobrem duas grandes áreas: produção de software portátil e avaliação de produtos e processos de desenvolvimento de software.

No campo de avaliação está sendo desenvolvido o projeto **Tecnologia para Avaliação da Qualidade de Software** - TAQS, cujo objetivo básico é o de desenvolver métodos para a avaliação da qualidade de produtos de software e do processo de desenvolvimento de software, além de instrumentos práticos para a sua aplicação ampla e facilitada. Desta forma, pretende-se aparelhar um avaliador com recursos para julgamento do nível de qualidade de software em relação a critérios e padrões internacionalmente reconhecidos.

Numa primeira fase, o projeto está estruturado para a geração de tecnologias para a avaliação de produtos de software - TAQS-PROD, onde se pretende atingir os seguintes pontos:

. especificação de métodos, expressos em um manual do avaliador, contendo procedimentos para o desenvolvimento de um processo de avaliação e para estimar os custos do processo;

. criação do Ambiente para Avaliação da Qualidade do Produto de Software - AAQPS, composto por:

1. Sistema de apoio à Especificação e ao Planejamento da Avaliação;
2. Sistema de gerenciamento de Bibliotecas de Módulos de Avaliação, Métricas e de um Banco de Dados Histórico;
3. Módulos de Avaliação para integrar a Biblioteca correspondente, compostos de procedimentos manuais, auxiliados por computador, ou automáticos.

Todo o ambiente deverá utilizar, sempre que possível, tecnologias já existentes, como ferramentas já comercializadas, e padrões já aceitos. Dentro deste contexto, atividades de pesquisa que colaborem para o aprimoramento e evolução deste ambiente e que ampliem a disponibilidade tecnológica deverão ser favorecidas.

No campo de produção de software portátil procura-se desenvolver técnicas para a produção de aplicativos que possam estar disponíveis em ambientes operacionais distintos através da utilização de padrões que forneçam a base sobre a qual os aplicativos são desenvolvidos. Tais padrões podem ser classificados segundo um modelo de referência, que deve funcionar como uma ferramenta de trabalho para os projetistas dos aplicativos,

oferecendo, para cada requisito do aplicativo, um componente que forneça mecanismos e serviços para o seu desenvolvimento.

Atualmente existem poucos modelos propostos, entre os quais se destaca o do grupo de trabalho da IEEE P1003.0, chamado OSE (**Open System Environment**) [IEE93a]. Este modelo está dividido em cinco componentes principais, aqui apresentados em inglês como definidos pelo grupo de trabalho:

1. "Aplication Software";
2. "Application Program Interface - API";
3. "Application Platform Service Areas";
4. "External Environment Interface";
5. "External Environment".

O terceiro componente, que trata dos serviços necessários nas plataformas nas quais os aplicativos devem estar disponíveis, está dividido em 4 componentes que são críticos para a obtenção de software portátil:

1. - "Serviços de Sistemas Operacionais e Linguagens";
2. - "Serviços de Interface com o Usuário";
3. - "Serviços de Informação";
4. - "Serviços de Comunicações"".

Espera-se que a utilização dessa arquitetura sirva como base para a implementação de software portátil, satisfazendo as expectativas da comunidade de processamento de dados, entre as quais podem ser citadas:

. aumento da produtividade dos programadores e usuários;

. facilidade no uso e suporte dos produtos desenvolvidos pela consistência obtida entre os mesmos; e

. aumento no retorno dos investimentos em sistemas de informação através da melhor utilização dos recursos de computação e da experiência de seus usuários.

A adoção dessa arquitetura pelo PQPS apresenta como vantagens relevantes, além de suas qualidades técnicas, o fato de ser independente de fornecedores e de estar sendo formalizada e padronizada por uma instituição de relevância mundial, a IEEE.

No PQPS decidiu-se inicialmente atacar os componentes de interface com o usuário e gerenciamento de serviços de informações.

Essa escolha foi motivada pela experiência da equipe, pela relevância que esses componentes têm na produção de software de qualidade e no fato de todas as arquiteturas propostas na literatura contemplarem e realçarem a importância dos mesmos.

Consciente da relevância de se obter uma solução satisfatória para o problema de portabilidade de interfaces gráficas, a IEEE constituiu um grupo de trabalho com o objetivo de definir uma interface padrão que permitirá aos desenvolvedores de software implementarem programas portáteis entre múltiplos sistemas de janelas e toolkits, tais como OSF/Motif, OPEN LOOK, Microsoft Windows, OS/2 Presentation Manager e Apple Macintosh.

Esta interface, que pode ser implementada através de uma camada de software sobre os toolkits nativos de diversas plataformas, é uma interface de programação para o desenvolvimento de interfaces gráficas e está sendo chamada de UAPI - **Uniform Application Program Interface**.

O grupo de trabalho da IEEE, P1201.1, já iniciou suas atividades, e os resultados obtidos [IEE93b] estão sendo incorporados em algumas ferramentas existentes no mercado. Uma destas ferramentas, XVT, adquirida pelo CTI, é o objeto deste trabalho.[1 2]

3 - ARQUITETURA E FLUXO DE DESENVOLVIMENTO

O produto XVT é composto por 2 módulos principais: um editor de interfaces gráficas e uma biblioteca multi-plataforma.

O primeiro módulo, XVT-DESIGN, é um gerador de interface e de código C que auxilia ao projetista construir e testar sua interface gráfica. Este módulo também gera um arquivo que contém a definição dos elementos da interface, codificado em uma linguagem denominada de **Universal Resource Language** - URL. **Resources, recursos**, no âmbito de interfaces gráficas, representam informações específicas da interface, tais como rótulos de menus, posições e tamanhos de elementos, etc. A

(1) Parte da funcionalidade que está sendo definida para a UAPI está baseada na funcionalidade da biblioteca XVT, como por exemplo a semântica das operações de criação e destruição de janelas (P1201.1 Draft, sc. 6.5.1).
(2) Os testes foram realizados com a versão 3.01 do sistema. No final da fase de elaboração deste trabalho foi lançada a nova versão do sistema, 4.0. Embora significativas, as mudanças realizadas nessa versão não alteram as conclusões obtidas à respeito do desenvolvimento de interfaces gráficas. Todas as referências relacionadas com a funcionalidade do sistema estão relacionadas com a versão anterior, a menos quando a nova versão é explicitamente citada.

linguagem URL só é utilizada por produtos XVT e é traduzida por um compilador também fornecido pela XVT para a linguagem de definição de recursos do ambiente alvo no qual o aplicativo irá ser executado.

O segundo módulo, a biblioteca multi-plataforma, fornece uma API com funcionalidade básica para a implementação de interfaces gráficas e está disponível entre as principais plataformas existentes: MS-Windows, Windows NT, OSF/Motif, Open Look, OS/2 Presentation Manager, Macintosh e para modo caracter para os ambientes UNIX e DOS.

