XIV Conferencia Latinoamericana de Informática
17avas. Jornadas Argentinas de Informática
e Investigación Operativa.
Buenos Aires, Setiembre 1988.

EDITORES DIAGRAMATICOS BASEADOS EM FORMALISMOS GRAMATICAIS

Roberto Tom Price **,+ e Eloi Luiz Favero *,+

SUMARIO

Este trabalho apresenta a idéia de construir geradores de editores para linguagens diagramáticas (editores diagramáticos) partindo do aproveitamento das técnicas utilizadas na construção de geradores de editores textuais com conhecimento da sintaxe da linguagem.

É apresentada uma comparação á nível operacional entre os dois modelos de editores (diagramático e textual) destacando-se as diferenças.

Como resultado da comparação é sugerida uma extensão, no mecanismo de especificação da sintaxe da linguagem, a fim de incorporar a idéia de nodos compartilhados na árvore sintática abstrata.

Este trabalho foi desenvolvido com apoio financeiro da CNPq, SID-Informática e FINEP.

---

* Bacharel em Ciências de Computação (UFRGS/RS-1987), Mestrando em Ciências da Computação (UFRGS/RS);
** Professor e Pesquisador (UFRGS/RS), Eng, MSc, DPhil.
+ Curso de Pós-Graduação em Ciências da Computação, UFRGS, Caixa Postal 1501, Porto Alegre, RS.

1. Motivação

O objetivo de um ambiente de desenvolvimento de desenvolvimento de software (ADS) é de proporcionar um ambiente de construção de software uniforme e integrado. Este ambiente deve prover ferramentas para a execução das tarefas associadas a cada etapa do ciclo de desenvolvimento de software.

ADSs são baseados em um conjunto de ferramentas diagramáticas e/ou textuais.

Uma ideia atraente para a criação de ferramentas, que manipulam linguagens, em ADSs é a geração das ferramentas a partir de descrições (formais ou semi-formais) das linguagens.

Para ferramentas textuais, já, existem diversos sistemas que geram editores interativos a partir de descrições das linguagens[Rep 84],[Don 84],[Sne 85], [Pri 84].

Estes editores chamados de editores com conhecimento de sintaxe (ECS), podem incorporar várias facilidades como edição por templates, formatação, verficação semantica por gramáticas de atributos e outros.

Para ferramentas diagramáticas não existe o mesmo avanço técnico, i.e. mecanismos geradores de editores diagramáticos a partir das especificações das linguagens.

Em princípio é possível encontrar uma representação textual para qualquer linguagem diagramática. Para a representação textual é possivel gerar um ECS, com o uso dos sistemas mencionados acima.

Quais são as modificações/extensões destes sistemas (geradores de EDSs) para gerar um editor diagramático ?

Este trabalho é um passo inicial na ideia de criar geradores de editores diagramáticos, partindo do aproveitamento das tecnicas utilizadas na geracao de editores textuais.

2. Estrutura do trabalho

No item 3 são sumarizados os principais conceitos relacionados com geradores de editores com conhecimento da sintaxe da linguagem (ECS). Os conceitos servem de base para a comparação entre os modelos de editores: textual e diagramático.

No item 4 a forma operacional do editor ECS e do editor diagramático (ED) é comparada; a discussão e centrada em exemplos de editores (textual e diagramático) para linguagem de diagramas de fluxo de dados (DFD). No item 5 são sumarizadas as diferenças.

No item 6 é apresentada uma sugestão de extensão na meta linguagem de especificação, a fim de permitir nodos compartilhados.

Por fim, no item 7 é redefinida a gramática livre de contexto (GLC), que especifica o DFD textual, a fim de tornar a representação interna do DFD mais apropriada para a linguagem diagramática do DFD.

Todas as figuras referenciadas no texto, exceto a figura 7, estão no final do artigo.

3. Geradores de EDSs uma visão geral

Neste item são apresentados rapidamentes os principais conceitos relacionados com geradores de editores de linguagens, servindo com base para a discusão que se segue.

Inicialmente os editores podem ser classificados em editores de propósito geral (sem conhecimento de qualquer aspecto da linguagem) e **editores com conhecimento** especifico de alguns aspectos de uma ou mais linguagens.

Sob o ponto de vista da estrutura de representação interna do texto, em um editor com conhecimento da sintaxe da linguagem (ECS), diferencia-se de um editor de propósitos gerais por possuir a representação interna em forma de árvore sintática abstrata (ASA), guardando as informações da estrutura sintática.

