UMA FERRAMENTA INCREMENTAL PARA APOIAR A GERACAO DE SOFTWARE

Eduardo Alberto Dermargos Namur e João José Neto

Departamento de Engenharia Elétrica da Escola Politécnica da USP
São Paulo - SP, Caixa Postal 11.455, CEP 01000, Tel: 815-9322.

RESUMO

Este trabalho descreve um ambiente extensivel de apoio à geração de programas, resultante de uma pesquisa realizada na área de engenharia de "software" [1] visando contribuir para a solução do problema geral de produção automática de programas.

A arquitetura proposta, as caracteristicas de utilização, bem como os ensaios realizados no sistema são a ênfase da apresentação.

Tal ambiente encontra-se implementado e disponivel, atualmente, para microcomputadores tipo IBM-PC, operando sob o sistema operacional DOS.

ABSTRACT

This paper describes a "software generation tool-kit", in the form of an extensible environment, as result of a software engineering research [1] oriented to the automatic software generation problem.

This presentation emphasizes the functional, architectural, and experimental aspects, while supressing operational details.

The proposed environment is currently available for IBM-like personal computers, running under DOS.

1. Introdução

Nas fronteiras atuais do desenvolvimento de "software", não se concebe mais a utilização dos modelos de produção baseados em ambientes tradicionais. De fato, cada vez mais, o aspecto das ferramentas de produtividade vem despertando o interesse de fabricantes e usuários de sistemas computacionais [4] que visam aumentar o desempenho dos recursos empregados e a redução de custos.

Esta preocupação, motivada pelo alto custo do "software" dentro de uma instalação típica, levou a Engenharia de Software a buscar as mais diversas soluções [2], variando desde a aplicação de metodologias na produção de "software", até a criação de ferramentas de geração automática ou semi-automática de programas. Este esforço produziu ferramentas adicionais àquelas normalmente encontradas nos sistemas de programação, visando oferecer aos usuários maiores recursos para o desenvolvimento de "software".

Com isso, surgiram inúmeras pesquisas para a produção de ferramentas CASE de apoio à geração de "software". Tais trabalhos apóiam-se em algumas linhas básicas, como a criação de sistemas especialistas, ambientes de auxílio à geração de aplicativos segundo metodologias estruturadas e núcleos de ambientes gerais de automação. Aliando-se a isso a introdução de novos modelos de programação (como a orientação a objetos) e a busca de padrões (UNIX, OSI, etc.), pode-se considerar que o quadro de produção de "software" encontra-se sujeito a grandes alterações nos próximos anos.

Assim, disputa-se a primazia de se obter um ambiente de desenvolvimento que seja efetivamente capaz de atender aos requisitos dos produtores de "software" quanto à facilidade, rapidez, segurança e confiabilidade.

A proposta a que se refere o presente trabalho é uma ferramenta capaz de automatizar o processo de desenvolvimento de ambientes de desenvolvimento de "software" que, sendo específicos

à aplicação aos quais se destinam, devem cobrir integralmente todos os aspectos ligados à produção de um determinado tipo de programa. Evidentemente que, sendo similares em suas características, os ambientes de desenvolvimento automatizados podem compartilhar um mesmo substrato e até mesmo algumas funções e procedimentos.

A idéia é de criar uma ferramenta particularizável que admita extensão. Através de particularização o usuário pode personalizar a ferramenta, adequando-a à solução dos problemas específicos à geração do tipo de programas cuja geração é desejada. Já a extensibilidade faz com que funções possam ser agregadas de forma incremental, ampliando a abrangência da solução por facilitar a evolução e a própria particularização.

Considerando-se que a especialização do objeto a ser gerado é a base para permitir que métodos sejam aplicáveis na sua geração, o sistema contempla (de forma específica) cada um dos elementos comuns às ferramentas de geração a saber: (1) uma linguagem e seu tratamento léxico e sintático, (2) uma interface com o usuário, (3) interfaces com o sistema operacional, a nível de primitivas e (4) manipulação de estruturas de informação em geral.

