UM EDITOR HIBRIDO (TEXTO E DIAGRAMAS) ORIENTADO POR ESTRUTURAS (DO TIPO GRAFO E ARVORE) *

Roberto Tom Price, Elói Luiz Favero

Curso de Pós-Graduação em Ciência da Computação
Universidade Federal do Rio Grande do Sul
CP 1501, 90020 Porto Alegre - RS - Brasil
Telex 0512268OCCUFBR e-mail tomprice@UFRGS.ANSP.BR

RESUMO

É apresentado um formalismo para especificação da sintaxe de linguagens textuais e diagramáticas adequado para geradores de ambientes de desenvolvimento de software orientados por estrutura. O formalismo consiste numa notação estendida de gramática livre de contexto permitindo a representação de estruturas de grafo, constituindo-se numa notação alternativa a gramáticas de grafo. A partir do formalismo proposto é apresentado um editor hibrido (para linguagens textuais e diagramáticas) orientado por estruturas.

ABSTRACT

This article introduces a formalism for the specification of the syntax of textual and diagrammatic languages suitable for the generation of structured-oriented environments of software development. The notation extends context-free grammars used for textual languages allowing the representation of graphs, as for diagrammatic languages. The suggested notation is an alternative to graph grammars. The article also introduces a hybrid structured-editor for textual and diagrammatic languages based on the proposed notation.

1. Introdução

Os ambientes de desenvolvimento de software (ADSs) do tipo orientados por estruturas se caracterizam, na taxonomia de [DAR 87], em ambientes que possuem facilidades de edição estrutural (estruturas sintáticas), independentes de linguagens, onde um editor para uma linguagem particular é

---

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

gerado a partir da descrição da sintaxe e semântica da linguagem alvo. Os geradores para estes ambientes são preparadores de tabelas para o editor dirigido por tabelas. As tabelas que descrevem os aspectos de uma linguagem podem ser substituídas para distintas linguagens, gerando desta forma distintos editores.

Existem dois tipos principais de editores orientados por estruturas. O primeiro utiliza estruturas de árvores sintáticas abstratas (ASAs) decoradas com atributos (por ex. PSG[SNE 86], CPS [REP 84], Mentor[DON 84], Gandalf[HAB 86], EDS [PRI 84]) e o segundo faz uso da estrutura de grafo decorada com atributos (por ex. IPSEN [NAG 83], [NAG 87]).

O primeiro tipo está bem desenvolvido tecnicamente. Utiliza as notações de gramática livre de contexto (GLC) e gramática de atributos (GA) para descrição da linguagem. GA para descrever os aspectos dependentes de contexto da linguagem. Esta abordagem está voltada principalmente para linguagens textuais de programação.

O segundo tipo, estruturas de grafo, não apresenta o mesmo amadurecimento técnico, porém a estrutura sintática é mais geral possibilitando tanto a descrição de linguagens textuais quanto de linguagens diagramáticas. O conceito de grafo está presente em muitas linguagens diagramáticas e textuais, principalmente em linguagens de mais alto nível. Muitas das linguagens de especificação de sistemas, como Diagrama de Fluxo de Dados (DFD), diagramas de Entidade e Relacionamentos (ER) são baseadas em grafos. O enfoque estrutura de grafo possibilita a especificação das ferramentas utilizadas na especificação de sistemas.

O formalismo "programmed attributed graph grammars" [ENG 87] variante de gramática de grafo é utilizado no ambiente IPSEN para implementação de ADSs. Neste ambiente foram construídas ferramentas para especificação de requisitos, especificação de projeto, implementação, integração, manutenção e outras atividades relacionadas com software [NAG 87].

Na UFRGS está em desenvolvimento um sistema gerador de

editores do tipo híbrido, para texto e para notações diagramáticos. Com este editor busca-se a criação de ferramentas para as diferentes fases do ciclo de desenvolvimento de software, bem como para diferentes abordagens e\ou metodologias empregadas no desenvolvimento de sistemas, tais como ferramentas para manipulação de descrições de requisitos de sistema, fluxo de controle do sistema em notação diagramáticas, relatórios de análise, especificações em notações formais, programas fontes, código para definição de bases de dados, etc.

Com o objetivo de descrever tanto ferramentas textuais como diagramáticas, dentro do tipo de ambientes orientados por estruturas basedas no conceito de grafo, foi desenvolvida uma notação alternativa à gramática de grafos. Este formalismo utiliza uma gramática livre de contexto (GLC) estendida pelo conceito de nodo compartilhado na árvore sintática abstrata de representação interna. Os aspectos dependentes de contexto (não estruturais) são especificados pelo formalismo de GA associado à descrição sintática.

