GRAEDIUS : Uma proposta para Definição de Interfaces Baseada em Gramática de Atributos[1]

Edson Gellert Schubert *
Marcelo Soares Pimenta *, +
Roberto Tom Price *

* CPGCC/UFRGS
+ CEC/UFSC

Endereço: Caixa Postal 1501 - CEP 90001
Porto Alegre - RS
E-mail: tomprice@sbu.ufrgs.anrs.br

Resumo

O uso de gramática de atributos é sugerido para a definição da interação entre o usuário e as aplicações. A proposta GRAEDIUS estende as definições sintáticas e semânticas através da agregação de representações ortogonalmente independentes para sintaxe concreta, técnicas de interação e seleção de regras de apresentação. O artigo descreve em linhas gerais a proposta e usa um exemplo para ilustrar a definição de interfaces.

1. Introdução

A pesquisa em interfaces com o usuário (IU) vem desenvolvendo ferramentas que auxiliam seu projeto e implementação, como SGIUs (Sistemas de Gerência de IUs). Um modelo de representação de interface com o usuário (MRIU), utilizado por ferramentas isoladas ou SGIUs, é um esquema notacional para descrever a interface com o usuário, mais notadamente o componente de diálogo. Este modelo forma a base usada para a descrição de IUs e influencia decisivamente a implementação da ferramenta. O poder descritivo de um modelo é o conjunto de IUs que podem ser descritas por ele, fornecendo uma medida do seu grau de generalidade.

Este trabalho propõe a utilização de gramática de atributos para a descrição de IUs. Este modelo de representação será referido por GRAEDIUS (GRamática de Atributos Estendida para a Descrição de Interfaces com o USuário) no texto.

Na seção 2 deste artigo são apresentadas algumas características de um dos principais MRIUs: Gramáticas Livres do Contexto (GLC). A seção 3 apresenta os conceitos básicos de GRAEDIUS como uma extensão de GLC. Na seção 4 um exemplo é utilizado para demonstrar a viabilidade da proposta. Finalmente, a seção 5 reune algumas observações conclusivas deste trabalho.

---

[1] - Este trabalho foi desenvolvido com o apoio financeiro do CNPq, da FINEP e da SID/Informática (Projeto ESTRA).

2. O Modelo de Gramáticas Livres do Contexto

Três dos principais MRIUs existentes na literatura são: o modelo de redes de transição de estados [SHN 86], [MAR 74], o modelo de eventos [GRE 86] e o modelo de gramáticas livres do contexto (GLC) [MYE 89].

GLC para descrição de IUs são estendidas com a invocação de ações semânticas. Suas características principais são:

- os terminais da gramática são os tokens de entrada que representam as ações do usuário;
- as produções da gramática definem a linguagem empregada pelo usuário em suas interações com a aplicação;
- ações semânticas (da aplicação) são associadas às produções da gramática e são executadas quando a produção é usada para o reconhecimento de uma entrada do usuário.

Se uma gramática G descreve o diálogo de uma IU, então a linguagem L(G) descrita por esta gramática contém todas as seqüências permitidas de ações do usuário. Pode-se então produzir um reconhecedor que reconhece L(G).

Se o reconhecedor é top-down, quando a produção é expandida, a ação associada à produção é executada. Uma produção é selecionada para expansão logo que o primeiro terminal que é capaz de ser gerado pelo corpo da produção (seu lado direito) for igual ao token de entrada corrente. Então a ação associada à produção será executada quando o início do comando é reconhecido.

Um reconhecedor bottom-up, pelo contrário, acumula tokens de entrada até que sejam iguais ao corpo (lado direito) de uma produção. O corpo é então substituído pelo não-terminal do lado esquerdo da produção e o processo continua até o símbolo inicial da gramática ser alcançado. Como a produção não é usada até que todos os símbolos de seu lado direito tenham sido reconhecidos, a ação associada só será executada quando o comando representado pela produção tiver sido completamente reconhecido.

Uma mudança da técnica de reconhecimento (top-down ou bottom-up) causa um dos principais problemas para a utilização de GLC na definição de IUs: como visto, a ordem de execução das ações semânticas depende do método (reconhecedor) adotado. Além disto, usando-se apenas GLC torna-se difícil realizar trocas de parâmetros e fazer verificações semânticas entre as produções.