Todas as operações de edição num ECS são executadas em função do **foco de atenção** definido pela posição do cursor; antes de executar uma operação de edição o usuário se posiciona em um ponto do texto e o sistema gera um menu de operações de edição, em função da posição do cursor.

Com os ECS surgem os conceitos de **sintaxe concreta** (representação do documento na forma textual) e **sintaxe abstrata** (representação interna do documento na ASA)[Don 84].

A representação do texto na árvore sintática abstrata é definida a partir da gramática livre de contexto (GLC). Todo nodo da ASA está associado a um não terminal ou terminal da GLC.

O processo de submissão de texto ("parse") consiste numa transformação da sintaxe concreta para a abstrata e o processo de visualização ("unparse") na transformação da abstrata para concreta.

O termo ECS ou EDS é utilizado usualmente para editores que incorporam também conhecimento da semântica estática e/ou de outras facilidades como por exemplo interpretação.

Algumas implementações de ECS são especificas para uma linguagem, como por exemplo "The Cornell Program Syntethizer" [Tei 81]. Muitas implementações são baseadas em tabelas, as quais definem os aspectos do conhecimento da linguagem. Com a substituição das tabelas para diferentes linguagens são gerados diferentes editores específicos. Surge assim o conceito de geradores de ECS.

Os geradores de ECS utilizam metas linguagens para especificação dos diversos aspectos que definem a linguagem [Ali 83][Don 84][Rep 84]. Diversos níveis de conhecimento são especificados na meta-linguagem. Para cada nível exitem notações e mecanismos associados, suscintamente mencionados abaixo:

a) nível sintático

concreta (representação textual para submissão e visualização):
- variante de BNF (Backus Naur Form/Backus Normal Form)

abstrata (representação interna):
- phila/operator[Don 84][Rep 84]
- CLASS, LIST, NODE [Sne 85][Bah 86]
- anotações na BNF [Fav 88]

formatação (especificação da apresentação do teste):
- associada a concreta ou abstrata

auxílios a submissão: menus e templates
- gerados automaticamente da abstrata ou
- associados à abstrata

b) semântica estática (verificação de tipos, definições de variáveis, etc)
- ações semânticas anotadas na abstrata
- gramática de atributos associada a abstrata

c) interpretação (semantica da execução)
- associda a sintaxe abstrata (caminhamento na ASA)
- compilação e execução de fragmentos (gramática de atributos + ações semânticas)

d) depuração (interpretação controlada)
- associada a interpretação

3.1 Forma geral de operação

O algoritmo de um EDS consiste num ciclo de transformações de sintaxe concreta para abstrata ("parse") e de abstrata para concreta ("unparse").

3.2 O processo de submissão do texto ("parse")

É responsável pela edição do documento (criação/alteração dos nodos da ASA). Toda operação se faz em função da posição do cursor.

Na implementação de ECS, quanto a forma de submissão de texto ("parse") existem duas abordagens: a edição direta do texto (reconhecimento incremental [Sch 84] [Ghe 79]) e a edição auxiliada por menus e "templates" [Tei 81][Med 82].

O reconhecedor incremental é gerado automaticamente a partir a associação entre a sintaxe concreta e abstrata.

Os menus em um EDS são gerados em função da posição do cursor sobre a ASA e pela classificação dos elementos sintáticos como CLASS (menu), NODE("template"), LIST(repetição).

Os comandos de edição baseados em menus, gerados a partir da sintaxe da linguagem, garantem a consistencia a nível sintático.

O "Syntethizer Generator" utiliza o formalismo de gramática de atributos para gerar um mecanismo que executa a análise semântica incremental. Este mecanismo opera sobre árvore sintática, enriquecida com atributos que são associados a cada nodo da ASA. O valor de um atributo representa uma propriedade

especifica de um nodo da ASA. Toda vez que uma operação de edição altera nodos da ASA as equações semânticas associadas aos nodos com valores de atributos afetados são reavaliadas [Rep 83][Rep 84].

3.3 O processo de visualização ("unparse")

Este processo é responsável pela visualização do texto na tela (em torno da posição corrente do cursor). Para tal é necessário manter uma relação entre os tokens terminais na tela com a sua posição na ASA. Cada elemento terminal da tela (sintaxe concreta) está internamente representado na ASA(sintaxe abstrata); um posicionamento (movimento do cursor) sobre o elemento na tela, implica num posicionamento sobre o elemento na ASA. Este posicionamento pode ser por teclas de função ou com a utilização de "mouse".

O processo de visualização pode executar os comandos de navegação (mudança do foco de atenção), auxilios a visualização (ex. alterar o número de caracteres de endentação, "holophrasting").