Esta notação apresenta as seguintes vantagens:

- Permite a especificação de ferramentas diagramáticas e textuais. Por exemplo pode-se com a mesma notação especificar tanto um DFD (diagramático) como um dicionário de dados (textual).

- Permite a integração de documentos textuais e diagramáticos a nível de sua representação interna, utilizando o conceito de grafo e\ou árvore.

- Busca o aproveitamento da tecnologia desenvolvida para os editores orientados por estrutura para linguagens textuais, com pequenas adaptações.

Convém ressaltar que este enfoque, de especificar linguagens diagramáticas por GLCs estendidas e associadas a GA,

foi desenvolvido como resultado do esforço de experimentação e elaboração do projeto Ambiente de Desenvolvimento de Software em andamento na UFRGS. Neste sentido contribuíram particularmente as experiências na construção de editores dirigidos por sintaxe [PRI 84], [FAV 87] e editores diagramáticos generalizados [PRI 88], [MEL 89], [FAV 89].

A notação proposta para linguagens diagramáticas (GLC associada a GA) é ilustrada num exemplo de especificação de um editor de diagramas de fluxo de dados (DFD) na seção 2. Na seção 3 é comentado o formalismo gramatical. Na seção 4 são comentados os aspectos da estrutura de representação interna de documentos associada às operações de edição, e por fim na seção 5 é comentada a construção do editor híbrido.

2. A especificação de uma linguagem diagramática

Nesta seção é especificada uma sintaxe para a liguagem diagramática de fluxo de dados (DFD).

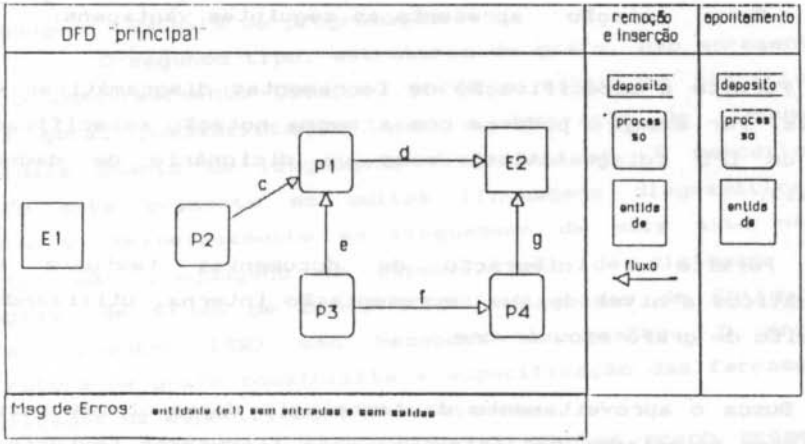

figura 1 - Uma página de um DFD

A partir da figura 1 pode-se observar alguns aspectos estruturais de um DFD:

1. Uma página lógica de um DFD é composta de um identificador (nome de página), um conjunto de entidades (quadrados), um conjunto de processos (bolhas), um conjunto de depósitos (retângulos) e um conjunto de ligações (arcos).
2. Cada entidade, depósito ou processo pode ser considerado como um item léxico, i.e. um objeto fechado para o nível sintático.
3. Uma ligação (fluxo de dados) ocorre entre dois elementos do tipo entidade, processo ou depósito. Um elemento pode participar de várias ligações e em toda ligação deve existir pelo menos um processo.

Numa notação de GLC (BNF) não existe o conceito de conjunto (item 1), porém pode-se descrever um conjunto por uma lista de elementos (sequência) e uma regra semântica que verifica a unicidade dos elementos da lista. Assim, descreve-se "um conjunto de entidades" como:

```
ListaDeEntidades --> Entidade ListaDeEntidades
ListaDeEntidades --> ^
```

Uma página pode ser descrita como:

```
pagina  -->  NomeDaPagina ListaDeEntidades ListaDeDepositos
             ListaDeProcessos ListaDeLigacoes
```

Acrescentando a restrição semântica de unicidade através de GA as produções lista de entidades e página são reescritas abaixo (utilizando só as iniciais dos identificadores para compactação):

```
P  --> !Ng Le Ld Lp Lf
{Le.ConjHerd := Le.ConjSint;}
...
Le --> E  Le
{msgErro := if E.name in Le1.ConjHerd
            then "entidade já definida"
            else  "";
  Le1.ConjSint := Le2.ConjSint + E.name;
  Le2.ConjHerd := Le1.ConjHerd;}
LE  --> ^
   { LE.ConjSint  := φ }
```

O atributo ConjSint "coleta" os atributos name enviando-os para cima na ASA até a produção P; neste produção o

ConjHerd, que armazena todos os atributos name, é passado para baixo; o teste de unicidade é executado sobre o ConjHerd.