2. Detalhamento do Sistema

Para se compreender o enfoque dado ao projeto de uma ferramenta de apoio à geração automática, devem ser considerados os aspectos de arquitetura e de funcionalidade do mesmo.

Além disso, como premissas básicas para a definição do sistema, foram adotadas:

a. Integração: sendo fundamental que a utilização do sistema fosse integrada às demais atividades do desenvolvimento, procurou-se, em particular, fazer com que a ferramenta consistisse em um envólucro ("shell") sobre o sistema operacional, permitindo a ativação de módulos e comandos externos, de maneira transparente ao usuário.

b. Independência: para uma efetiva generalidade da ferramenta, as suposições acerca do ambiente de desenvolvimento foram reduzidas a zero, sendo que o sistema não estabelece linguagens, métodos, e outras ferramentas como pré-requisito para as funções. De fato, tais estas características podem ser definidas pelo usuário.

c. Modularidade: não só a modularidade da ferramenta enquanto programa executável, mas ainda a divisão dos programas-fonte e das bibliotecas do mesmo, foram projetadas assumindo a extensão prevista para o sistema, através de uma estrutura que favorecesse a adaptação, armazenamento, controle e documentação.

d. Reusabilidade: a técnica fundamental para o aumento da produtividade do método foi o reaproveitamento de código. Para isso, além da modularidade, o projeto do sistema envolveu uma divisão funcional das rotinas e o projeto das interfaces e estruturas de dados que, apoiados por uma linguagem de descrição de "software", pudessem favorecer uma programação de elementos de uso genérico.

Adicionalmente, para facilidade de operação o sistema é dirigido por menus. Usando o conceito clássico de semântica operacional, foi implementada uma máquina virtual para incorporar as diversas funções do usuário, e mesmo a criação de uma linguagem de definição de "software" (LDS) própria do usuário. Para favorecer a portabilidade, a codificação foi feita na linguagem "C".

2.1. Arquitetura

Para a obtenção da arquitetura do sistema proposto, foram considerados os aspectos comuns às diversas ferramentas e que são, de fato, relevantes na maioria dos programas.

Com relação aos aspectos de interfaceamento com o usuário e com o sistema operacional, bem como com relação à manipulação de dados, optou-se pela criação de rotinas gerais, através de uma linguagem LDS que incorporasse mecanismos de encapsulamento si-

milares aos da programação orientada a objetos [3] convenientes a esta generalidade. Para facilitar este trabalho, os programas-fonte das rotinas presentes internamente no sistema são partes integrantes da ferramenta.

Já no aspecto de tratamento de linguagem, a proposta implica em um gerador automático de reconhecedores sintáticos para permitir a escolha da linguagem de geração. Para isso, adotou-se o método de geração de reconhecedores sintáticos descrito em [5], sendo que a partir de uma descrição da linguagem em uma metalinguagem similar à notação de Wirth, são gerados os autômatos de reconhecimento, que ao serem interpretados por uma máquina virtual, desencadeiam, nas transições de estado, as ações que implementam os aspectos semânticos relacionados à linguagem.

Com isso, chegou-se aos elementos que deveriam constar da ferramenta, e a forma de estrutura-los modularmente. A figura 1 mostra os módulos do sistema que então são detalhados a seguir:

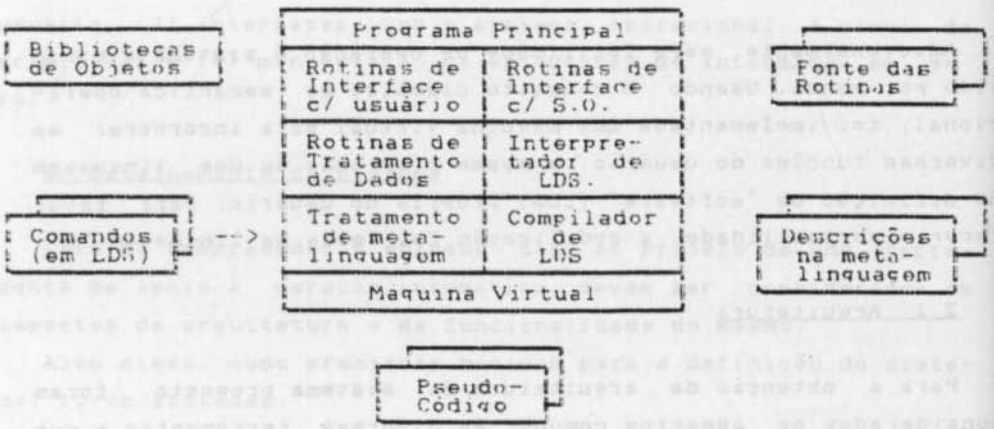

Figura 1 - Componentes do sistema.

* Máquina Virtual: Necessária para implementar os autômatos que interpretam os processos do sistema, dentro do conceito de semântica operacional.

* Rotinas Gerais de Tratamento de Informações: Necessárias para a manipulação de estruturas de dados pelos vários componentes do sistema, e de fundamental importância para a obtenção da sua estrutura modular, que garante a generalidade do mesmo.

* Rotinas Gerais de Interface com o Sistema Operacional: Usadas para isolar as particularidades do sistema operacional visando garantir a portabilidade do sistema, bem como para prover as funções necessárias à operação dos ambientes de geração.

* Rotinas Gerais de Interface com o Usuário: Usadas para prover a funcionalidade geral da interface homem-máquina, dentro do ambiente de desenvolvimento.

* Meta-Linguagem: Necessária para a formalização de linguagens de definição de "software" (LDS) ou de meta-linguagens do usuário, a serem tratadas pelo sistema.

* LDS: Necessária para a definição da semântica dos processos a serem executados pelo sistema, através da interpretação por meio da máquina virtual.

* Comandos: "Programas" na LDS, responsáveis pela implementação dos menus e por ditar a funcionalidade inicial do sistema, servindo de base para especialização por parte do usuário.

* Pseudo-código: Representações codificadas que armazenam os autômatos utilizados para os processos de tratamento da LDS e da metalinguagem

Nota: A extensibilidade e particularização requerem que o sistema seja considerado como sendo constituido pelo conjunto de fontes e objetos, para execução ou ligação.

2.2. Modelo de Execução

O modelo de execução do sistema implementa de forma concreta uma máquina virtual de interpretação de processos, conforme a figura 2 abaixo:

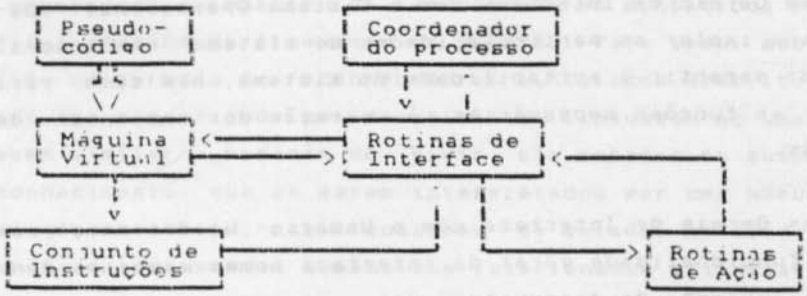

Figura 2 - Estrutura de execução.

Na figura estão representados os elementos da estrutura de interpretação a ser usada para a execução dos processos relacionados à linguagem. São eles:

1) Máquina Virtual: é o componente (rotina) que irá interpretar o pseudo-código, baseando-se em um conjunto pré-estabelecido de instruções. Esta rotina é de uso geral e deve consistir no núcleo de qualquer implementação de autômatos de pilha.

2) Conjunto de Instruções: é um grupo de funções (cada uma das quais associada a um código de instrução) projetadas especificamente para atuar sobre um dado tipo de interpretação. Correspondem aos aspectos de controle de execução do autômato.

3) Pseudo-Código: é um programa expresso na linguagem de montagem da máquina virtual, onde as instruções correspondem univocamente as funções primitivas definidas no conjunto de instruções.