A portabilidade fornecida pela biblioteca foi obtida através da implementação de uma camada de software sobre os toolkits das várias plataformas. A norma IEEE P1201.1 [IEE93b] utiliza o nome **Uniform Application Program Interface - UAPI** para se referenciar a essa interface. Essa solução é conhecida na bibliografia pelo nome de mínimo denominador comum, uma vez que fornece suporte somente para os elementos encontrados em todas as plataformas [LEE92]. Desta forma, objetos e funções específicos de uma plataforma (como por exemplo os controles do tipo "gauge" encontrados no MOTIF) não podem ser diretamente suportados pela UAPI. O sistema XVT soluciona esse problema ao permitir o desenvolvimento de controles customizados (**custom controls**), que serão apresentados na 5ª seção. O sistema permite também que o programador acesse diretamente os elementos de um toolkit específico, mas, neste caso, a portabilidade de seu código fica comprometida.

O fluxo simplificado de desenvolvimento de um aplicativo para o ambiente MS-Windows no ambiente XVT é apresentado na Figura 1.

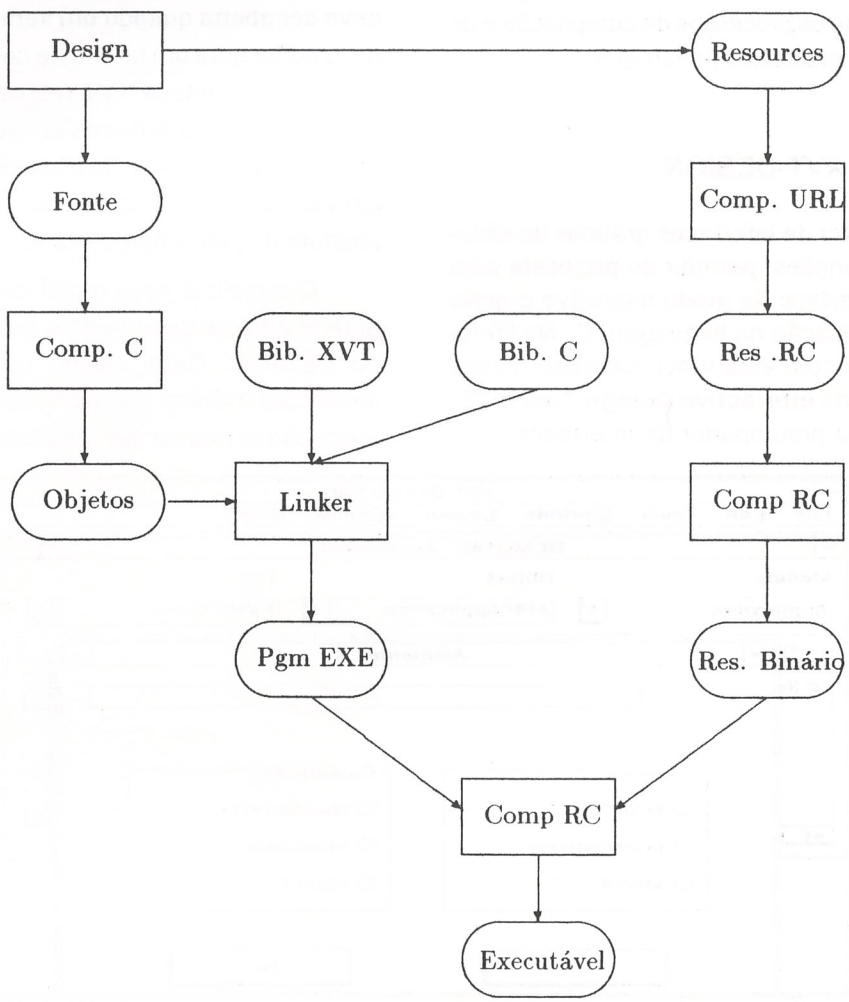

Figura 1 - Fluxo de Desenvolvimento

O aplicativo gerado pode ser distribuído sem a necessidade do pagamento de royalties.

Os testes realizados até o momento utilizaram a versão do editor gráfico que gera código C; existe outra versão que gera código C++.

Um arquivo do tipo "makefile" com os comandos para a compilação, link-edição e demais passos para a geração do aplicativo também é gerado automaticamente. O DESIGN fornece um template deste arquivo, possibilitando que o projetista da interface possa adaptá-lo, se necessário, para as características de seu projeto. Note que, com uma cópia do DESIGN disponível em um único ambiente de desenvolvimento, é possível gerar os arquivos necessários para a produção do aplicativo em todas as plataformas suportadas. Para tanto, os arquivos gerados devem ser transportados para o ambiente alvo, no qual deve haver uma cópia da biblioteca XVT, e onde os processos de compilação e de link-edição devem ser realizados novamente.

4 - XVT-DESIGN

DESIGN é o editor de interfaces gráficas do sistema XVT e possui 2 funções: permitir ao projetista criar os objetos da sua interface de modo interativo e gerar um arcabouço da aplicação na linguagem C. Na literatura, uma ferramenta com esta funcionalidade recebe normalmente o nome de **Interactive Design Tool** (IDT), editor de resources ou prototipador de interfaces.

Usando operações básicas de um editor gráfico do tipo WYSIWYG, o projetista da interface pode criar os objetos de sua interface: janelas, menus, push-buttons, grupos, campos de edição, etc, e posicioná-los convenientemente. Cada objeto é criado com atributos padrão que podem ser alterados pelo projetista através de janelas de diálogo próprias.

Uma série de facilidades para o lay-out de interfaces é oferecida pelo DESIGN; por exemplo, o projetista pode selecionar um conjunto de controles e alinhá-los uniformemente. O projetista também pode utilizar um grid para auxiliá-lo a posicionar os controles nas janelas sendo definidas. A Figura 2 apresenta a tela do DESIGN durante a definição de uma janela de diálogo.

Enquanto o projetista vai definindo os objetos de sua interface, trechos de código C que realizam as operações padrão de acordo com as ações do projetista vão sendo gerados automaticamente pelo DESIGN. Por exemplo, se o projetista define que uma janela de diálogo deve ser aberta quando um item do menu for selecionado, o editor gera um trecho de código com a chamada da função da biblioteca XVT responsável pela abertura da janela com os parâmetros convenientemente preenchidos. A Figura 3 apresenta a tela do DESIGN com o código gerado automaticamente por esta ferramenta para a abertura da janela apresentada na Figura 2.

