Relatório Técnico

Núcleo de Computação Eletrônica

O Gerador Automático de Programas - FAST CASE -

Márcio A. O. Pinto
Renato N. Bastos
Eber A. Schmitz
Denis S. Silveira

NCE - 20/00

Universidade Federal do Rio de Janeiro

O Gerador Automático de Programas

FAST CASE

Márcio A. O. Pinto[1] pinto@posgrad.nce.ufrj.br
Renato N. Bastos[1] bastos@lci.ufrj.br
Eber A. Schmitz[1] eber@nce.ufrj.br
Denis S. Silveira[2] denis@cos.ufrj.br

Núcleo de Computação Eletrônica
Instituto de Matemática[1]
Universidade Federal do Rio de Janeiro
Caixa Postal 2324, CEP:20001-970
Rio de Janeiro, RJ – Brasil

Programa de Engenharia de Sistemas e Computação
COPPE/Sistemas[2]
Universidade Federal do Rio de Janeiro
Caixa Postal 68.511, CEP:21.945-970
Rio de Janeiro, RJ – Brasil

RESUMO

Este artigo apresenta um gerador automático de programas para o ambiente *Delphi*. Este módulo, que é uma extensão do FAST CASE, obtém as especificações do programa a ser gerado a partir dos dados contidos nos modelos de requisitos e de análise e projeto. A estrutura do programa gerado segue a arquitetura padrão da Metodologia Rápida. O código fonte, ao ser compilado no ambiente *Delphi* gera um programa executável.

ABSTRACT

This paper presents an automatic code generator for the *Delphi* Development Environment. This module which is an extension of FAST CASE, obtains the program specifications from the requirements and design models stored in the project database. The structure of the generated code follows a standard architecture and can be compiled to an executable program.

Palavras-chave: CASE, UML, Engenharia de Software, Orientação a Objetos, Arquitetura de Software, Engenharia de Requisitos;

1. Apresentação

Este trabalho mostra uma ferramenta para geração automática de código fonte acoplada a ferramenta FAST CASE, que é uma ferramenta CASE para suporte ao desenvolvimento de sistemas orientado a objetos, voltada para o ambiente de ensino [4] [5]. Esta ferramenta recebe como entrada modelos (*estático e dinâmico*) de um sistema e produz como saída o esqueleto do código fonte para este sistema no ambiente *Delphi*. Os dados de entrada para a geração de código são:

- a descrição formal dos casos de uso. Um caso de uso descreve uma funcionalidade de um sistema de informação e deve ser escrito em linguagem natural. Jacobson [2] apresentou uma proposta para especificação de requisitos onde os casos de uso são descritos como máquinas de estado. Segundo ele, cada passo do caso de uso é mapeado em um estado na máquina de estado, e cada um dos eventos gerados pelo usuário são mapeados nas transições do estado. Desta forma, conhecendo-se a especificação formal de um caso de uso, podemos gerar automaticamente o objeto de controle encarregado da seqüência do caso de uso;
- a especificação das classes do domínio do problema. Cada uma das classes do domínio do problema deve ser especificada com seus atributos e métodos; e

- a especificação das classes de interface. Cada uma das classes de interface, deve ser especificada com suas propriedades, que representam os campos a serem lidos ou escritos, juntamente com os eventos gerados pelo usuário do sistema.

2. Arquitetura Padrão do Código Gerado

A arquitetura de software descreve os módulos componentes de um sistema e suas respectivas formas de interação [1]. A arquitetura utilizada como padrão para o código gerado utiliza as definições apresentadas na Metodologia Rápida [3]. Em suma, ela deve definir três aspectos importantes de um sistema:

1. os mecanismos utilizados para o armazenamento de dados;

   objetos entidades (domínio): são os objetos que realmente compõem o escopo do problema. Normalmente, essas informações deverão ser persistidas para posterior recuperação;

2. os mecanismos utilizados para a interface com o usuário;

   objetos de interface: são encarregados da interação entre usuários e sistema. Esses objetos devem transformar os eventos externos em sinais internos e os eventos internos em sinais para o mundo externo;