4. Editor de DFD textual VS diagramático

A figura 1 mostra um DFD representado diagramaticamente. Na figura 2 está representado na forma textual o mesmo DFD.

Para esta representação textual do DFD é possivel gerar um EDS, utilizando um gerador de EDSs, como o "sinthesizer generator" [Rep 84] ou EDS-UFRGS [Pri 87].

A representação textual é definida pela GLC da figura 3. Esta GLC especifica precisamente quais são as sentenças sintaticamente válidas para o DFD. A verificação da semantica estática pode ser especificada pelo formalismo de gramática de atributos (GA).

As mensagens de erros gerados pelo mecanismo de GA são apresentadas na tela como mostra a figura 4 e, a cada ciclo de execução de um comando as mensagens são atualizadas.

Admitindo que a representação interna do DFD textual, em forma de ASA,é adequada para um editor diagramático, para mostrar que é possível a criação de editores diagramáticos partindo das técnicas de geração de editores textuais, é suficiente mostrar que é possível executar as operações relacionadas com a submissão (reconhecimento - "parse") e a visualização ("unparse") dos objetos diagramáticos representados na ASA.

A análise semântica, executada pelo formalismo de GA, pode ser utilizada também para o DFD digramático, pois a representação interna do DFD diagramático é também uma ASA. A notação utilizada para a especificação da análise semântica estática para os ECSs aplica-se de modo direto para os EDs.

As quatro operações de edição, que seguem, exemplificam a forma de interação no ECS vs a forma de interação no ED:

i) criação do depósito "D1" (entidades e processos são criados da mesma forma).

ii) criação do fluxo "a" entre a entidade "E1" e o processo "P2".

iii) remoção do fluxo "a".

iv) remoção da entidade "P1"

As figuras 5a a 5d exemplificam uma possível forma de interação para o editor de texto do DFD. Este editor de texto para DFD foi criado apenas para comparação com o editor diagramático de DFD; a representação diagramática de um DFD é mais expressiva e natural.

i) Posiciona-se o "mouse" sobre o "placeholder" {lista_de_depósitos} e digita-se "D1" (figura 5a-5b).

ii) Posiciona-se o "mouse" sobre o {fluxo} e digita-se a frase "a: E1 --> P2" (figura 5b-5c).

iii) Posiciona-se o "mouse" sobre a frase "a: E1 --> P2" e aciona-se a função "REMOVE" (figura 5c-5b).

iv) Posiciona-se o "mouse" sobre a frase "P1" e aciona-se a função "REMOVE" (figura 5c-5d).

As figuras 6a a 6e exemplificam uma possível forma de interação para editores diagramáticos de DFD

i) Seleciona-se o "placeholder" {depósito} do menu; o elemento e levado para dentro da janela de edição na posição desejada; e, a seguir, digita-se o seu identificador "D1" (figura 6a-6b).

ii) Apontamento de "E1" (6b-6c), seguido do apontamento de "P2" (6c-6d). Para a criação do arco segue-se o processo anterior (i). Por fim o editor deve associar os elementos apontados e o arco com um novo fluxo (6d-6e). Entende-se por apontamento: posiciona-se o mouse sobre o elemento visado e ativa-se a função "APONTAR" (mais adiante no artigo é definida a semântica desta função).

iii) Posiciona-se o mouse sobre o arco, elemento gráfico que identifica o fluxo na tela, e tecla-se "REMOVER" (6e-6b).

iv) Posiciona-se sobre o processo e tecla-se "REMOVER" (figura 6e-6f). No DFD textual a existência de fluxos que referenciam processos não definidos não cria maiores problemas. Porem, para o DFD diagramático não tem sentido a existência de um fluxo (arco) que referencia um processo inexistente. Assim parece normal remover juntamente com o processo todos os fluxos que o referenciam.

5. Comparação entre edição textual e diagramática

As diferenças entre EGs e EDSs surgem em função dos elementos primitivos e, principalmente, pela inexistência, nos EGs, do conceito de ordem na escrita do texto.

5.1 Elementos primitivos

Os elementos primitivos (nível léxico) para o ED são objetos gráficos, enquanto que no EDS são tokens de caracteres.

Para a especificação dos tokens léxico em EDSs é utilizada uma notação do tipo gramática regular.

Para especificar os elementos primitivos gráficos é necessário um meta editor gráfico (MEG) [Mel 88]. O meta editor gráfico permite a especificação de elementos gráficos compostos de elementos geométricos primitivos (circulo, polignos, etc.), com rótulos (textos) associados.