O projetista pode modificar ou completar o código gerado através de um editor de programas incorporado ao DESIGN. Cada objeto da interface recebe um identificador único que pode ser utilizado no código da aplicação se houver necessidade da aplicação manipular

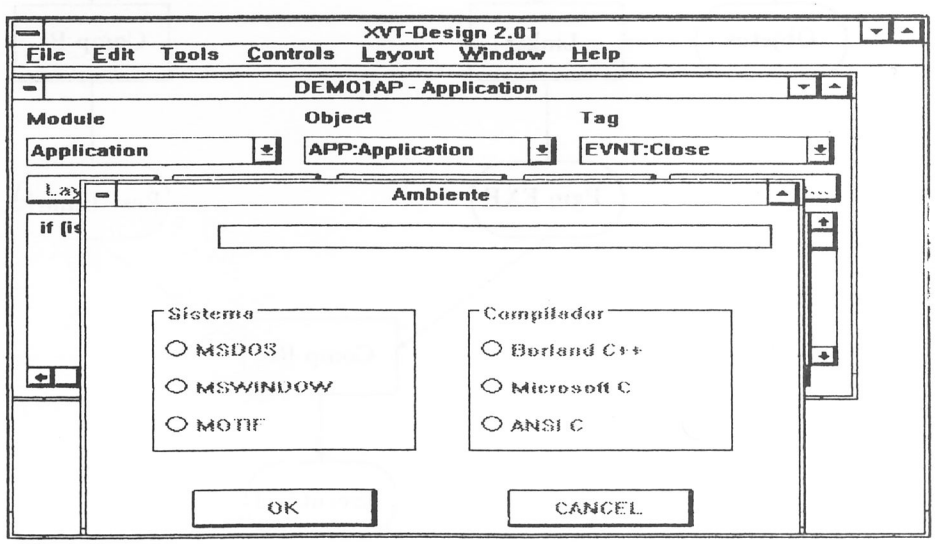

Figura 2 - Definição de Janelas Através do DESIGN

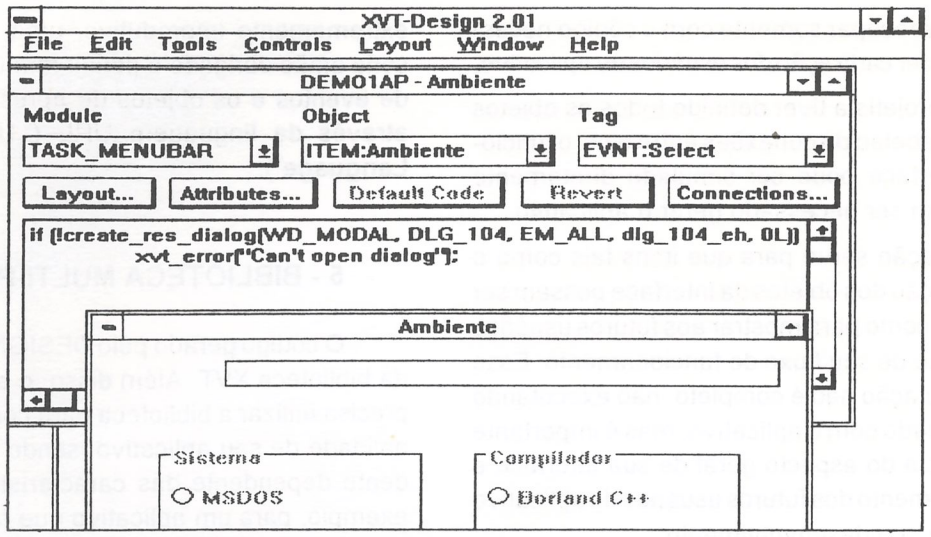

Figura 3 - Código de Abertura de Janela

os objetos da interface. Um exemplo típico deste caso ocorre quando a aplicação encontra uma situação de erro e informa essa condição ao seu usuário através de uma janela de alerta ou de erro, necessitando, para tanto, do identificador dessa janela.

O código gerado pelo DESIGN é suficiente para o controle de boa parte das operações da interface gráfica, tais como a abertura e o fechamento de janelas. Porém, alguns detalhes precisam ser acrescentados pelo projetista da interface, como por exemplo a manipulação da barra de rolagem ("scroll bar") e o sincronismo com o conteúdo da janela associada. O código acrescentado fica automaticamente incorporado ao código gerado pelo DESIGN.

Neste ponto o projetista deve ser bastante criterioso para não misturar o código relativo ao funcionamento da aplicação com o código responsável pelo controle da interface gráfica, de modo que decisões de projeto que afetem a interação com os usuários do aplicativo sejam isoladas das decisões que afetem somente a estrutura do aplicativo ou seu código correspondente. Essa independência entre o código da interface e do aplicativo, conhecida na literatura como **independência de diálogo** [HH89], é de extrema importância para o processo de desenvolvimento interativo e a futura manutenção do aplicativo. O projetista também deve usar o DESIGN para inserir chamadas para as rotinas da aplicação; no âmbito de interfaces gráficas essas chamadas são denominadas como **call-back's**.

Como um programa de interface gráfica típico, um programa XVT deve ser orientado por eventos, e deve ser estruturado de tal forma que exista um trecho de código associado a cada ação do usuário do aplicativo.

Ao passo que programas que não possuem interfaces gráficas recebem dados diretamente de seus usuários, arquivos, etc, programas com interface gráfica recebem dados na forma de eventos, tais como "mouse down", "character typed" ou "update window". Normalmente, cada evento é gerado a partir de ações realizadas pelo usuário (movimentação do mouse, manipulação de seus botões, digitação de caracteres, fechamento de janelas, ...).

Um programa orientado por eventos deve possuir uma rotina responsável por captar os eventos gerados por usuários do programa em tempo de execução, bem como as informações associadas aos eventos (posição do mouse na tela, botão que foi manipulado, tecla que foi pressionada, ...) e enviar essas informações para a rotina responsável pelo tratamento dos eventos do objeto que sofreu a ação do usuário; essa rotina é denominada na literatura como **gerente de eventos**.

O DESIGN gera um gerente de eventos para cada janela do aplicativo, e o mapeamento dos eventos para o gerente de eventos correspondente é realizado automaticamente por uma rotina de controle da biblioteca XVT.

Embora aparentemente possa parecer trabalhoso desenvolver um programa orientado por eventos, o fato do DESIGN gerar automaticamente os gerentes de eventos necessários simplifica bastante o trabalho do projetis-

ta, que deve se preocupar somente com o código necessário para responder cada ação dos usuários do aplicativo.

Quando o projetista tiver definido todos os objetos da interface e estabelecido conexões entre eles, o funcionamento da interface pode ser simulado diretamente pelo DESIGN sem ser necessário gerar o aplicativo.

Essa simulação serve para que itens tais como o tamanho e a posição dos objetos da interface possam ser verificados, bem como para mostrar aos futuros usuários do aplicativo parte de seu fluxo de funcionamento. Esse processo de simulação não é completo, não executando o código C associado com o aplicativo, mas é importante para dar uma idéia do aspecto geral de sua interface e permitir o envolvimento dos futuros usuários do aplicativo desde o início de seu desenvolvimento.