O atributo "msgErro" é associado à linha de mensagens de erro da janela de edição (figura 1).

Descreve-se uma ligação (fluxo de dados), assumindo que a orientação do arco é dirigida da ocorrência do primeiro elemento para a ocorrência do segundo elemento, como:

```
Ligacao --> IdFluxo Elemento Elemento
Elemento --> Entidade ! Deposito ! Processo
```

A notação proposta explora de forma simples a possibilidade de compartilhamento de nodos na ASA, permitindo que um operador de compartilhamento (?) prefixado a um elemento da produção indique que a produção referida deva já existir na ASA. Na ASA é criado um ponteiro para cada instância de nodo compartilhado. O exemplo abaixo mostra a utilização da notação proposta para a especificação de uma ligação entre dois elementos que devem já existir na ASA.

```
Ligacao --> IdFluxo Elemento Elemento
Elemento --> ?Entidade ! ?Deposito ! ?Processo
```

Estas produções podem ser lidas como: "uma ligação é gerada pela criação do nodo IdFluxo seguido de dois elementos, sendo que estes elementos são compartilhados na representação interna da árvore" (figura 3).

As verificações semânticas não apresentadas nestas produções, devem ser descritas em GA, como foi mostrado acima na restrição para a unicidade do conjunto.

Na figura 2 é apresentada a descrição da estrutura do DFD.

3. O formalismo gramatical (GLC extendida)

Como foi comentado na introdução, duas abordagens são utilizadas para a descrição de aspectos estruturais (livres de contexto) de linguagens para os geradores de editores. A primeira usa um formalismo compatível com GLCs e a segunda é baseada em gramática de grafos. As regras gramaticais em ambos

os formalismos, podem ser classificadas em cinco tipos (ver por ex. [DON 84], [REP 84] [NAG 87]):

tipo seleção: para seleção de um conjunto finito de alternativas (ex. na descrição acima a produção El-elemento);

tipo opcional: para expressar que uma estrutura é opcional (ex. Osp, na figura 2);

tipo lista: uma ou mais ocorrência de elementos agrupados sob este nodo (ex. Lp, Le, Ld, Lf, na figura 2);

tipo agregação: um nodo é construido pelo agrupamento de diversos elementos constituintes(ex. Sp, F, na figura 2);

tipo elementar: é um item terminal na GLC (ex. !Ng, !Eg, !Dg, !Fg, ?Eg, ?Pg). Existem duas classes de itens terminais: um gera nodos na ASA (!) e o outro gera ponteiros entre nodos da ASA (?), figura 2 e figura 3.

```
%Significado dos identificadores
Sp : sub-processo
Le : lista de entidades
Lp : lista de processos
Ld : Lista de depósitos
Lf : Lista de fluxos
Osp: Sub-processo opcional
El : classe de elementos para apontamento
P  : nao terminal  processo
Pg (processo), Eg(entidade), Fg(Fluxo),
Dg(depósito), Ng(nome de página):símbolos léxicos terminais%

Dfd --> Sp
Sp  --> !Ng Le Ld Lp Lf
Le  --> !Eg Le
Le  --> ^
Ld  --> !Dg Ld
Ld  --> ^
Lp  --> P  Lp
Lp  --> ^
Lf  --> !Fg Lf
Lf  --> ^
F   --> !Fg El El
El  --> ?Eg
El  --> ?Pg
El  --> ?Dg
P   --> Pg Osp
Osp --> Sp
Osp --> ^
```

figura 2-descrição estrutural de um DFD

Todo elemento gráfico (que possui uma representacao concreta na tela) deve estar associado a um tipo elementar, visto como um objeto léxico fechado para o nível sintático. O tipo elementar corresponde a uma folha na estrutura de representação interna.

Normalização

São impostas duas restrições para a descrição das produções:

a) as produções devem definir uma GLC do tipo LL(1) [AHO 86]

b) todas a produções deve estar normalizadas, i.e. podem ser classificada num dos 5 tipos citados acima.