Para ligação entre o nivel sintático e léxico é criado um cabeçalho para todo elemeno gráfico. No cabeçalho são especificados atributos herdados ou sintetizados; um elemento gráfico primitivo pode ser visto como um nodo da GA.

5.2 Ordem de escrita do texto Vs caminhamento em pré-ordem na ASA

Nos EGs a ordem dos tokens na tela não tem relação com a estrutura da ASA; ao contrario no EDSs a ordem de escrita do texto está relacionada com um caminhamento em pre-ordem na ASA. Este conceito de ordem tem duas implicações:

5.2.1 A geração de menus

No EDS do DFD (textual) os "placeholders" são inseridos no meio do texto, na posição definida pela formatação, na ordem de escrita do texto.

Para o ED de DFD, como o conceito de ordem não existe, usualmente os "placeholders" (como primitivas gráficas) são deixados no menu lateral da janela de edição (figura 6). A seleção do "placeholder" do menu no ED corresponde ao posicionamento sobre o "placeholder" no EDS, que por sua vez corresponde a um posicionamento sobre a ASA. Após a escolha o usuário traz o elemento para a janela de edição - isto é define a posição do elemento na página.

5.2.2 O processo de visualização

A formatação para EDSs é gerada automaticamente (caminhamento em pre-ordem) a partir da sua especificação associada a sintaxe abstrata. No ED o processo de visualização

guarda a posição (explicitada pelo usuário no momento da criação) de cada elemento gráfico primitivo.

O processo de visualização exibe os objetos que são referenciados na ASA, como nodos da janela de edição corrente. A janela corrente usualmente está associada a uma página; o conteúdo da página é definido explicitamente pela edição dos elementos gráficos. Nos EDSs o formatador divide automaticamente o texto em páginas, pelo caminhamento na árvore.

No ED o processo de visualização é programado (não é automático) para executar o comando de troca de página. Por exemplo para o ED do DFD a troca de página é necessária para a explosão (refinamento) de um processo; a página contém um nível do DFD, sem os refinamentos.

O comando de troca de página necessita de um posicionamento do mouse sobre um processo, seguido da ativação da função "NOVA_PAGINA" (equivale a digitação de SUB_DFD "Pi" no editor textual).

Uma alteração da posição de um elemento gráfico primitivo ou uma troca de página não afeta a estrutura da ASA.

6. Nodos compartilhados Vs apontamento

Para um ED é natural a escolha de elementos (apontamento) que estão na tela para a composição de novos elementos (operação ii). Em linguagens textuais esta forma de referenciar elementos é feita pela escrita de um identificador associado a unidade que é referenciada (por ex. o a explosão do processo "P1" da figura 1 é indicada pela rescrita do identificador "P1" após a palavra reservada SUB_DFD).

Esta forma de composição por apontamento de elementos introduz uma idéia de compartilhamento de nodos na ASA. sugerindo uma extensão da GLC para permitir nodos compartilhados na ASA (fig 7).

```
Fluxo --> Elementol : TipoElemento
          Elemento2 : TipoElemento
          Arco_gr.

TipoElemento --> Atividade | Deposito | Processo.

...
```

figura 7: especificação do fluxo (nodos compartilhados)

A notação "Elementol : TipoElemento" denota que o Elementol é do tipo TipoElemento, porém não é um nodo e sim uma referência para um nodo do TipoElemento. Uma instância deste tipo de nodo é criada pela operação de apontamento. A operação de apontamento é executada em duas etapas: a) pela marcação dos nodos na tela e, b) ligação com um nodo sintático. Na figura 6 o menu de "apontamento" mostra quais são os elementos que podem ser apontados.

No caso de utilização do mecanismo de GA, os nodos compartilhados não podem possuir atributos herdados, somente

sintetizados. Uma alteração nos atributos do nodo compartilhado implica numa reavaliação de todos os nodos que o referenciam.

7. A redefinição da GLC para o ED de DFD

Na figura 8 é apresentada uma GLC mais apropriada para o editor diagramático. Esta GLC foi especificada considerando os aspectos particulares de edição nos EGs.

Nesta gramática o refinamento de um processo é associado ao própio processo na ASA. A hierarquia de decomposição de processo é representada de forma mais natural nesta GLC.

Na GLC da figura 3 a consistência de que o refinamento é válido somente para processo é feita no nível semântico, nesta GLC a a mesma consistência passa para o nível sintático.

8. Conclusão

O DFD é uma linguagem baseada no conceito de grafo. O conceito de grafo na notação proposta neste trabalho é representado com a utilização de nodos compartilhados.