Um ponto positivo apresentado pela versão 4.0 do DESIGN é a possibilidade de trabalho em equipe, com a integração de módulos desenvolvidos separadamente. Dessa forma, se um projeto for dividido em sub-sistemas, a interface gráfica de cada sub-sistema pode ser implementada paralela e independentemente da implementação das interfaces gráficas dos outros sub-sistemas, da mesma forma que possivelmente ocorrerá com a implementação do restante dos sub-sistemas. O DESIGN oferece facilidades para a integração final das interfaces gráficas desenvolvidas separadamente.

De maneira geral, a utilização desta ferramenta é razoavelmente simples, exigindo somente um pequeno período de treinamento da parte de seus usuários. O desenvolvimento de interfaces gráficas com o XVT sem o emprego do DESIGN não é aconselhável por ser extremamente improdutivo, uma vez que o projetista seria então obrigado a definir manualmente os gerentes de eventos e os objetos de apresentação da interface através da linguagem URL (“**Universal Resource Language**”).

5 - BIBLIOTECA MULTI-PLATAFORMA

O código gerado pelo DESIGN utiliza várias rotinas da biblioteca XVT. Além disso, o programador também precisa utilizar a biblioteca para complementar a funcionalidade de seu aplicativo, sendo o esforço correspondente dependente das características do mesmo. Por exemplo, para um aplicativo que faça entrada e validação de dados o programador precisará somente desenvolver o código que apanha os valores digitados e passá-los para as rotinas de validação do aplicativo. Por outro lado, para um editor gráfico, todo o trabalho de manipulação de figuras deverá ser codificado pelo programador com a utilização das rotinas da biblioteca.

Como foi visto, a API XVT é mapeada sobre os toolkits das plataformas suportadas, de modo que as aplicações possuem o look-and-feel das plataformas nas quais são executadas.

Por exemplo, a seguinte chamada:

open-window ();

gera uma janela com o look-and-feel WINDOWS (Figura 4) ou com o look-and-feel MOTIF (Figura 5), dependendo somente do ambiente para o qual o aplicativo for gerado.

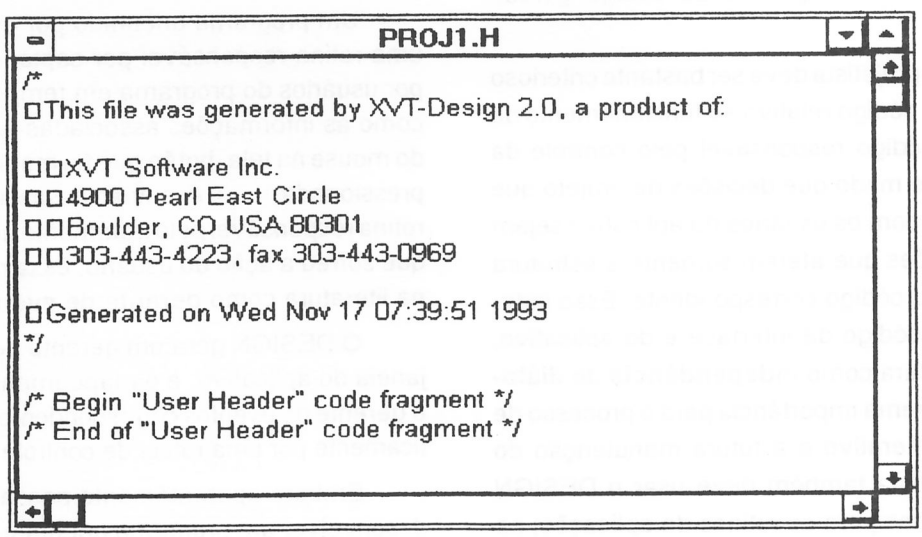

Figura 4 - Janela com look-and-feel Windows

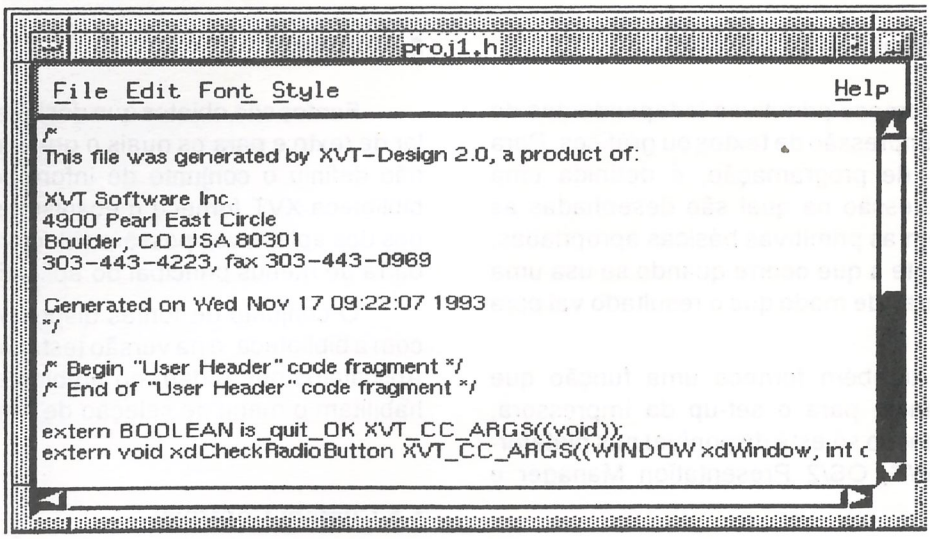

Figura 5 - Janela com look-and-feel MOTIF

Embora as barras de rolagem (scroll bar) das janelas tenham comportamentos distintos,[3] a rotina que trata os eventos desses objetos não precisa saber se estão recebendo eventos de uma janela WINDOWS ou MOTIF, pois em termos de programação é desenvolvido um código único, independente de plataforma.

A biblioteca dispõe de primitivas para a manipulação dos objetos básicos de interfaces gráficas:

. Janelas . Controles
. Eventos . Operações Gráficas
. Menus . Manipulação do clip board

A biblioteca fornece também outras funções que serão sucintamente apresentadas a seguir:

5.1 Diálogos pré-definidos:

O XVT fornece diálogos pré-definidos para as seguintes ações:

. solicitar uma resposta ao usuário do tipo sim/não;
. ativar uma janela de erro ou de alerta;
. solicitar uma cadeia de caracteres digitada pelo usuário;
. ativar uma janela para escolha de nome de arquivo;
. iniciar a operação de impressão

Esses diálogos são ativados a partir de rotinas da biblioteca; por exemplo, a janela mostrada na Figura 6 foi exposta simplesmente através da chamada da seguinte rotina:

open-file-dlg ();