4) Rotinas de Ação: é um grupo de funções (opcionais) que implementam os aspectos semânticos da interpretação. São acionadas

pelas rotinas que implementam o conjunto de instruções da máquina virtual, com base nas instruções do pseudo-código.

5) Rotinas de Interface: é um grupo de funções que implementam as interfaces de uma determinada interpretação com os demais componentes do sistema. Simulam os dispositivos de E/S da máquina.

6) Coordenador: é a rotina (ou parte da rotina) responsável por coordenar a execução do autômato. Este elemento provê a associação entre os vários componentes citados, e, através das rotinas de interface, inicializar, ativar, terminar uma interpretação, etc.

2.3. Funções do Sistema

Como funções básicas, acessíveis por meio do menu original do ambiente de geração, o usuário dispõe de:

1) Compilação de LDS: consiste na geração de código "C" correspondente a uma descrição LDS.

2) Tratamento de meta-linguagem: consiste em exprimir os autômatos na "linguagem de montagem" da máquina virtual.

3) Geração de autômatos: corresponde à montagem dos programas, resultantes da manipulação de meta-linguagem, apresentados anteriormente. Esta função não foi incorporada à função anterior para permitir uma eventual alteração manual do código gerado.

4) Re-geração do sistema: é um processo bastante simples e que é executado inteiramente pelo ambiente "C". Consiste da ligação dos vários componentes executáveis do sistema (isto é, suas rotinas), armazenados em bibliotecas.

5) Término da execução: responsável por finalizar a execução do sistema, com o fechamento de arquivos e liberação das áreas utilizadas.

2.4. Utilização do Sistema

A utilização do sistema como ferramenta de apoio à criação de geradores de "software" se dá por extensão e particularização. Tais mecanismos estão explicados a seguir:

Extensão do sistema

A extensão do sistema pode ser no aspecto da linguagem de definição de software (LDS) ou no aspecto operacional.

A extensão da LDS envolve a alteração da descrição da mesma por meio de uma meta-linguagem e a criação das ações semânticas associadas. Tais ações devem ser relacionadas internamente no sistema. Para facilitar esta atividade, o usuário poderá escrever tais rotinas em LDS e a seguir "compilá-las", gerando código-objeto a ser incorporado de forma permanente ao sistema por meio de uma re-geração.

A extensão dos comandos do sistema permite ao usuário introduzir novos procedimentos operacionais. Estes comandos, codificados em LDS são conhecidos por seu nome. Através deste expediente, o usuário pode definir procedimentos de edição, compilação, e outras atividades pertinentes ao desenvolvimento, e ativá-las diretamente dentro do sistema. Note-se que os batch-files do DOS podem ser ativados diretamente por instruções da LDS.

Particularização do sistema

A particularização do sistema permite transformar o mesmo em um gerador de qualquer tipo de "software". Para isso, devem ser seguidos os seguintes passos:

- criação da LDS utilizada no processo desejado de geração e descrição da mesma em termos da meta-linguagem do sistema;
- criação dos autômatos associados, e a codificação dos vários componentes da interpretação (com exceção da máquina virtual);
- introdução da função em menus ou diretamente na linha de comando por extensão de comandos do sistema, para ativação do processo de geração implementado.

3. Experimentos realizados

O sistema implementado foi testado para verificar sua capacidade de apoiar a geração automática de "software". Dois ensaios, em particular, denotam a facilidade de se estender o sistema, bem como torná-lo específico para uma dada finalidade de geração.

3.1 Ensaio de particularização

Para comprovar as características de adaptação de utilização da ferramenta, optou-se por um ensaio de geração de compiladores. Para isso, o ambiente foi particularizado, sendo introduzidas as rotinas usualmente encontradas nos compiladores (de fato, tais rotinas já existiam no ambiente original, bastando a generalização para que as mesmas pudessem ser utilizadas de modo geral). Entre essas rotinas tem-se:

- rotina de manipulação de tabela de símbolos;
- rotina de emissão de referências cruzadas;
- rotina de tratamento de erros de compilação (análises léxica e sintática;
- rotinas de geração de código.