Uma alternativa a esta notação é a geração de ED a partir de formalismos que possuem (embutido) o conceito de grafo. Não existe ainda um formalismo, baseado no conceito de grafo, que permite (na prática) a geração de ED. Para o DFD a notação de nodos compartilhados parece não apresentar maiores inconvenientes.

Um argumento favorável a esta notação é o conceito de decomposição (refinamento), associado a uma estrutura de árvore hierarquica, presente em quase todas as linguagens (diagramáticas ou textuais). Este conceito é expressado naturamente em árvores (como estruturas sintáticas).

Outro argumento favorável é a utilização do mecanismo de gramática de atributos para executar a análise semântica de forma incremental; possibilitando a especificação dos aspectos de semântida da linguagem diagramática em uma notação declarativa.

Uma vantagem na utilização do mesmo formalismo (ainda que estendido) para a geração de ediores textuais e diagramáticas esta na compatibilidade da representação interna de documentos diagramáticos e textuais, facilitando a construção de de ferramentas integradas para os ambientes de dessenvolvimento de projetos em geral.

Os autores estão desenvolvendo um protótipo de um gerador de editores diagramático (ED), baseado nas idéias expostas neste artigo. O protótipo está sendo construido pela adaptação de um gerador de editores dirigidos por sintaxe com um meta editor gráfico [Pri 84][Mel 88].

9. Referências

[All 83] Allison, Lloyd; "Syntax Directed Program Editing". Software Practice And Experience, vol. 13, 453-465 (1983).

[Bah 86] Bahlke, Rolf e Snelting, Gregor; "Context-sensitive editing with PSG environment"; Lecture Notes in Computer Science: "advanced programming environment" no 244, june, 1986.

[Don 84]
Donzeau-Gouge, U.; Kahn, G.; Lang, B. & Melese, B. "Document Structure and Modularity in Mentor". SIGPLAN NOTICES, Vol 19, no 5,pg. 141-8, May 1984.

[Fav 88]
Favero, E.L. e Price R.T. "A Implementação de editores dirigidos por sintaxe - Algumas considerações" (não publicado) UFRGS, março 1988.

[Ghe 79]
Ghezzi C. e Mandrioli D. "Incremental Parsing", ACM TOPLAS, vol 1, 58-70, 1979.

[Med 82]
Medina-Mora R., "Syntax_directed editing: towards integrated programing environments". Dep. Comp. Sc., Carnegie-Mellon Univ., Pittsburgh, 1982. (Ph.D. dissertation).

[Mel 88]
Melo W.L.M. e Price R.T. "O editor gráfico generalizado". (a ser publicado nos anais do SEMISH, VIII CONGRESSO NACIONAL DA SOCIEDADE BRASILEIRA DE COMPUTAÇAO, Julho de 1988).

[Pri 84]
Price, R.T. "A Prototype syntax driven language editor". Brighton, University of Sussex, 1984. (Tese de Doutorado).

[Pri 87] Price R.T. e Favero E.L. "Introdução a Linguagem de Especificação (LDE)". VII Congresso da Sociedade Brasileira de Computação, 1987.

[Rep 83]
Reps T.; Teitelbaum T. e Demers A. "Incremental Contex-dependent Analysis for Language-based Editors". ACM TOPLAS vol 5, no 3, July 1983.

[Rep 84]
Reps T. e Teitelbaum T., "The synthesizer generation. SIGPLAN NOTICES, vol 19, no 5, 42-8, may 1984.

[Sne 85]
Snelting, G. "Experience with the PSG - Programming System Generator. Lecture Notes In Comp. Sci. 148-162 vol 2: Formal Methods & Softw. Development. Springer-Verlag, 1985.

[Sch 84]
Schwartz M.D.; Delisle N.M. e Begwni V.S. "Incremental compilation in Magpie". SIGPLAN NOTICES vol 19, no 6, June 1984.

[Tei 81]

Teitelbaum T., Reps T. e Horwitz S., "The Why and Wherefore of the Cornell Program Synthesizer", SIGPLAN NOTICES, Vol 16, no 6,pg. 8-16, June 1981.