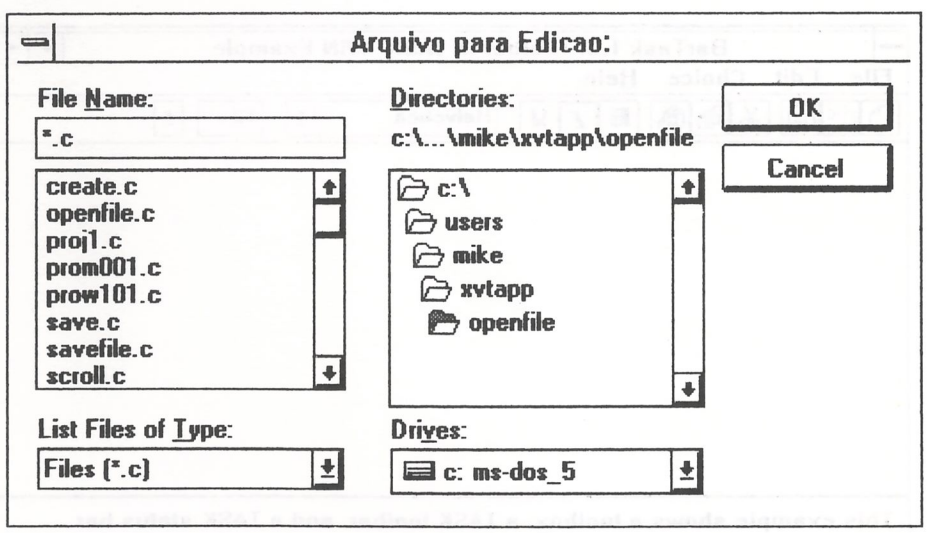

Figura 6 - Janela de Abertura de Arquivos em Windows

(3) O tamanho do controle da barra de rolagem de uma janela MOTIF varia de acordo com a proporção entre o tamanho do arquivo apresentado na janela e o tamanho da janela, enquanto que no ambiente Windows o tamanho do mesmo elemento é fixo.

5.2 Suporte às operações de impressão:

A biblioteca fornece primitivas independentes de plataforma para a impressão de textos ou gráficos. Para tanto, em termos de programação, é definida uma janela do tipo impressão na qual são desenhadas as figuras e textos com as primitivas básicas apropriadas, de forma semelhante a que ocorre quando se usa uma janela gráfica normal, de modo que o resultado vai para a impressora.

A biblioteca também fornece uma função que apresenta uma janela para o set-up da impressora; atualmente esta função só está disponível nas plataformas MS-WINDOWS, OS/2 Presentation Manager e Macintosh.

5.3 Compilador HELP:

O sistema fornece facilidades para a produção de janelas de auxílio com explicações para os usuários finais dos aplicativos.

Para tanto, é necessário que antes da geração do aplicativo seja definida uma lista de tópicos com uma explicação associada a cada tópico em um arquivo do tipo texto. Este arquivo é transformado pelo utilitário CCHELP, distribuído junto com o sistema XVT, de forma que o resultado obtido possa ser utilizado em tempo de execução.

Embora portátil, o formato do arquivo de mensagens de auxílio obtido é bastante rudimentar, fornecendo um único nível de auxilio, não permitindo o emprego de gráficos nem de hipertexto.[4]

5.4 Fontes:

Fontes são objetos que designam um estilo particular de texto e para os quais o grupo de trabalho da IEEE não definiu o conjunto de informações associadas. A biblioteca XVT fornece mecanismos para que os usuários dos aplicativos possam selecionar fontes a partir da barra de menus principal do aplicativo.

O conjunto de fontes disponíveis varia de acordo com a biblioteca, e na versão testada é bastante limitado. Aplicativos que rodam no ambiente DOS-caracter não habilitam o menu de seleção de fontes.

5.5 Custom Controls:

Além de fornecer os elementos gráficos básicos de interfaces gráficas, o sistema permite que se desenvolvam novos controles customizados para atender necessidades específicas. Isto é obtido através de objetos do tipo "custom controls". Um objeto desse tipo, obtido a partir de primitivas existentes, também é portátil entre plataformas. Atualmente algumas firmas já comercializam alguns objetos deste tipo, e a própria XVT desenvolveu e comercializa à parte objetos custom control do tipo tabela ou planilha.

A existência de objetos desse tipo confere ao sistema XVT a característica de **extensibilidade**, definida em [HH89] como a possibilidade de incluir novos elementos gráficos aos elementos suportados diretamente pela ferramenta empregada para produção da interface gráfica.

A Figura 7 apresenta uma janela cuja barra de ícones foi obtida através de um objeto customizado.

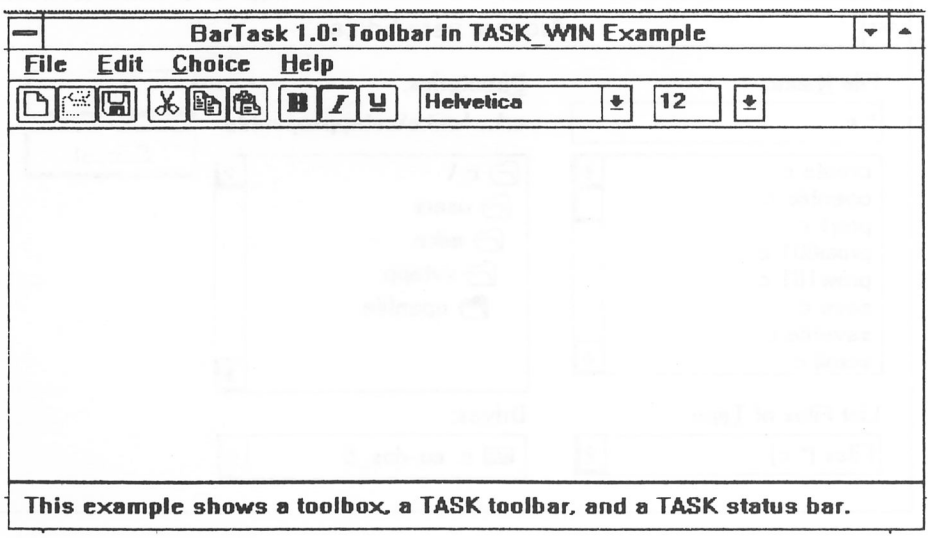

Figura 7 - Exemplo de Uso de Objeto Customizado

(4) A versão mais recente do sistema, 4.0, passou a gerar janelas de auxílio do tipo hipertexto.

5.6 Ícones:

Embora as operações de manipulação de ícones estejam disponíveis nas bibliotecas de todas plataformas, esses objetos são tratados de maneira não portátil pela biblioteca XVT. O fato do formato destes objetos ser dependente das plataformas justifica, em parte, esta restrição, que deverá ser eliminada das futuras versões da biblioteca.

5.7 Suporte a operações específicas de plataformas:

A biblioteca possui uma rotina que fornece uma interface consistente mas não portátil para utilizar a funcionalidade específica das plataformas. Cada plataforma oferece códigos de escape que podem ser utilizados pelos aplicativos para realização de ações específicas da plataforma. Esses códigos são passados como parâmetros para a rotina **xvt-escape**, e para cada código estão definidos argumentos que também devem ser passados para essa rotina.

Por exemplo, para os ambientes MS-WINDOWS e Macintosh existem o código de escape **xvt-esc-get-printer-info** que devolve informações à respeito da resolução da impressora.