3. Modelo de GRamática de Atributos Estendida para a Definição de Interfaces com o USuário (GRAEDIUS)

Por definição, uma Gramática de Atributos (GA) é uma tripla (G,A,E) onde:

- G é uma GLC, isto é, G=(N,T,P,S) onde

  N é um conjunto de símbolos não terminais
  T é um conjunto de símbolos terminais
  P é um conjunto de produções
  S é o símbolo inicial da gramática;
- A é um conjunto de atributos onde

  A(X) = atributos para X pertencente a N ∪ T
  e
  A(X) e A(Y) são disjuntos sss X <> Y
- E é um conjunto de equações de atributos onde

  E(p) = equações para a produção p pertencente a P

GA tem sido usada na prática como um poderoso formalismo para descrever semântica de linguagens de programação. Técnicas incrementais de avaliação de atributos têm tornado possível a construção de ambientes de programação baseados em editores de linguagens [ESP 89].

A idéia de usar GA para definição de IUs surgiu a partir da similaridade entre os problemas relacionados ao diálogo interativo e os relacionados ao reconhecimento incremental. Assim como no reconhecimento incremental, uma ação do usuário num diálogo interativo (evento) desencadeia uma série de ações da aplicação (e possíveis modificações da própria IU) que podem por sua vez desencadear outras, num processo de propagações sucessivas. Esta dependência entre ações do usuário e ações da aplicação (e da IU), direta e indiretamente desencadeadas, torna extremamente adequada a utilização de GA.

O mesmo mecanismo de avaliação que permite que a correção semântica de programas possa ser automaticamente verificada sempre que alguma mudança é feita na sua árvore sintática abstrata correspondente (em um editor dirigido pela sintaxe), permite a automatização do controle do diálogo entre o usuário e a aplicação [FAV 87]. Com isto integra-se a definição da IU à definição do resto do sistema, através do uso do mesmo formalismo para ambas as definições.

Scott Hudson mostrou, em seu sistema Planit, que apresentações podem ser computadas através de atributos [HUD 86]. No entanto, apesar de usar GA para apresentações, Hudson fez uso também de diagramas de transição de estados para manipular/descrever diálogos, o que implica na existência de um formalismo híbrido para descrição de IU: um para o componente de apresentação e outro para o componente de diálogo.

A idéia da presente proposta é usar um banco de dados relacional para armazenar as apresentações e expressar o controle de diálogo e da exibição das apresentações em equações semânticas de uma GA. Quando o usuário realiza uma ação, o feedback semântico desta ação é refletido mudando-se o que é exibido. Para esta mudança, utiliza-se as informações de apresentação contidas nas relações. A apresentação é definida, desta forma, independentemente do diálogo, possibilitando que

um modo de diálogo possa ter várias opções de apresentação. Além disso, o controle do diálogo pode ser gerenciado dinamicamente através da avaliação dos atributos, o que implica que uma aplicação pode ter diferentes modos de diálogo. Estas características podem ser facilmente incorporadas devido à flexibilidade do formalismo de GA.

A combinação de relações e GA foi proposta primeiramente para verificação semântica de linguagens de programação [HOR 86], no contexto de um gerador de editores dirigidos pela sintaxe [REP 84]. Esta proposta acrescia à GA básica operadores de conjuntos e da álgebra relacional, criando uma Gramática Relacionalmente Atributada.

Para o efetivo suporte à definição de IU, nossa proposta é criar uma GA estendida da seguinte forma:

(1) Declaração de relações. Uma relação pode ser vista como uma tabela, onde cada linha representa uma tupla e cada coluna contém valores de um domínio definido. As colunas são muitas vezes denominadas "atributos" mas neste artigo usaremos o nome "campo" para evitar confusões. As relações são, como veremos adiante, definidas na seção **Definições de Relações**;
(2) Associação de funções de interação às produções. Uma função de interação é uma função pré-definida que provê suporte às técnicas de interação (menus, form-fill, question/answer, linguagem de comandos, etc.) e ao controle de suas execuções. Seus parâmetros podem ser valores de atributos da gramática ou valores de campos de uma relação declarada.
(3) Equações semânticas incluindo operações sobre as relações como os operadores de conjunto UNIÃO, INTERSECÇÃO, DIFERENÇA e PRODUTO CARTESIANO e os operadores da álgebra relacional SELEÇÃO, PROJEÇÃO E JUNÇÃO.