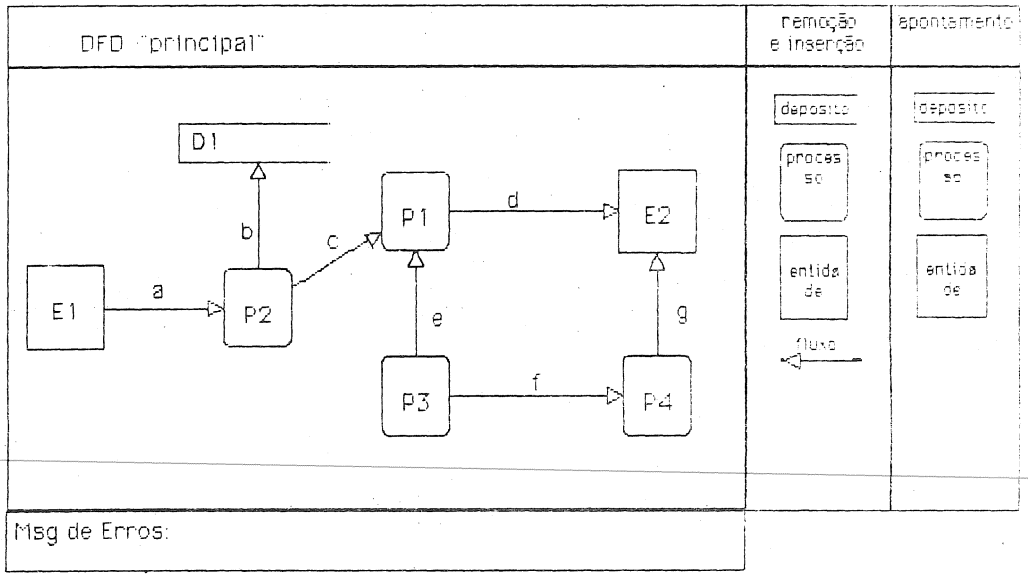

figura 1: representação de um DFD diagramático

```
DFD  principal
   e1, e2 : ENTIDADES;
   d1 : DEPOSITOS;
   p1, p2, p3, p4 : PROCESSOS;
   FLUXOS
         a : e1 ---> p2;
         b : p2 ---> d1;
         c : p2 ---> p1;
         d : p1 ---> e2;
          ...
    END_FLUXOS
    SUB_DFD  p3
          p3.1, p3.2, p3.3 : PROCESSOS;
          d3 : DEPOSITOS;
        FLUXOS
           d3_f1 : p3.2 ---> p3.1;
           d3_f2 : p3.1 ---> p3.3;
            ...
       END_FLUXOS
   END_SUB_DFD p3;
END_DFD principal;
```

figura 2: uma representação textual para o DFD

```
DFD  principal

   e1, e2 : ENTIDADES;

   d1, d1 : DEPOSITOS;  |ERRO1:  nomes de depósitos duplicados |

   p1, p2, p3, p4 : PROCESSOS;

   FLUXOS

         f1 : e1 ---> p2;

         f2 : p3 ---> e1;

         f3 : p3 <--> d2;  |ERRO2:  depósito (d2) nao especificado |

         f4 : d1 ---> e1;  |ERRO3:fluxo sem processo |

          ...

    END_FLUXOS

    SUB_DFD  p3

          p3.1, p3.2, p3.3 : PROCESSOS;

          d3 : DEPOSITOS;

        FLUXOS

           d3_f1 : p3.2 ---> p3.1;

           d3_f2 : p3.1 ---> p3.3;

            ...

                        |ERRO4:  fluxos nao utilizados|
                        |ERRO5:  depósitos sem entras ou  saidas|

       END_FLUXOS

   END_SUB_DFD p3;

END_DFD principal;
```

figura 4: exemplos de mensagens de erros geradas pela mecanismo de gramática de atributos

```
DFD  principal
   e1, e2, {entidade} : ENTIDADES;
   {lista_de_depósitos}
   p1, p2, p3, p4, {processo} : PROCESSOS;
   FLUXOS
         c : p2 ---> p1;
         d : p1 ---> e2;
         ...
         {fluxo}
    END_FLUXOS
    {lista_de_sub_DFDs}
END_DFD principal;
```

figura 5a: um DFD parcialmente editado

```
DFD  principal
   e1, e2, {entidade}: ENTIDADES;
   d1, {deposito} : DEPOSITOS;
   p1, p2, p3, p4, {processo} : PROCESSOS;
   FLUXOS
         c : p2 ---> p1;
         d : p1 ---> e2;
         ...
         {fluxo}
    END_FLUXOS
    {lista_de_sub_DFDs}
END_DFD principal;
```

figura 5b: Após a inclusão do depósito "D1"

```
DFD  principal
   e1, e2, {entidade}: ENTIDADES;
   d1, {deposito} : DEPOSITOS;
   p1, p2, p3, p4, {processo} : PROCESSOS;
   FLUXOS
         c : p2 ---> p1;
         d : p1 ---> e2;
         ...
         a : e1 ---> p2;
         {fluxo}
    END_FLUXOS
    {lista_de_sub_DFDs}
END_DFD principal;
```