Uma segunda maneira de se manipular aspectos específicos das plataformas é através da rotina **xvt-set-value** com atributos não portáteis. Por exemplo, para o ambiente MS-WINDOWS, um aplicativo pode definir e usar fontes específicos deste ambiente com o uso dessa rotina.

Essas possibilidades, embora impactem a portabilidade dos aplicativos, são importantes por encapsularem os toolkits originais das plataformas e permitem que o sistema XVT possua a característica de **escapabilidade**, definida em [HH89] como a possibilidade de acessar o toolkit da plataforma alvo para implementar elementos não suportados diretamente pela ferramenta empregada para produção da interface gráfica.

6 - TESTES REALIZADOS:

Com a finalidade de verificar a funcionalidade oferecida pelo sistema bem como a portabilidade do código gerado pelo editor de interfaces DESIGN foram definidos e implementados alguns protótipos.

A escolha dos mesmos foi realizada de modo a se cobrir os principais elementos de uma interface gráfica. Todos os testes foram desenvolvidos com o editor de interfaces do sistema.

Interfaces simples, que utilizem somente janelas de diálogos, radio-buttons, check-boxes e campos de edição, são rapidamente implementadas e podem ser obtidas sem um grande trabalho adicional de programação ao código gerado pelo DESIGN.

As operações gráficas, desenhos de figuras, por exemplo, também podem ser facilmente programadas e não exigem um profundo conhecimento da biblioteca.

As operações de rolagem de tela (scroll) não são implementadas automaticamente, de modo que devem ser programadas manualmente. Embora a quantidade de código que precisa ser desenvolvido não seja grande, a documentação relacionada com este assunto não é muito clara, de modo que foi necessário um tempo razoável para que esta operação pudesse funcionar corretamente em nosso protótipo.

A manipulação de fontes de caracteres de uma maneira portátil e independente de plataforma é um problema complexo; a funcionalidade fornecida pela biblioteca não permite se obter, por exemplo, todos os recursos fornecidos pelo ambiente MS-WINDOWS. Este é um ponto no qual a possibilidade de se utilizar diretamente primitivas específicas das plataformas pode justificar a perda de parte da portabilidade do aplicativo gerado.

A biblioteca não fornece rotinas para validação de campos de entrada de dados nem facilidades do tipo "picture". Embora a implementação dessas funções pelo projetista da interface não seja uma tarefa complexa, não deixa de ser um fator limitante em termos de produtividade.

Os manuais são simples, mas muitas vezes é necessário se recorrer aos exemplos distribuídos com o sistema para se obter uma visão completa de determinadas operações, como por exemplo a operação de rolagem de tela.

O primeiro protótipo desenvolvido permite que seus usuários, utilizando uma opção da barra de menus, abram 2 janelas de diálogo. As Figuras 8 e 9 apresentam uma das janelas do protótipo abertas respectivamente nos ambientes MS-WINDOWS e OSF/Motif.

Não foi programada nenhuma rotina de consistência para os dados digitados, e quando o botão "OK" é selecionado os mesmos são apresentados em uma janela.

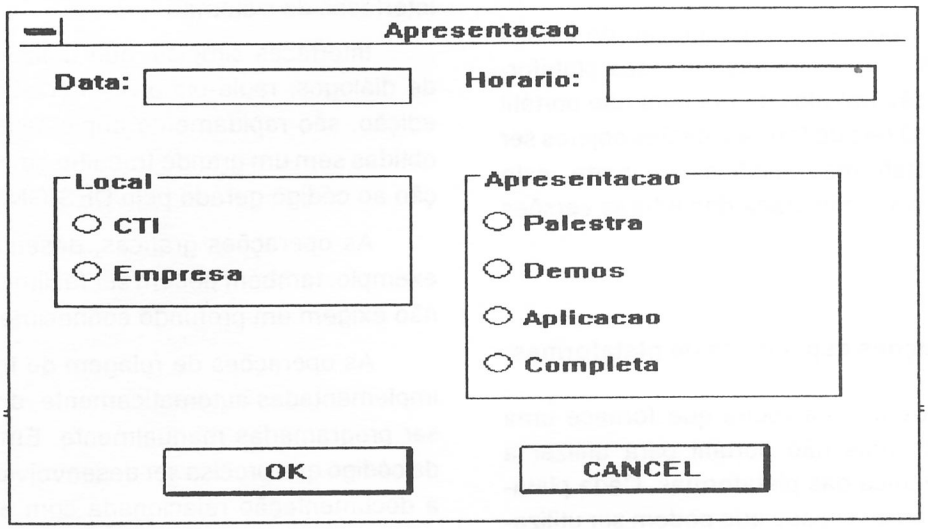

Figura 8 - Janela do Protótipo em Ambiente Windows

O aplicativo foi desenvolvido inicialmente no ambiente MS-WINDOWS com o compilador Borland C/C++ versão 3.1 e foi transportado para os ambientes OSF/MOTIF e DOS-Caracter sem alterações em seu código fonte.

A eficiência dos programas obtidos é satisfatória não tendo sido percebido nenhum impacto no desempenho dos mesmos; o tamanho dos executáveis é mostrado a seguir:

- Ambiente -	- Tamanho -
MS-WINDOWS	31.170K
DOS	419.004K
Motif	1.146.800K

No ambiente MS-WINDOWS, os executáveis gerados utilizam bibliotecas DLL fornecidas junto com o sistema, o que explica o tamanho obtido.

O sistema tem problemas para apresentar os objetos da interface no ambiente DOS-Caracter quando os mesmos não têm sua origem coincidente com a origem de um caracter na tela. Como a interface deste protótipo foi desenvolvida com o DESIGN no ambiente MS-WINDOWS foi necessário que a opção de grid estivesse ligada com o tamanho das células igual a 1 caracter.

7 - CONCLUSÕES E PERSPECTIVAS - PRÓXIMOS PASSOS

Os testes realizados até o momento foram suficientes para a avaliação da ferramenta; os principais resultados serão apresentados a seguir.

A portabilidade da biblioteca foi bastante testada e nunca foi encontrado nenhum problema de transporte de suas rotinas, exceto pelo fato dela não suportar a manipulação de ícones de uma maneira independente de plataforma. Isso significa que grande parte do problema de transporte do código do aplicativo fica resolvida, restando ao programador tomar os cuidados necessários para garantir que seu código também pode ser transportado para outros ambientes sem problemas. Nesse campo, a utilização do padrão POSIX é fundamental, e técnicas como as apresentadas em [LEV91] e [ADR90] são de extrema importância.