As produções que não podem ser classificados nos tipos descritos acima devem ser reecritas. Por exemplo a produção

(A --> B C, A --> D E, A --> ^)

deve ser reescrita, como por exemplo para as produções abaixo:

(Ao --> A, Ao --> ^,
A --> B C, A --> D E),

a primeira do tipo opcional e a segunda do tipo alternativo.

Extensão semântica para grafos

A extensão para conceito de grafo esta relacionada ao tipo elementar. Arcos são criados pelo compartilhamento de nodos. Cria-se uma relação de um nodo tipo elementar para outro nodo do tipo elementar através da notação de apontamento (?). Neste texto os nodos tipo elementar de apontamento são também chamados de donatários (que recebem uma doação). Os nodos referenciados pelos nodos donatários devem ter sido criados anteriormente.

Na representação interna uma instância de um nodo tipo elementar de apontamento pode ser vista como um ponteiro para o nodo elementar compartilhável que está sendo referenciado (figura 3).

Numa estrutura de grafo a remoção de um nodo implica na remoção de todos os arcos incidentes. Portando numa operação sobre a representação interna que envolva a remoção de um nodo compartilhado deve-se propagar o efeito para os nodos donatários - i. e. remove-se o nodo compartilhado juntamente com todos os nodos donatários.

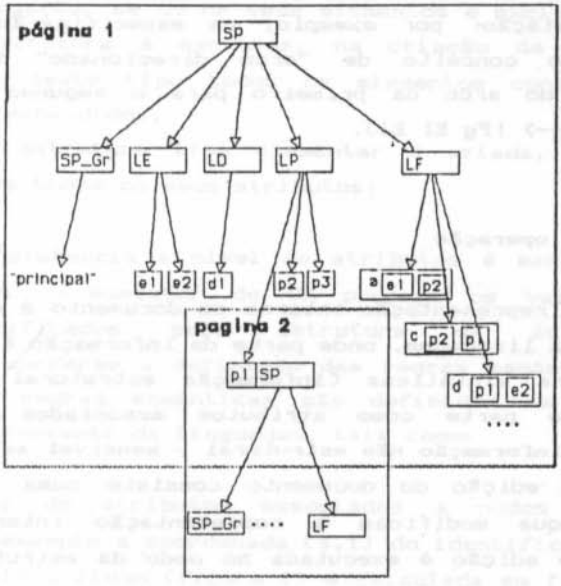

figura 3 - A árvore sintática (como grafo) da figura 1

A nível de consistência estrutural, na semântica para extensão para grafo, criaram-se as regras abaixo:

- é obrigatória a existência do nodo compartilhado (a ser apontado) no momento da criação de um nodo donatário;
- na remoção de um nodo compartilhado remove-se juntamente todos os nodos donatários;

Discutiu-se a extensão da notação GLC para permitir o conceito de nodos compartilhados apenas a nível de elementos terminais da GLC, i.e. apenas podem ser compartilhados elementos terminais. O conceito de compartilhamento aplica-se

também para não-terminais da GLC. Por exemplo em um documento de "hiper-texto" os elementos compartilhados podem ser fragmentos de texto, onde cada fragmento está associado a um não-terminal. A compartilhamento de não-terminais é tema de um trabalho futuro.

A notação proposta apenas introduz o conceito de nodo compartilhado. O conceito de grafo (nodo e ligações) é assumido na interpretação: por exemplo, na especificação apresentada assume-se o conceito de "arco direcionado" (assumindo a orientação do arco da primeiro para o segundo elemento da produção F --> !Fg E1 E1).

4. Forma de operação

A representação interna do documento é definida pela descrição da linguagem, onde parte da informação é representada na estrutura sintáticas (informação estrutural - livre de contexto) e parte como atributos associados a estrutura sintática (informação não estrutural - sensivel ao contexto).

A edição do documento consiste numa sequência de operações que modificam a representação interna. Toda a operação de edição é executada no nodo da estrutura indicado pelo cursor.

São considerados dois tipos de operação: as estruturais e as não estruturais. As operações estruturais modificam a estrutura sintática do documento: a inclusão de um Processo no exemplo da seção 2. As não estruturais modificam os atributos associados a estrutura sintática: a modificação do "nome" do Processo no exemplo da seção 2.