As funções de interação e os operadores sobre as relações são providos pelo avaliador das equações semânticas da GRAEDIUS. O avaliador do GRAEDIUS pode ser construído como uma "casca" sobre um avaliador de GA "standard", que em geral é uma "casca" sobre o reconhecedor da gramática. Este enfoque permite, além de uma implementação mais rápida e segura, independência do método de avaliação (incremental, não-incremental, dinâmico, estático, por demanda, etc. [HOO 89]) e de reconhecimento (top-down ou bottom-up) adotados. GRAEDIUS configura-se, portanto, como um aprimoramento de GLC para representação de interfaces pela agregação de representações - concreta, abstrata e semântica - ortogonais entre si.

Uma especificação em GRAEDIUS subdivide-se em 9 seções, a saber :

. Definição de Constantes
. Definição de Apresentações
. Definição de Relações
. Sintaxe Abstrata & Semântica
. Sintaxe Concreta de Entrada

. Sintaxe Concreta de Saída
. Formatos de Interação
. Regras de Associação de Entrada
. Regras de Associação de Saída

Cada um destas seções especifica uma parte da aplicação. Esta modularização, apesar de não necessária, é interessante por facilitar a leitura, compreensão e programação da aplicação.

Na seção **Definição de Constantes** são declaradas as constantes usadas nas outras seções.

Na seção **Definição de Apresentações** são descritas as apresentações como um conjunto de parametros fisicos (cores, tamanho de janelas, estilos de texto, etc.).

Na seção **Definição de Relações** são descritas as tabelas de apresentações. Cada tabela é composta por atributos correspondentes às informações relativas aos parâmetros fisicos de apresentação definidos na seção anterior.

Na seção **Sintaxe Abstrata & Semântica** são declarados os componentes sintáticos e semânticos da aplicação em si. As produções do componente sintático, definem a composição sintática da aplicação. O componente semântico, definido pelas equações semânticas associadas às produções, determina o interrelacionamento de valores dos atributos das produções.

Nas seções **Sintaxe Concreta de Entrada** e **Sintaxe Concreta de Saída** são definidas as técnicas de interação que estarão disponiveis para cada uma das necessidades de interação. Esta definição associa, para cada nome de produção da sintaxe abstrata em que há interação, um conjunto (possivelmente unitário) de funções de interação. Na definição destas funções estão descritos os parâmetros que programam a sua apresentação e o seu comportamento. São informadas as funções de entrada (comunicação usuário-aplicação) e as de saída (comunicação aplicação-usuário).

Na seção **Formatos de Interação** são definidos os parâmetros de entrada e saída utilizados nas funções de interação. Os formatos indicam os tipos das informações, a direção da comunicação - entrada (Input) ou saída (Output) - e as origens (no caso de saída) ou destinos (no caso de entrada) das informações. Estes formatos funcionam como macros, sendo expandidos no momento da chamada da função de interação.

Nas seções **Regras de Associação de Entrada** e **Regras de Associação de Saída** são definidas as relações existentes entre as produções da **Sintaxe Abstrata & Semântica** e as definições feitas na **Sintaxe Concreta de Entrada** e na **Sintaxe Concreta de Saída**. O projetista de interfaces estabelece estas relações entre as funções de interação e as produções da **Sintaxe Abstrata & Semântica**, indicando onde há necessidade de interação. A escolha das regras poderia ser feita dinamicamente, selecionando-se uma entre as já definidas.

O modelo proposto GRAEDIUS possui as seguintes vantagens:

- modularidade e flexibilidade nas definições de IU (diálogos e apresentações);

- suporte para definição e controle (a tempo de execução) de IU;

- independência entre IU e aplicação, a tempo de compilação (pela alteração da gramática), e entre diálogo e apresentação, a tempo de execução (pela troca dos atributos de apresentação), isto é, uma aplicação pode ter diferentes modos de diálogo e cada diálogo diferentes formas de apresentação;

- independência do método de reconhecimento da gramática e do método de avaliação dos atributos.

O projeto AMADEUS (Ambientes e Metodologias Adaptáveis para o DEsenvolvimento Unificado de Software), ora em desenvolvimento no Instituto de Informática-UFRGS, adota o formalismo de GA para definição e implementação de ferramentas de engenharia de software. Definindo-se a IU através de GRAEDIUS, o ambiente AMADEUS permite a definição do componente computacional e do componente de diálogo através de um mesmo formalismo. O suporte para isto são as ferramentas (avaliadores de GA, tradutores dirigidos por sintaxe, reconhecedores incrementais, entre outras) disponíveis e/ou em desenvolvimento no AMADEUS. A utilização do mesmo formalismo para definição da aplicação e da interface permite a possibilidade de aproveitamento das ferramentas já desenvolvidas.