Embora o código da aplicação possa ser transportado sem problemas, nem sempre o mesmo ocorre com alguns elementos da interface, pois em determinadas situações o tamanho ou a posição desses elementos precisam ser ajustados quando a

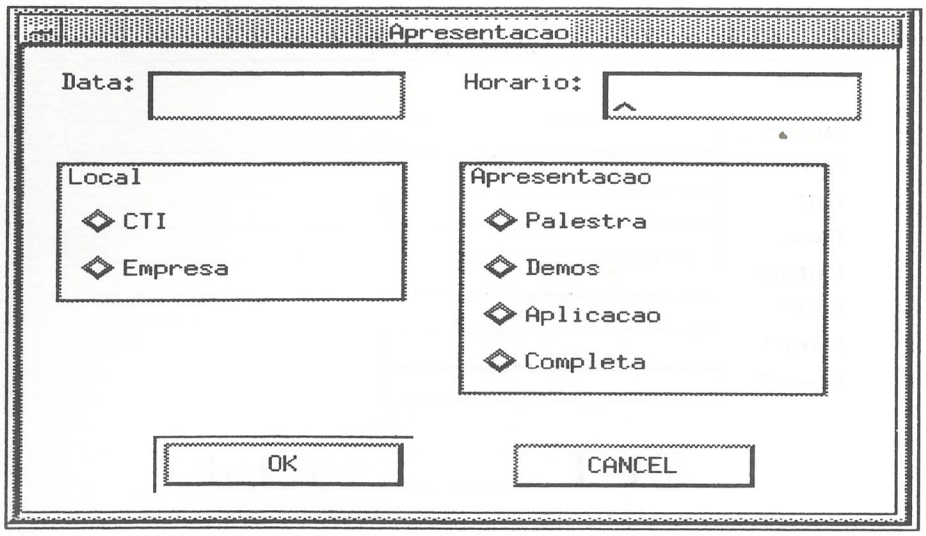

Figura 9 - Janela do Protótipo em Ambiente Motif

interface muda de plataforma. A Figura 10 apresenta uma janela de diálogo originalmente definida pelo Design no ambiente MS-WINDOWS. Note que quando o aplicativo foi transportado para o ambiente OSF/MOTIF os campos de entrada de dados ficaram sobrepostos, conforme pode ser observado na Figura 11.

Essa situação ocorre porque o DESIGN gera os elementos da interface com o look-and-feel da plataforma na qual ele está sendo executado; no exemplo apresentado, a relação entre a distância e a altura dos campos de edição apresentava um aspecto agradável no ambiente MS-WINDOWS, o que deixou de ocorrer quando o aplicativo foi transportado para o ambiente MOTIF. Existem soluções simples e parciais para esse problema, como por exemplo a manutenção de um arquivo com a definição dos elementos da interface em cada plataforma para qual o aplicativo é transportado. De qualquer forma, o fato do sistema não possuir um gerente de geometria, que permitiria que a distância entre elementos da interface pudesse ser definida, não deixa de ser uma limitação.

Nenhum dos testes realizados apresentou problemas em termos de desempenho, não tendo sido identificada nenhuma situação na qual a camada da biblioteca XVT tenha provocado degradação no tempo de resposta do aplicativo desenvolvido, de modo que o desempenho da interface gráfica é dependente do desempenho do toolkit no qual o aplicativo será executado. Os testes realizados também constataram que a biblioteca é suficientemente eficiente para a implementação de editores gráficos em máquinas PC 386/486.

A qualidade das interfaces geradas foi bastante satisfatória, e em nenhum dos testes realizados foi encontrado algum elemento que pudesse identificar a utilização ou não da biblioteca XVT. De forma análoga do que ocorre com e eficiência dos aplicativos, a qualidade da interface também é dependente dos toolkits nativos.

Um ponto que, de certa forma, compromete a qualidade dos aplicativos gerados é o gerenciamento de fontes. A biblioteca XVT fornece mecanismos para o usuário dos aplicativos selecionar fontes a partir da barra de menus principal, mas o conjunto de fontes disponíveis varia de acordo com a biblioteca. Por exemplo, a biblioteca para o ambiente WINDOWS fornece somente os fontes Modern, Roman, Script, Courier, Helvética e Times Roman, nos estilos Bold, Italic e Underline, de modo que usuários de aplicativos desenvolvidos pelo XVT para a plataforma MS-WINDOWS não poderão contar com os outros fontes que por ventura estejam instalados no ambiente.

Embora o sistema XVT seja suficientemente genérico para poder ser utilizado em qualquer área de aplicação, algumas ferramentas existentes são mais voltadas para áreas específicas, permitindo uma maior produtividade para o desenvolvimento de sistemas nas mesmas.

Por exemplo, para o desenvolvimento de aplicações do tipo cliente-servidor para as plataformas MS-WINDOWS e Motif que manipulem bases de dados através da linguagem SQL existem no mercado ferramentas que fornecem uma produtividade bastante superior à oferecida pelo sistema XVT, uma vez que tratam essas áreas de aplicação de forma mais completa. Aplicações genéricas, entretanto, continuam sendo melhor tratadas como um produto mais abrangente como o XVT.

O tempo necessário para que um programador utilize o editor de interfaces DESIGN de uma maneira produtiva é razoavelmente curto (aproximadamente uma ou duas semanas). O domínio total da biblioteca que possui cerca de 300 rotinas, é mais demorado, levando entre 2 e 3 meses, dependendo principalmente da expe-

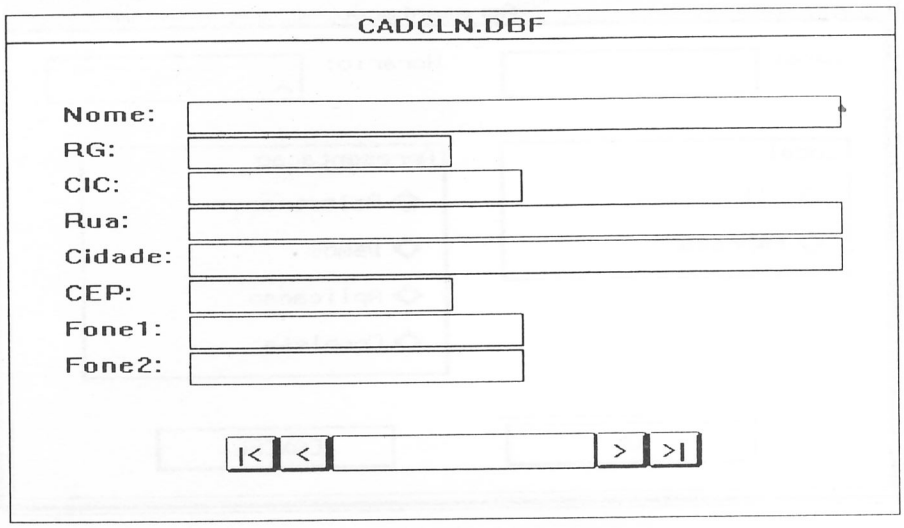

Figura 10 - Janela com look-and-feel Windows

riência que o programador possua no desenvolvimento de aplicativos com interfaces gráficas.

