FAD: UMA FERRAMENTA CASE PARA ENGENHARIA REVERSA[1]

Maikon Adiles Fernandez BUENO[2], Renato Bobsin MACHADO[3],
Huei Diana LEE[4], Feng Chung WU[5]

Escrito para apresentação na II JORNADA CIENTÍFICA DA UNIOESTE
11 a 13 de junho de 2003 – Unioeste – PRPPG – Campus de Toledo - PR

RESUMO: A utilização não sistematizada de metodologias de desenvolvimento na construção de softwares, origina problemas durante a atividade de manutenção. Esse fato tem motivado o desenvolvimento de ferramentas de Engenharia de Software Apoiadas por Computador (CASE). Nesse contexto, desenvolveu-se a Ferramenta de Auxílio à Documentação de Sistemas – FAD, para auxiliar o processo de Engenharia Reversa, que tem por finalidade relacionar itens básicos de um software e apresentar relatórios contendo informações que ajudam na compreensão do fluxo de dados de um sistema, além de exportar classes para uma ferramenta clássica de modelagem Orientada a Objetos.

PALAVRAS CHAVES: Metodologia de Desenvolvimento, Documentação de Sistemas, Modelagem de Sistemas

FAD: A REVERSE ENGINEERING CASE TOOL

ABSTRACT: The non systematic use of system developing methodologies generates problems during the maintaining phase which has motivated the development of Computer Aided Software Engineering tools (CASE). In this context, it was developed a system documentation aid tool which helps during the Reverse Engineering process. This process has the objective to relate system basic items and to present reports containing information that helps on the system data flow comprehension besides exporting classes to a classic Object Oriented modeling tool.

KEYWORDS: Developing Methodologies, System Documentation, System Modeling

INTRODUÇÃO: O advento das linguagens visuais agravou o problema do desenvolvimento de inúmeros softwares com falhas nos mecanismos de modelagem e documentação. Muitos desenvolvedores constróem softwares sem documentação, o que ocasiona uma perda considerável de tempo durante a fase de manutenção, ou mesmo, durante o desenvolvimento, quando se trata de um software de maior porte. Nesses sistemas, a única fonte de documentação existente é o próprio código. A Engenharia Reversa, inserida nesse contexto, constitui um amplo conjunto de tarefas relacionadas ao entendimento e modificação de sistemas de software. Suas duas principais atividades são a identificação e o relacionamento dos componentes de um sistema e a criação de descrições de alto nível de

[1] Projeto de pesquisa desenvolvido no Laboratório de Bioinformática – LABI , Centro de Engenharias e Ciências Exatas, UNIOESTE, Foz do Iguaçu - PR. Caixa Postal 961 CEP 85870-900, Foz do Iguaçu, PR Tel.: 45 5752727, ramal 1114, Fax: 45 5752733.
[2] Aluno do Curso de Ciência da Computação, estagiário do LABI, Centro de Engenharias e Ciências Exatas, UNIOESTE, Foz do Iguaçu – PR; labi@unioeste.br.
[3] Professor do Centro de Engenharias e Ciências Exatas, UNIOESTE, Foz do Iguaçu – PR; Coordenador da Área de Computação do LABI.
[4] Professora do Centro de Engenharias e Ciências Exatas, UNIOESTE, Foz do Iguaçu – PR; Coordenadora Geral do LABI.
[5] Pesquisador da Faculdade de Ciências Médicas da UNICAMP, Campinas-SP; Coordenador da Área Médica do LABI.

vários aspectos do sistema (CHIKOFSKY, 1993). A Ferramenta de Auxílio à Documentação de Sistemas - FAD, tem como objetivo reduzir esforços durante a aplicação do processo de Engenharia Reversa em um sistema. O FAD, assim como outras ferramentas, não permite a recuperação total da documentação a partir de códigos fontes. Entretanto, esse aplicativo apresenta características importantes e raramente encontradas em ferramentas semelhantes. Uma dessas características é a interpretação de classes escritas na linguagem Borland Delphi (BORLAND DELPHI, 2003) para geração de arquivos que podem ser importados pela ferramenta Rational Rose 2000 (RATIONAL, 2003). Para a realização de testes do FAD, utilizou-se o Androsys, Sistema de Gerenciamento de Dados para Reprodução Humana, desenvolvido em Borland Delphi 6.0 e Interbase 6.0, como sistema gerenciador de banco de dados (TROIANO, 2003).