Durante as operações é preservada a estrutura sintática (GLC), bem como o conceito de grafo. A consistência da estrutura sintática é mantida pelas operações de inserção e remoção que são dirigidas pela estrutura sintática, seguindo as regras abaixo (baseadas na classificação das produções):

- para estruturas tipo seleção os elementos a serem exibidos

no menu são as possíveis estruturas alternativas;
- uma estrutura tipo opcional pode ser incluída ou excluída livremente;
- numa estrutura tipo lista pode-se incluir ou excluir livremente (em qualquer ponto da lista) qualquer número de elementos.
- no tipo agregação, se um de seus elementos é excluído então toda a estrutura é excluída; na criação de uma nova ocorrência deste tipo todos os elementos constituintes devem ser expandidos;
- quando uma estrutura tipo elementar é criada, devem ser preenchidos todos os seus atributos;

A consistencia a nível de atributos é mantida pelo executor de GA. O executor de GA propaga os valores dos atributos modificados pela estrutura da árvore de representação, conforme a definição das regras semânticas das produções. Nas regras semânticas são definidos os aspectos dependentes de contexto da linguagem, tais como:

a) a avaliação de atributos associados a nodos do tipo elementar. Por exemplo a coordenada (X,Y) do identificador "a", do elemento gráfico fluxo (figura 1) é calculada em função das coordenadas da entidades "e1" e do processo "p2". Isto é ilustrado nas regras que seguem (o fluxo !Fg herda as coordenadas dos elementos terminais que são por ele ligados)

```
F  --> !Fg E1 E1
       { Fg.ponto1 := E11.ponto;
         Fg.ponto2 := E12.ponto;}
E1 --> ?Eg
       { E1.ponto := Eg.ponto;}
E1 --> ?Dg
       { E1.ponto := Dg.ponto;}
...
```

b) a avaliação de atributos associados a mensagens de erro (análise sensivel ao contexto) são exibidas na janela de edição (exemplo do item 2 - figura 1).

5. Um editor híbrido

A idéia central do editor híbrido é a geração de ferramentas textuais e/ou diagramáticas a partir de descrições num mesmo formalismo gramatical (GLC e GA).

Usualmente, os documentos sao divididos em páginas lógicas. Uma página é uma unidade (porção) conceitual de um documento. A integração entre o editor textual e o diagramático é feita a nível de páginas - uma página é textual ou diagramática. Cada página é descrita por um conjunto de produções.

A associação do conceito de página à definição gramatical pode ser feita pelo prefixo EDE (editor diagramático) ou EDS (editor textual) na produção que está associda a uma página. A figura 4 ilustra a idéia.

```
%Significado dos identificadores
Ps : Pseudo-Código
Lc : Lista de comandos
C  : comando
Var: variavel
com : Comentário
exp : expressão%

start :: Dfd --> Sp
EDE   :: Sp  --> !Ng Ops Le Ld Lp Lf
         ...
         Ops --> Ps
         Ops --> ^
         ...
EDS   :: Ps  --> Lc
         Lc  --> C  ";"  Lc
         C   --> se  exp  entao C senao C
         C   --> begin  Lc  end
         C   --> Var ":=" exp
         C   --> Com
         ...
```

figura 4 - Especificação de um DFD com páginas de diagrama e páginas de texto

Os prefixos EDE e EDS indicam respectivamente a ativação do editor diagramático e do editor de texto. Cada editor é ativado sobre uma página lógica. O EDE e EDS podem ser recursivamente ativados. Pela descrição, acima, quando o EDE expande a produção (Ops --> ps) o EDS é ativado.

A nível de arquitetura, a figura 5 mostra um esboço do núcleo do editor, onde pode-se ver um BD comum para todo o ambiente, um mecanismo de GA para execução das regras semânticas, um módulo que implementa facilidades a nível de sintaxe abstrata (representação interna na ASA baseada na descrição em GLCs estendidas), e dois mecanismos distintos a nível de sintaxe concreta (representação externa) para documentos textuais e para documentos diagramáticos.

A necessidade de dois mecanismos para a sintaxe concreta se deve principalmente a natureza distinta de textos e diagramas: a) Textos possuem o conceito de ordem de escrita que diagramas não tem; b) Um texto consiste em numa composição de caracteres de um alfabeto bem definido; já os elementos terminais de um diagrama são objetos gráficos que podem ter texto como componentes.