4. Exemplo

O exemplo a ser apresentado é a simulação de uma simples calculadora, ou seja, os valores resultantes das expressões fornecidas são exibidos quando solicitado pelo usuário. As expressões permitidas podem ser observadas na gramática descrita no apêndice A. Para calcular o valor de uma expressão, são necessárias as seguintes ações :

a) entrada da expressão a calcular.
b) solicitação da exibição do resultado.

Uma vez exibido o resultado, pode-se entrar com uma nova expressão ou encerrar as tarefas.

A interface proposta para esta calculadora prevê três estilos de interação:

- linguagem de comandos para:
  a) mudança da apresentação física da IU (comando "**apresenta**");
  b) exibição de resultados (comando "**resultado**");
  c) encerramento das tarefas (comando "**fim**").

- menus para a escolha da forma de apresentação dos resultados;

- form-fill (preenchimento de formulários) para a entrada da expressão.

Como apresentação da IU entende-se a sua apresentação física, isto é, os atributos que são utilizados na sua visualização. Alguns destes atributos são, por exemplo : as cores, os fontes de letras, os padrões das linhas, tamanho e coordenadas das janelas, etc.

A seguir apresentamos um trecho da gramática gerada para a calculadora. Apesar de imcompleto, o exemplo é considerado suficiente para ilustrar a proposta. A definição completa da calculadora é encontrada em [PIM 90].

.
.
.

```
Sintaxe Abstrata & Semântica :      /* componente da aplicação */

001 . <calculadora> ->   <inicial>
                         {preparação.apres   = inicial.apres}
                             <preparação>

002 . <inicial>     ->   <épsolon>
                         {inicial.apres      = "apresi"}

003 . <preparação>  ->   <operação>
                         {operação.apres     = preparação1.apres}
                             <preparação>
                         {preparação2.apres  = operação.apres}

004 . <preparação>  ->   <fim>

005 . <operação>    ->{apresenta.apres   = aperação.apres}
                             <apresenta>
                         {operação.apres     = apresenta.apres
                          resultado.apres    = apresenta.apres}
                             <resultado>

006 . <operação>    ->   <expressão>
                         {resultado.apres    = apresenta.apres}
                             <resultado>     /* calcula */

007 . <fim>         ->   fim                /* encerra */

008 . <apresenta>   ->   apresenta          /* solicita troca   */
                                            /* de apresentação  */
009 . <resultado>   ->   resultado          /* mostra resultado */

010 . <resultado>   ->   <épsolon>
```

.
.
.

```
Sintaxe Concreta de Entrada : /* associando às produções    */
                              /* técnicas de interação de  */
                              /* entrada                   */

001 . <preparação>  -> FORM-FILL(jan0,preparação.apres,
                                 lin1,col3,Msg1,
                                 lin3,col1,Input1 )
002 . <apresenta>   -> MENU(jan4,apresenta.apres,
                            tab_apres.nome_apres,
                            Input2)

Sintaxe Concreta de Saída : /* associando à produção a */
                            /* exibição da saída      */

001 . <fim>         -> EXIBE(jan1,fim.apres,
                             lin2,col3,Msg2 )
002 . <resultado>   -> EXIBE(jan3,resultado.apres,
                             lin1,col3,Msg3,
                             lin_corr,col_corr,Msg4 )


Formatos de Interação : /* Informa :                      */
                        /* tipos de parâmetros           */
                        /*    direção de comunicação     */
                        /*    origens e destinos         */
                        /*          das informações      */

Msg1 = String,Output,">"
Msg2 = String,Output,"Término normal de execução"
Msg3 = String,Output,"O valor da expressão é : "
Msg4 = Real,Output,resultado.valor

Input1 = String,Input,Parser_Buffer
Input2 = String,Input,apresenta.apres


Regras de Associação de Entrada :
    /* <sintaxe abstrata> -> <sintaxe concreta de entrada> */

001 . <preparação>/003    -> <preparação>/001

002 . <apresenta>/008     -> <apresenta>/002


Regras de Associação de Saída :
    /* <sintaxe abstrata> -> <sintaxe concreta de saída>   */

001 . <fim>/007           -> <fim>/001

002 . <resultado>/009     -> <resultado>/002
```