MATERIAL E MÉTODOS: O FAD, dividido em dois módulos, tem como objetivo principal a geração de relatórios que vinculem campos de formulários com atributos de tabelas e classes. O primeiro compreende atividades manuais, como cadastro de classes, tabelas, formulários e seus inter-relacionamentos. O segundo consiste em tarefas automatizadas, limitadas a sistemas desenvolvidos em linguagem Delphi 6.0 e Interbase 6.0 como Sistema Gerenciador de Banco de Dados – SGBD. A funcionalidade desse segundo módulo implica na interpretação de arquivos fontes e exportação de classes para o formato UML (*Unified Modeling Language*) do software Rational Rose 2000, assim como o registro de formulários, classes e tabelas em uma base de dados. Para o desenvolvimento do aplicativo FAD foram utilizadas a ferramenta Borland Delphi 6.0 e o SGDB Interbase 6.0. A Figura 1 apresenta a arquitetura do FAD.

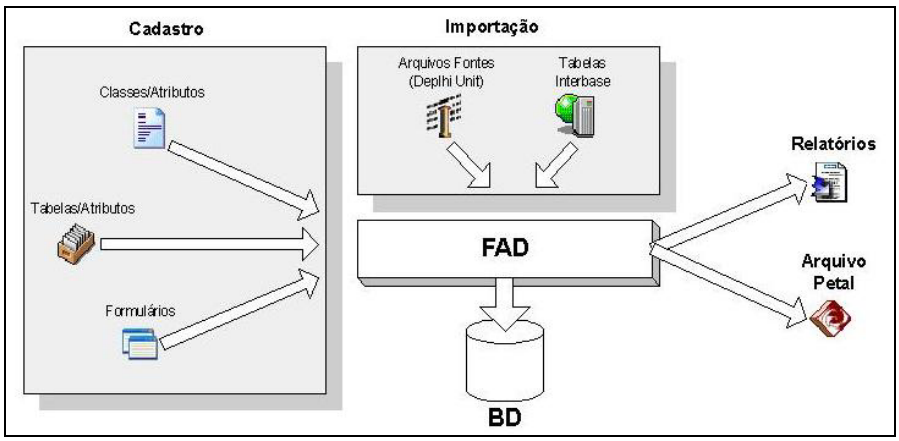


FIGURA 1 Diagrama de arquitetura do FAD

O processo de interpretação do arquivo é realizado por meio de regras semânticas. Para isso é efetuada uma leitura seqüencial do código fonte, no qual as declarações são limitadas por ponto e vírgula. O processo aplica a classificação de cada item como atributo (variáveis ou componentes no caso de um formulário) ou método (funções, procedimentos ou eventos de algum componente ou do próprio formulário). Para o êxito desse processo são efetuadas buscas por palavras chaves, nas quais padrões como *function*, *procedure*, *destructor* e *constructor* definem métodos; os demais itens são definidos como atributos. Em complemento a essas regras, os formulários possuem tratamento especial, sendo um atributo identificado pela presença de determinados tipos ou componentes existentes na linguagem Delphi, entre as quais *TEdit*, *TMemo*, *TComboBox, TDbGrid, Integer, String*, e outros que, de alguma forma, podem ser relacionados a um campo de uma tabela. Nesse caso, desconsidera-se componentes com caráter apenas visual ou de serviço, tais como: *TLabel*, *TMenu*, *TOpenDialog* e *TDataSource*. Uma exceção às regras relacionadas a formulários é permitida optando-se por importá-lo como classe, o que acarreta na interpretação de todos os atributos. Essas regras foram definidas por meio da sintaxe BNF

(*Backus Naur Form*) disponibilizada na linguagem Delphi e adaptada às funcionalidades implementadas na ferramenta FAD, a qual é apresentada na Figura 2. Após o processo de classificação de todas as declarações e o armazenamento dos atributos de uma classe ou formulário, o aplicativo disponibiliza a opção de geração de um arquivo Petal, o qual pode ser importado pela ferramenta Rational Rose 2000. Esse arquivo contém os dados de uma classe ou formulário como atributos e métodos. Cada um deles pode ter propriedades específicas como, por exemplo, tipo para atributos, argumentos e retorno de métodos. O processo de cadastro automatizado de tabelas de um banco de dados cujo SGDB é Interbase, realiza-se por meio da execução de sentenças SQL, no qual primeiramente são listadas e disponibilizadas ao usuário todas as tabelas existentes. Após a escolha da tabela que deseja-se cadastrar, são obtidos os nomes dos campos da tabela os quais são registrados na base de dados do FAD.

```
ClassType -> Ident = CLASS
               [ClassFieldList]
               [ClassMethodList]
             END

ClassVisibility -> [PUBLIC | PROTECTED | PRIVATE | PUBLISHED]
ClassMethodList -> (ClassVisibility MethodList)/';'...
ClassFieldList -> (ClassVisibility ObjFieldList)/';'...

MethodList -> (MethodHeading [';' VIRTUAL])/';'...

MethodHeading -> ProcedureHeading
              -> FunctionHeading
              -> ConstructorHeading
              -> DestructorHeading

ProcedureHeading -> PROCEDURE Ident [FormalParameters]
FunctionHeading -> FUNCTION Ident [FormalParameters] ':' (Type)
ConstructorHeading -> CONSTRUCTOR Ident [FormalParameters]
DestructorHeading -> DESTRUCTOR Ident [FormalParameters]

FormalParameters -> '(' FormalParm/';'... ')'
FormalParm -> [VAR | CONST | OUT] Parameter
Parameter -> IdentList
          -> Ident ':' Type '=' ConstExpr

ConstExpr -> <constant-expression>
Ident -> <identifier>
IdentList -> Ident
ObjFieldList -> (IdentList ':' Type)/';'...
Type -> (Components | DataType)
Components -> (TEdit | TMemo | TComboBox...]
DataType -> <StandardDelphiTypes>
```