3. os mecanismos utilizados para o controle do sistema;

   objetos de controle: são encarregados do sequenciamento do caso e, formalmente, implementam a máquina de estado do caso de uso;

2.1 Objetos de Domínio

A implementação de cada objeto de domínio do problema é estruturada com o uso de três classes:

- ***TGerenteObjeto***: uma instância desta classe atua como gerente do conjunto, atendendo a serviços relativos ao conjunto de elementos;
- ***TTabObjeto***: esta classe representa a estrutura de dados que faz o acesso aos dados em memória permanente. Note a relação de composição da classe *TGerenteObjeto* com a classe *TTabObjeto*. No código gerado, a classe *TTabObjeto* é implementada por um componente do tipo *TTable*; e
- ***TUmObjeto***: cada objeto desta classe representa um elemento do conjunto. Este objeto é importante quando desejamos manipular um único elemento do conjunto, por exemplo, para sua criação ou alteração.

2.2 Objetos de Controle

Na arquitetura padrão, fazemos com que cada caso de uso seja mapeado em uma classe de controle correspondente. Esta classe contém uma máquina de transição de estados que controla o comportamento dinâmico do objeto de controle, em função de seu estado corrente e dos eventos que provocam as mudanças de estado. Os objetos de controle possuem apenas um serviço, que é o de tratar as sinalizações provindas da interface. Este serviço, que chamamos de *evTrataEvento*, tem sua especificação derivada do caso de uso que ele controla.

Se o caso de uso possuir uma especificação formal, como por exemplo, um diagrama de transição de estados, o código fonte é gerado automaticamente.

3. Gerador de Código

A operação da geração de código foi dividida em duas fases. São elas: preparação do modelo e a geração propriamente dita.

3.1 Preparação do Modelo

Esta é a fase responsável pela criação dos modelos de onde o gerador irá retirar todas as informações necessárias. Nesta fase devemos criar:

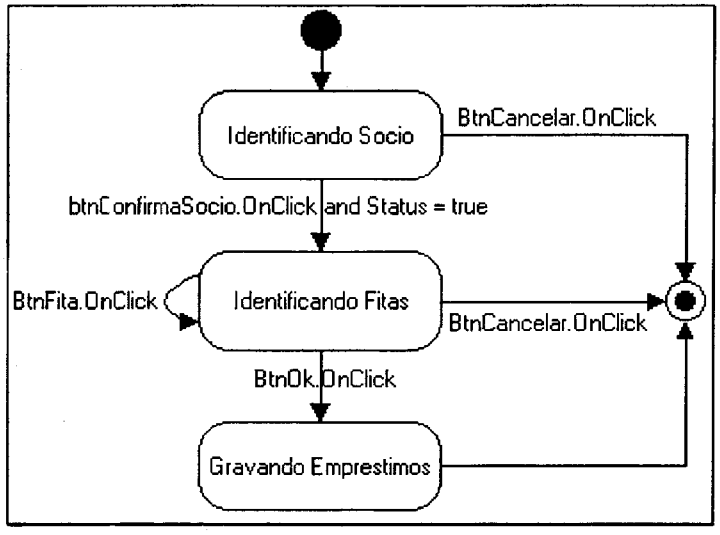

Figura 1- Diagrama de estado de um caso de uso

- um modelo de requisitos contendo os casos de uso, com suas respectivas definições, usando o FAST CASE;
- um modelo de classes do domínio do problema. Para cada classe, é necessário definir seus atributos e os serviços;
- um modelo das classes de interface. Para cada classe é necessário definir as propriedades que representam os componentes nela apresentados; e
- um diagrama de transição de estados associado à classe que irá controlar os casos de uso. Este diagrama deve mostrar os passos do caso de uso, os eventos requeridos para cada uma das transições. Em cada estado são indicadas as rotinas executadas na entrada, durante e na saída de cada estado (*Entry, Do, Exit*). Cada transição é anotada com o nome do evento que causa esta transição.