figura 5c: Após a inclusão do fluxo "a: e1 ---> p2".

```
DFD  principal
   e1, e2, {entidade}: ENTIDADES;
   d1, {deposito} : DEPOSITOS;
   p2, p3, p4, {processo} : PROCESSOS;
   FLUXOS
         c : p2 ---> p1;
         d : p1 ---> e2;
         ...
         a : e1 ---> p2;
         {fluxo}
    END_FLUXOS
    {lista_de_sub_DFDs}
END_DFD principal;
```

figura 5d: após a remoção do processo "p1"

```
dfd ---> SubProcesso.

SubProcesso  ---> Id_Gr
                  ListaEntidades
                  ListaDepositos
                  ListaProcessos
                  ListaFluxos .

ListaEntidades ---> Entidade_Gr   ListaEntidades.
ListaEntidades ---> Empty.

ListaProcessos ---> Processo ListaProcessos.
ListaProcessos ---> Empty.

Processo ---> Processo_Gr
              [ SubProcesso ]

Fluxo ---> Elemento1:TipoElemento
           Elemento2:TipoElemento
           Arco_GR .

TipoElemento ---> Entidade | Deposito | Processo.
```

figura 8: uma especificação da sintaxe para o DFD diagramático

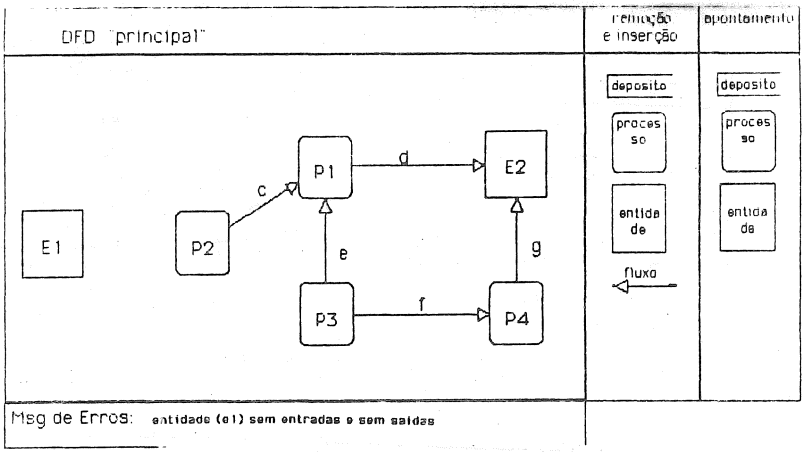

figura 6a: um DFD parcialmente editado

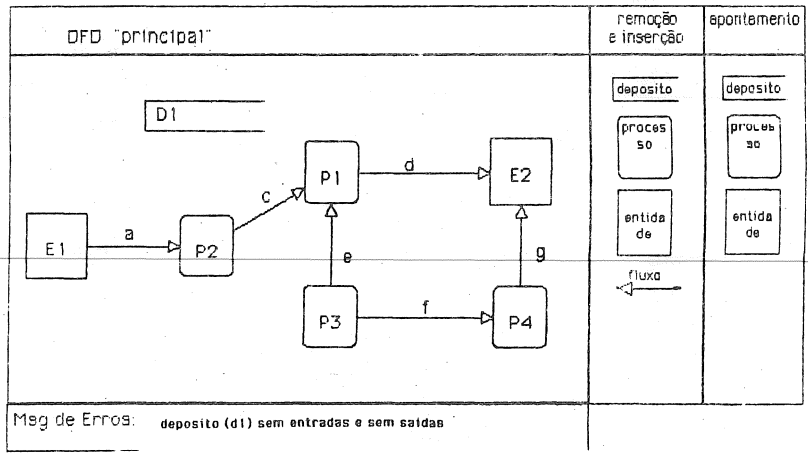

figura 6b: após a criação do depósito "d1"