FIGURA 2 Gramática simplificada de declaração de classes aceita pelo FAD

RESULTADOS E DISCUSSÃO: Após o cadastro das classes, tabelas, formulários e seus respectivos atributos, o usuário pode relacionar campos de tabelas com atributos de classes. Como um atributo de classe pode ser instância de uma outra estrutura, esse poderá ser relacionado com um ou mais atributos de tabelas. Por exemplo, considere as declarações de classes da Figura 3 e suponha que na base de dados não haja nenhuma tabela para armazenar especificamente os endereços das pessoas. Desse modo, o endereço de cada pessoa está armazenada na tabela *TPessoa*, juntamente com os dados da pessoa. Internamente, o endereço de uma pessoa (instância de *CPessoa*) será carregado para o atributo *Endereço*. Sendo assim, o atributo *Endereço* pode ter relacionado a si vários atributos da tabela como, nesse caso, *Rua* e *Bairro*.

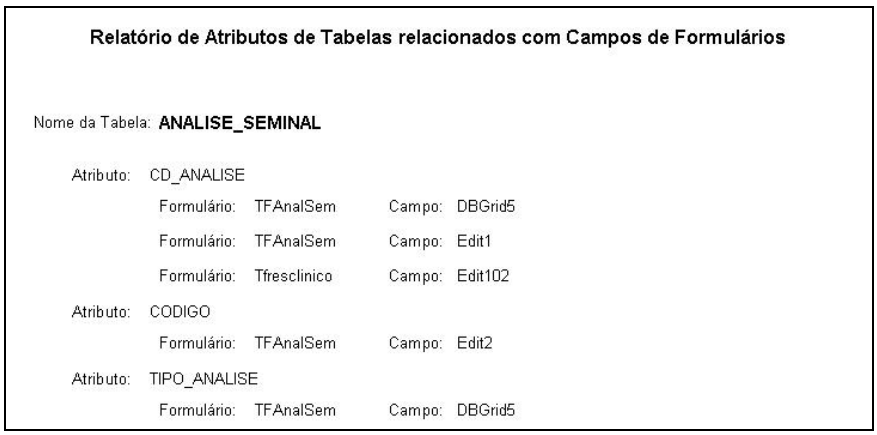
```
CPessoa = class
private
  Nome: String;
  Endereço: CEndereco;
public
  function getNome(): String;
  function getEndereco(): CEndereco;
  procedure setNome(_Nome: String);
  procedure setEndereco(_Endereco: CEndereco);
end;
```

```
CEndereco = class
private
  Rua: String;
  Bairro: String;
public
  function getRua(): String;
  function getBairro(): String;
  procedure setRua(_Rua: String);
  procedure setBairro(_Bairro: String);
end;
```

FIGURA 3 Exemplo de atributo de classe ligado a mais de um atributo de tabela

Do mesmo modo, um componente de um formulário poderá ser relacionado com um ou mais atributos de classes e um atributo de classe poderá ter associado a si um ou mais componentes de formulários. Por existir componentes visuais que se comunicam diretamente com campos de tabelas, o aplicativo também disponibiliza a opção de relacionamento entre campos de tabelas com componentes de formulários. Semelhante ao relacionamento entre atributos de classes e campos de tabelas, aqui um componente visual pode ser relacionado com um ou mais campos de tabelas. Com os relacionamentos realizados, o aplicativo FAD é capaz de gerar três tipos de relatórios que auxiliam na compreensão do fluxo de dados do sistema, sendo de grande valia para o projeto de interface. São eles:

- *relatório de atributos de tabelas relacionados com campos de formulários* (Figura 4): exibe uma relação entre cada atributo da tabela selecionada, e os campos de formulários que estão ligados a esse atributo;