A tabela relacional *tab_apres*, utilizada na produção 2 da Sintaxe Concreta de Entrada, possui tuplas correspondentes às definições de apresentação. Adicionalmente são definidas as tabelas *tab_cores*, *tab_janelas* e *tab_fontes*, que contém

respectivamente informações relativas às cores, janelas e fontes que podem ser usadas numa definição de apresentação. Tem-se então tabelas com a seguinte definição :

```
                    .
                    .
                    .
Domínio : boolean   status;
Domínio : char(30)  nome_apres,op_ativas,op_desat,alerta;
Domínio : char(10)  nome_jan;
Domínio : num(02)   f_norm,l_norm,f_rev,l_rev,num_lin,num_col;
Domínio : num(03)   coord_x,coord_y;
Domínio : num(05)   id_cor,id_fonte;

Tabela : Tab_Apres   = (nome_apres,id_cor,nome_jan,id_fonte);
Tabela : Tab_Cores   = (id_cor,f_norm,l_norm,f_rev,l_rev);
Tabela : Tab_Janelas = (nome_jan,status,num_lin,num_col,
                        coord_x,coord_y);
Tabela : Tab_Fontes  = (id_fonte,op_ativas,op_desat,alerta);
                    .
                    .
                    .
```

A sintaxe da função MENU é:

MENU(janela, apresentação , origem , destino)

onde :

janela é o nome de identificação da janela onde ocorrerá a apresentação do MENU;
apresentação identifica a apresentação usada;
origem indica a origem das opções do MENU;
destino indica qual a variável que receberá o valor de retorno da função.

A função MENU exibe ao usuário um menu cujas alternativas dependem do tipo de parâmetro de origem informado. Se o tipo for um campo de uma tabela, então serão fornecidas como opções os distintos valores deste campo. Caso contrário, as opções são fornecidas como conjunto de informações, onde cada opção do menu é definida como uma informação de saída na seção Formatos de Interação. Um exemplo de definição de um conjunto de opções pode ser:

```
Cjo_Opções = Msg7 / Msg8 / Msg9 / Msg12
```

onde :

Msgi é uma referência a uma opção do menu;
"/" é o separador entre as opções;
Cjo_Opções é o nome de referência para um conjunto de opções que é usado como parâmetro da função MENU.

No exemplo, a função MENU na produção 2 da Sintaxe Concreta de Entrada, exibe ao usuário um menu cujas alternativas são os distintos valores do campo *nome_apres*, da tabela

tab_apres, que correspondem às apresentações disponíveis, no caso apenas as apresentações *apres1* e *apres2*. Esta interação dá-se na janela *jan4* e utiliza-se da apresentação cujo nome está na variável *apresenta.apres*. A opção escolhida será armazenada na variável indicada por *Input2*, a saber, a variável *apresenta.apres*.

A sintaxe da função FORM_FILL é:

FORM_FILL (janela , apresentação ,
[[lin,col,origem]* , [lin,col,destino]+]+)

onde :

janela é o nome de identificação da janela onde ocorrerá a apresentação do FORM_FILL;
apresentação identifica a apresentação usada;
origem indica o formato de origem das informações mostradas no FORM_FILL;
destino fornece o nome da variável que receberá o valor informado pelo usuário.
lin,col são coordenadas da janela.

A função FORM_FILL utilizada na produção 1 da **Sintaxe Concreta de Entrada**, exibe ao usuário um formulário onde são apresentados uma mensagem e uma área para a entrada de dados do usuário. No exemplo, a interação será processada na janela *jan0*, utilizando-se da apresentação indicada pela variável *preparação.apres*. A mensagem exibida é um "prompt" (">", símbolo maior) e a entrada digitada pelo usuário será associada pelo formato *Input1* ao "buffer" de entrada do reconhecedor, através da palavra reservada *Parser_Buffer*.

A sintaxe da função EXIBE é:

EXIBE (janela , apresentação , [lin,col,origem]+)

onde :

janela é o nome de identificação da janela onde ocorrerá a apresentação da interação;
apresentação identifica a apresentação usada;
origem indica uma lista de formatos de origem das informações mostradas na interação;
lin,col são coordenadas da janela.

A função EXIBE mostra ao usuário as mensagens definidas na seção **Formatos de Interação**. A janela e a apresentação utilizados são informados, respectivamente, como primeiro e segundo parâmetros. No exemplo, EXIBE é usada nas produções 1 e 2 da **Sintaxe Concreta de Saída** para apresentar, respectivamente, mensagens de fim de sessão e o valor resultante da expressão fornecida. As palavras reservadas *lin_corr* e *col_corr* correspondem às coordenadas correntes do cursor.