3.2 Geração do Programa

O programa *Delphi* gerado consiste de um conjunto de unidades que são mapeadas seguindo a arquitetura padrão. As unidades geradas são:

- unidade de controle: gera uma unidade de compilação para cada caso de uso. Cada unidade contém uma máquina de estado que segue o mesmo comportamento especificado pela máquina de estado do caso de uso;
- unidade de interface: gera uma unidade com o formulário (padrão *Delphi*) para cada caso de uso. Cada unidade gerada, já provê o mecanismo de *callback* para sinalizar ao objeto de controle, e os eventos informados pelo usuário.

- unidade de domínio do problema: gera uma unidade contendo todas as classes de domínio, mapeadas segundo a arquitetura padrão mostrada acima. O gerador também cria uma tabela para cada classe de domínio.

As classes de domínio e interface já são geradas com alguns serviços padrões. Por exemplo, para as classes de domínio, os seguintes serviços são gerados:

function *ExisteElemento: TObject;*

function *ListaElementos: TList;*

Os casos de uso que são do tipo manutenção já geram um código padronizado com os serviços de inclusão, alteração e exclusão de registros da tabela de armazenamento. A figura 2 mostra um esquema de como funciona a geração.

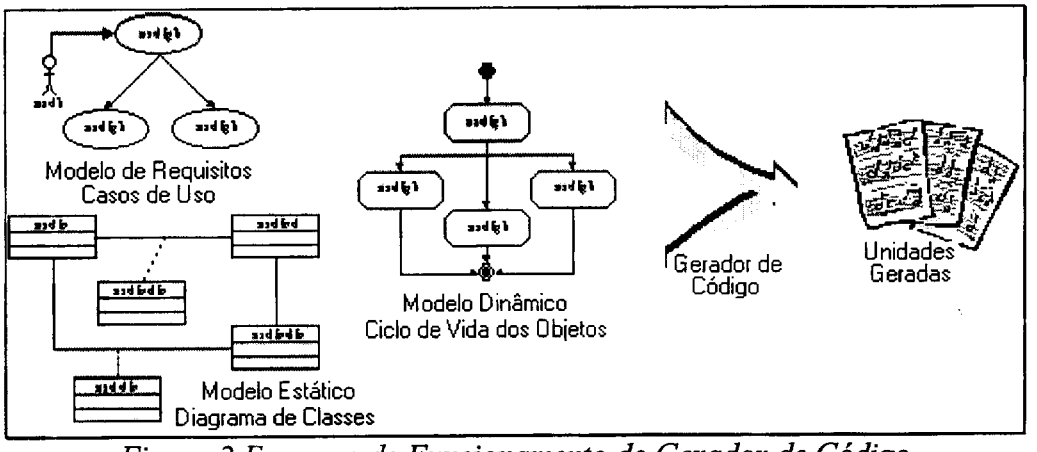

Figura 2 Esquema de Funcionamento do Gerador de Código

4. Conclusão

Este artigo mostrou um sistema para geração de código para o ambiente *Delphi* a partir de modelo de projeto armazenado no FAST CASE. Este gerador já vem sendo utilizado em turmas da graduação, onde vêm apresentando como resultados: o melhor entendimento dos alunos para a arquitetura de programas e uma redução no tempo gasto em atividades mecânicas de programação, desta forma proporcionando mais tempo livre para a construção dos modelos dos sistemas.

5. Referências Bibliográficas

[1] Garlan et al., Architectural Styles, Design Patterns, and Objects, IEEE Software, Jan/Fev 1997, pp.43-52;

[2] Jacobson et al., Object-Oriented Software Engineering – A Use Case Driven Approach, Addison Wesley, 1992;

[3] Schmitz, E., Silveira, D., Desenvolvimento de Software Orientado a Objetos, Editora BRASPORT, ISBN 85-7452-039-X, 2000;

[4] Schmitz, E., Silveira, D., FAST CASE: Uma Ferramenta para o Desenvolvimento Visual de Sistemas Orientados a Objetos, XIII SBES, Caderno de Ferramentas, pp. 29-32, 1999;

[5] Silveira, D., FAST CASE: Uma Ferramenta para o Desenvolvimento Visual de Sistemas Orientados a Objetos, Tese de Mestrado, IM/NCE/UFRJ, 1999;