A experiência do CTI permite afirmar que, embora exista um custo associado, é técnica e economicamente viável o desenvolvimento de interfaces gráficas multi-plataformas.

O Programa de Qualidade e Produtividade em Software do CTI continuará a absorver tecnologia para a produção de interfaces gráficas portáteis e a transferir esta tecnologia para a indústria nacional de software. Entre as metas propostas nesta área, destacam-se as seguintes:

. Tecnologia de Objetos

Além de ser considerada uma tecnologia avançada para a produção de software, o desenvolvimento de interfaces gráficas foi um dos primeiros campos nos quais a tecnologia de objetos foi empregada com sucesso. A decisão de se iniciar o estudo do sistema XVT com sua versão não orientada a objetos foi motivada pelo fato de se procurar inicialmente atuar em um patamar tecnológico mais próximo ao praticado em nível nacional. Procurando avançar neste patamar está prevista a compra da versão do sistema XVT orientada a objetos.

. Teste de Interfaces Gráficas

Testar aplicativos com interface gráfica é uma atividade mais trabalhosa do que testar aplicativos com interface baseada em caracteres [THE92], [MYE93] e [HS93]. Existem no mercado algumas ferramentas desenvolvidas para automatizar esta atividade. O PQPS já adquiriu uma ferramenta deste tipo e está iniciando sua avaliação.

. Métodos

Devido ao forte conteúdo semântico associado a uma interface gráfica, seu projeto é mais complexo do que o projeto de uma interface tradicional [LR93], [MAY92]. Espera-se iniciar o desenvolvimento de uma metodologia que busque indicar os principais passos que devem ser seguidos para a obtenção de uma interface gráfica de qualidade.

Um projeto deste tipo desenvolvido em uma instituição de pesquisa com as características do CTI, que tem como um dos seus objetivos induzir e apoiar a introdução de tecnologia nas empresas nacionais, só pode obter sucesso se conseguir repassar efetivamente seus resultados para o setor produtivo do país. O PQPS escolheu como método preferencial para repassar sua experiência o desenvolvimento de projetos conjuntos com empresas nacionais.

A atividade de divulgação do ambiente para a produção de interfaces gráficas bem como a busca de empresas interessadas em desenvolvimentos conjuntos deverá continuar dentro das atividades do PQPS. Os contatos já realizados com associações como a Assespro e o Programa Softex 2000 apresentaram resultados iniciais animadores e deverão ser aprofundados. O CTI procurará facilitar o acesso dos pequenos produtores de software aos resultados obtidos através de, por exemplo, convênios com o SEBRAE e com o Programa SOFTEX 2000.

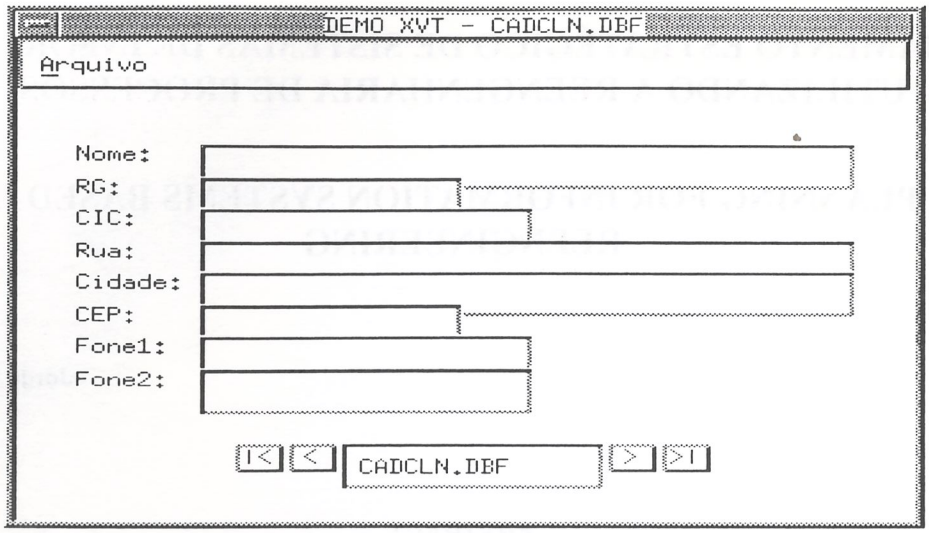

Figura 11 - Janela com look-and-feel Motif

O CTI espera que o caminho de integração com a indústria nacional de software seja percorrido com sucesso e se coloca à disposição dos interessados para os contatos que forem necessários.

Agradecimentos:

A José Carlos Maldonado, pelas sugestões e incentivos dados durante o desenvolvimento deste trabalho.

REFERÊNCIAS BIBLIOGRÁFICAS

[ADR90] D. Keppel A. Dolenc, A Lemmke and G. V. Reilly. **Notes on writing portable programs in C.**, Technical report, CSE, University of Washington, Novembro 1990.

[API94] Steve Apiki. **Paths to platform independence.** Byte, 19(01): 172 - 178, 1994.

[BA] L. Bass and G. Abowd. **Issues in the evaluation of user interfaces tools**. Technical report, Software Engineering Institute.

[HH89] H. Rex Hartson and Debora Hix. **Human-computer interface development: Concepts and systems for its management**. ACM Computer Surveys, 21(1) 5 - 92, 1989.

[HS91] Debora Hix and Robert S. Schulman. **Human-computer interface development tools: a methodology for their evaluation**. Communications of the ACM, 34(3) 75 - 87, 1991.

[HS93] Ellis Horowitz and Zafar Singhera. **Graphical user interface testing**. Technical report, Department of Computer Science - University of Southern California, 1993.

[IEE93a] IEEE. **P1003.0 Draft 16.1 - Draft Guide to the POSIX Open System Environment,** October 1993.

[IEE93b] IEEE. **P1202.1/Draft 7 - Draft Stantard for Information Technology - Uniform Application Program Interface - Graphical Users Interfaces**, March 1993.

[LEE92] Geoff Lee. **Object-Oriented GUI Application Development.** PTR Prentice-Hall, 1992.

[LEV91] Donald Levine. **POSIX Programmer's Guide - Writing Portable UNIX Programs.** O Reilly and Associates, 1991.

[LR93] Clayton Lewis and John Rieman. **Task-Centered User Interface Design.** Livro em Domínio Público disponível via ftp anônimo em ftp.cs.colorado.edu, 1993.

[MAY92] Deborah J. Mayhew. **Principles and Guidelines in Software User Interface Design.** PTR Prentice-Hall, 1992.

[MR92] Brad A. Myers and Mary B. Rosson. **Survey on user interface programming**. Technical report, Computer Science Department - Carnegie Mellon University, Fevereiro 1992.

[MYE93] Brad A. Myers. **Why are human-computer interfaces difficult to design and implement**. Technical report, Computer Science Department - Carnegie Mellon University, Julho 1993.

[THE92] Lee The. **Stress tests for gui programs**. Datamation, pages 5 - 92, 1992.