A linguagem de comandos é o estilo de interação "default" de GRAEDIUS, ou seja, todo corpo de produção que consiste de um terminal é reconhecido como um comando. Portanto, esta técnica de interação não precisa ser definida na seção **Sintaxe Concreta de Entrada**, pois todo comando faz parte da **Sintaxe Abstrata** e seu reconhecimento é naturalmente tratado.

No exemplo, a recepção da digitação do usuário se dá através de um Form_Fill (produção 1 da **Sintaxe Concreta de Entrada**) e sua passagem ao reconhecedor é direta, através do uso do "buffer" de entrada. As entradas possíveis são os comandos : apresenta, resultado, fim e os dígitos e operadores da expressão a ser calculada.

Algumas das funções apresentadas não podem ser diretamente definidas a partir dos MRIUs citados na seção 2 do artigo. Entre estas, inclue-se a capacidade de alterar dinamicamente os atributos de apresentação e as regras de associação (de entrada/saída). Para suportar esta capacidade, inerente à GRAEDIUS, outros MRIUs precisam de extensões ou de "malabarismos" descritivos. Algumas alterações aplicadas a um MRIU podem tornar sua utilização mais complexa, talvez até inviável.

Um MRIU é difundido se sua notação é simples de aprender e fácil de usar. Em geral, quanto maior o poder descritivo de um MRIU, maior é a complexidade de sua notação, o que dificulta seu uso e aprendizado. GRAEDIUS atenua esta dificuldade configurando-se como um formalismo bastante simples, fácil de usar e, no entanto, com grande poder descritivo.

5. Conclusões

Foi apresentada, em linhas gerais, uma proposta de definição de interfaces através de gramática de atributos. As principais características do modelo GRAEDIUS são a independência entre as especificações de sintaxe abstrata e semântica, sintaxe de entrada, sintaxe de saída, formas de apresentação e técnicas de interação. Um exemplo comentado ilustrou uma definição de interface através de GRAEDIUS.

Um primeiro protótipo desenvolvido sobre o reconhecedor/avaliador sintático e semântico do projeto AMADEUS demonstra a viabilidade prática do mecanismo proposto. Estudos e experimentações adicionais são necessários para analisar o desempenho operacional, bem como a conveniência para o analista, para o projetista de interface e para o usuário final, desta forma de definição da interação usuário-aplicação.

6. Bibliografia

[ESP 89] ESPERANÇA,L.G.; FAVERO,E.L.; PRICE,R.T. "Um Algoritmo de Reconhecimento Incremental". Porto Alegre,CPGCC/UFRGS,1989. Relatório de Pesquisa 117.

[FAV 87] FAVERO,E.L. "A Implementação de um Núcleo Gerador de Editores Dirigidos por Sintaxe em um Microcomputador". Porto Alegre, PGCC da UFRGS,1987.

[GRE 86] GREEN,M. "A Survey of Three Dialog Models". ACM Trans.Graph.,Jul. 1986.

[HOO 89] HOOVER,R. "Attribute Grammars : Their Use and Evaluation". Comunicação Pessoal (Workshop de Ambientes de Programação Dirigidos por Sintaxe, Valinhos, 23 a 25 de agosto de 1989)(não publicado).

[HOR 86] HORWITZ,S. & TEITELBAUM,T. "Generating Editing Environments Based on Relations and Attributes". ACM TOPLAS,Oct. 1986.

[HUD 86] HUDSON,S. & KING,R. "Implementing a User Interface as a System of Attributes". ACM Computer Graphics, Dec. 1986

[MAR 73] MARTIN,J. "Design of Man-Computer Dialogues". Prentice-Hall, Englewood Claffs,N.J.,1973.

[MYE 89] MYERS,B. "User Interface Tools : Introduction and Survey". IEEE Software Eng.,Jan. 1989.

[PIM 90] PIMENTA,M.S.; SCHUBERT,E.G.; PRICE,R.T. "O Uso de Gramática de Atributos para a Definição de Interfaces com o Usuário". Porto Alegre,CPGCC/UFRGS,1990. Relatório de Pesquisa (não publicado).

[REP 84] REPS,T. & TEITELBAUM,T. "The Synthesizer Generator". SIGPLAN Notices, May 1984.

[SHN 87] SHNEIDERMAN,B. "Designing the User Interface : Strategies for Effective Human-Computer Interaction". Addison-Wesley,Reading,Mass.,1987.