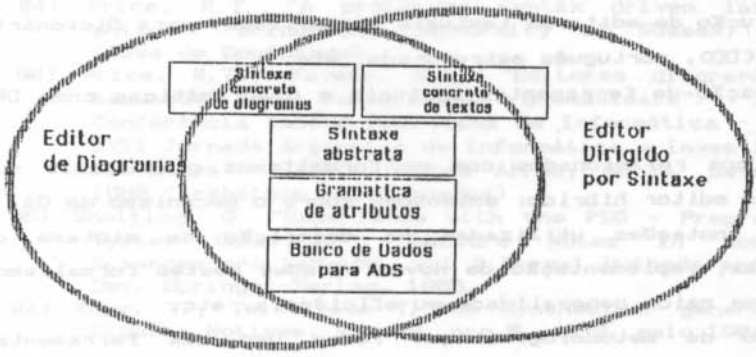

figura 5 - O núcleo do Editor Híbrido

Isto implica na existência de diferentes algoritmos de formatação das estruturas sintática e de procedimentos

especiais para a edição de objetos gráficos, tratados pelo editor híbrido como elementos terminais.

6. Conclusão

Neste trabalho foi apresentado um formalismo para especificação de sintaxe de linguagens para geradores de ambientes de desenvolvimento de software do tipo orientados por estrutura. O formalismo apresenta-se como uma notação alternativa à gramática de grafo. As vantagens da utilização da GLC estendida pelo conceito de nodos compartilhados está na facilidade de especificação de documentos textuais e diagramáticos pelo mesmo formalismo, a integração da representação interna dos documentos em uma única base, bem como a possibilidade de utilização de técnicas largamente utilizadas na contrução de editores textuais estruturados como GA, reconhecedores GLC e algoritmos de geração de menus e aspectos de interface.

Diversos atividades de pesquisa, relacionados com o editor híbrido, estão em andamento no projeto ADS-URGS:

a) construção de editores diagramáticos especificos para DFD, ER (entidade relacionamento), Diagrama estruturado, etc.

b) Construção de editores textuais especificos, para Dicionário de dados (DD), português estruturado, etc.

c) Integração de ferramentas textuais e diagramáticas como DFD e DD, etc.

d) Trabalhos relacionados com os formalismos gramaticais que suportam o editor híbrido: extensões sobre o mecanismo de GA, e sobre as notações utilizadas na descrição da sintaxe de ferramentas; implementação de novas versões destes formalismos visando uma maior generalidade ou eficiência, etc.

e) Estudo de metodologias que fazem uso das ferramentas geradas.

7. Bibliografia

[AHO 86] Aho, A.V.; Sethi, R.; Ullman, J.A.; "Compiler Principles, Techniques, and Tools", Reading

Massachusetts, 1986.

[DAR 87] Dart, S.A.; Ellison R.J.; Feiler P.H.; Habermann A.N. "Software Development Environments". IEEE Computer, Nov. 1987.

[DON 84] Donzeau-Gouge, U.; Kahn, G.; Lang, B.; Melese, B. "Document structure and modularity in Mentor". Sigplan Notices, vol 19, nro 5, pg. 141-8, maio, 1984.

[ENG 87] Engels, G. "Graph Grammar Engineering: A software specification method". Lecture Notes In Computer Science nro 291, 1987.

[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.

[FAV 89] Favero, E.L.; Price, R.T.; "A implementação de um editor diagramático para DFD, baseada em formalismos gramaticais. In: IX Congresso da SBC, 16-21, julho, Uberlândia, MG, 1989.

[HAB 86] Habermann, A.N.; Notkin, D. "Gandalf: Software developments", IEEE Trans. Softw. Eng., Dez. 1986, pp. 1117-1127.

[MEL 89] Melo, W.L.M. "Uma proposta de um editor diagramático generalizado", CPGCC da UFRGS, Março, 1989. (dissertação de mestrado).

[NAG 83] Nagl, M. "A software development environment based on graph technology". Lecture Notes In Computer Science nro 153, 1983.

[NAG 87] Nagl, M. "Software specification by graph grammmars". Lecture Notes In Computer Science nro 291, 1987.

[PRI 84] Price, R.T. "A prototype syntax driven language editor". Bringhton, University of Sussex, 1984. (Tese de Doutorado).

[PRI 88] Price, R.T.; Favero, E.L. "Editores diagramáticos baseados em formalismos gramaticais". In: XIV Conferência Latino Americana de Informática - CLEI; XVII Jornada Argentina de Informática e Investigação Operativa - SADIO. Buenos Aires, 26-30, Setembro, 1988. (trabalhos selecionados)

[SNE 86] Snelting, G. "Experience with the PSG - Programming System Generator", Lecture Notes In Computer Sciences nro 148-162, vol 2: Formal Methods and Sof. Dev. Springer-Verlag, 1985.

[REP 84] Reps, T.; Teitelbaum T. "The Synthesizer generator". Sigplan Notices, vol 19, nro 5, 42-8, maio 1984.