Uma vez implementadas as rotinas, utilizando-se uma definição, em metalinguagem, de uma linguagem BASIC, pôde-se obter um compilador onde as rotinas semânticas consideradas possuiam, em média, três instruções da LDS. Tal codificação não levou mais do que dois dias para concluir completamente a "geração" de um compilador.

3.2 Ensaio de extensão

O ensaio de extensão consistiu na introdução, no sistema, de uma característica de geração de telas. Para isso, adotaram-se as metáforas de visualização [2] de tela formatada, menus e janelas. Para isso, adotou-se o seguinte procedimento:

- criação de uma linguagem de definição de telas e sua definição em termos da meta-linguagem;

- codificação das rotinas de ação do gerador de tela, baseadas nas (já existentes) rotinas primitivas de interface com o usuário;

- introdução de um comando na LDS para a ativação de uma interpretação de uma descrição de tela a ser gerada.

Com isso, o sistema ganhou um comando de geração de telas de modo geral. O trabalho dispendido nesta tarefa não foi superior a uma semana.

4. Conclusões

A utilização do sistema como ferramenta de apoio à geração de "software" de forma automática foi comprovada, e deve-se basicamente às seguintes características:

- facilidade de extensão, proporcionada pela geração automática de autômatos que representam os processos de geração;

- facilidade de adaptação, pela existência de uma linguagem poderosa e de alto nível para codificação da semântica dos processos envolvidos na geração automática, nos seus aspectos fundamentais;

- facilidade de generalização, proporcionada pela arquitetura do sistema e pelo encapsulamento de dados oferecido pela LDS original do mesmo.

Com isso, tem-se implementada uma ferramenta capaz de produzir geradores de "software" de um modo geral, além de se ter um instrumento poderoso de prototipação e estudo de ambientes de desenvolvimento experimentais, considerando todos os aspectos que se têm implementados.

A utilização efetiva envolverá um estudo detalhado, por parte do engenheiro de software usuário do sistema. Após isso, o mesmo estará apto a particularizar tal ferramenta para criar uma família de ambientes de geração de software que, por serem baseados

no mesmo substrato, podem compartilhar a maior parte do seu código.

O sistema apresentado pode ser enquadrado dentro de várias categorias de ambientes. Como um "assistente" na produção de "software" [6], um núcleo de ambiente genérico [7] e uma base para a produção de ambientes [8]. Todas estas características, especializadas ao caso de geradores de "software" dão ao sistema a esperada generalidade de aplicação para solucionar o problema geral de desenvolvimento automatizado de "software".

5. Referências

1. Namur, E. A. D. - Um sistema incremental de apoio à geração de software. Dissertação de mestrado apresentada à EPUSP, 1990.
2. Fisher, A. S. - CASE: Using Software Development Tools. John Wiley & Sons, Inc., New York, 1988.
3. Cox, Brad J. - Object-Oriented Programming - An Evolutionary Approach . Addison-Wesley Publishing Company, 1987.
4. Damasio, E. - Pesquisa. INFO, Rio de Janeiro, Ago., 1989, p.25-34.
5. Neto, J. J. & Magalhães, M. E.S. - "Reconhecedores Sintáticos: Uma Alternativa Didática para Uso em Cursos de Engenharia". In: XIV CONGRESSO NACIONAL DE INFORMATICA. São Paulo, 1981, p. 171-81.
6. Karimi, J. & Konsynski, B. R. - An Automated Software Design Assistant. IEEE Trans. on Soft. Eng., 14(2):194-210, Feb., 1988.
7. Lamsweerde, A. van et alli - The Kernel of a Generic Software Development Environment. SIGPLAN Notices, New York, 22(1): 208-217, Jan., 1987.
8. Reppy, J. H. & Gansner, E. R. - A Foundation for Programming Environments. SIGPLAN Notices, New York, 22(1):218-27, Jan., 1987.