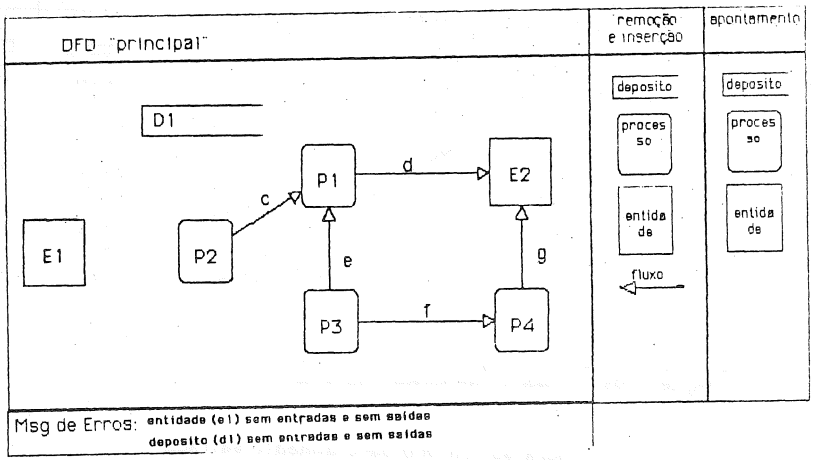

figura 6c: a entidade "E1" está apontada para a criação do fluxo

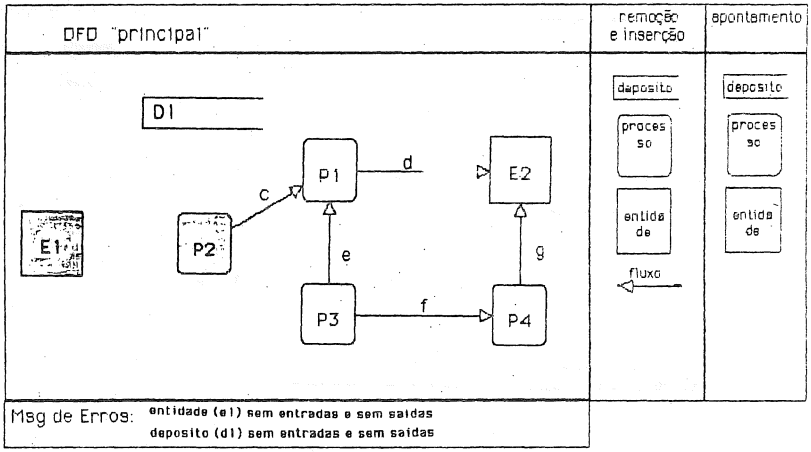

figura 6d: a entidadade "E1" e o processo "P2" estão apontados para a criação do fluxo

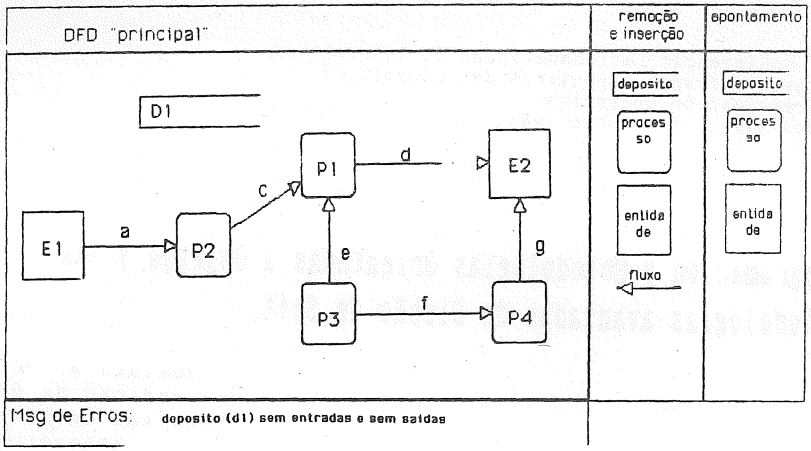

figura 6e: após a criação do fluxo "a"

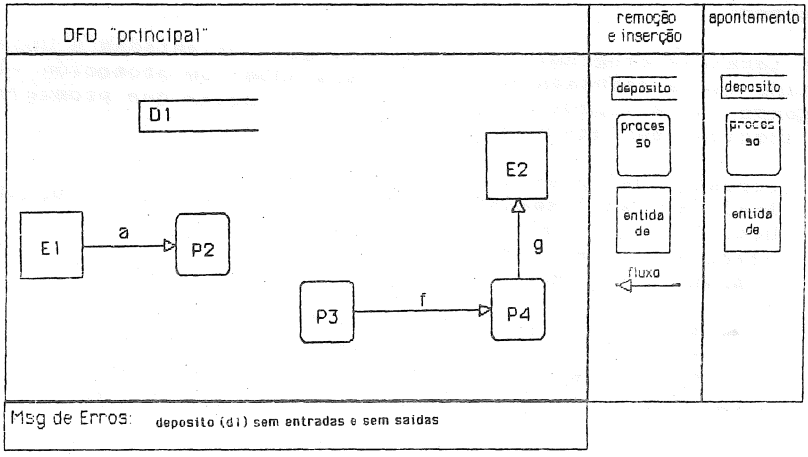

figura 6f: após a remoção do processo "P3"