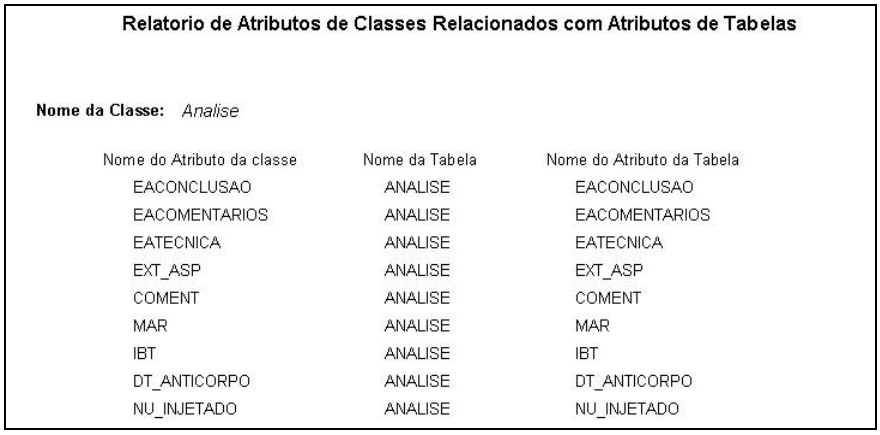
Relatório de Atributos de Tabelas relacionados com Campos de Formulários

Nome da Tabela: **ANALISE_SEMINAL**

Atributo:	CD_ANALISE		
Formulário:	TFAnalSem	Campo:	DBGrid5
Formulário:	TFAnalSem	Campo:	Edit1
Formulário:	Tfresclinico	Campo:	Edit102
Atributo:	CODIGO		
Formulário:	TFAnalSem	Campo:	Edit2
Atributo:	TIPO_ANALISE		
Formulário:	TFAnalSem	Campo:	DBGrid5

FIGURA 4 Relatório de atributos de tabelas relacionados com campos de formulários

- *relatório de atributos de classes relacionados com atributos de tabelas* (Figura 5): exibe um relatório contendo os atributos de tabelas relacionados com os atributos da classe;



Relatorio de Atributos de Classes Relacionados com Atributos de Tabelas

Nome da Classe: *Analise*

Nome do Atributo da classe	Nome da Tabela	Nome do Atributo da Tabela
EACONCLUSAO	ANALISE	EACONCLUSAO
EACOMENTARIOS	ANALISE	EACOMENTARIOS
EATECNICA	ANALISE	EATECNICA
EXT_ASP	ANALISE	EXT_ASP
COMENT	ANALISE	COMENT
MAR	ANALISE	MAR
IBT	ANALISE	IBT
DT_ANTICORPO	ANALISE	DT_ANTICORPO
NU_INJETADO	ANALISE	NU_INJETADO

FIGURA 5 Relatório de atributos de classes relacionados com atributos de tabelas

- *relatório de campos de formulários relacionados com atributos de classes e tabelas* (Figura 6): exibe uma relação contendo cada campo de determinado formulário, e todos os atributos de classes que estão a ele relacionados.

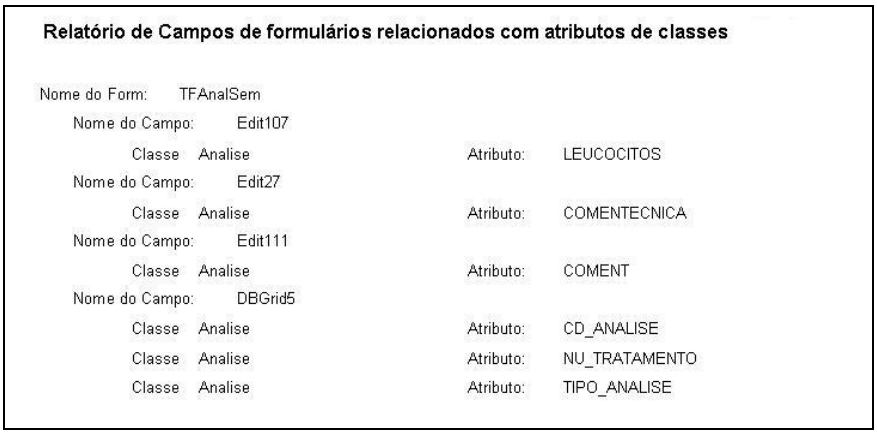


FIGURA 6 Relatório de campos de formulários relacionados com atributos de classes

O entendimento do funcionamento de um sistema a partir do código, durante a manutenção, não é um processo trivial (VERONESE, 2002). A falta de documentação durante o processo de desenvolvimento e decisões de projetos, dificultam a manutenção de sistemas elevando o tempo despendido para a sua realização. Esse fato praticamente não ocorreria se existisse documentação adequada do sistema. Dentro do paradigma de desenvolvimento Orientado a Objetos - OO, existem diversos tipos de documentações que auxiliam o desenvolvedor, tais quais: diagrama de classes, diagrama de colaboração, diagrama de pacotes, entre outras. Dessas citadas, os diagramas de classes tem importância fundamental na condução do processo de construção de um software OO. O FAD auxilia na construção de um diagrama de classes a partir do código fonte. Por meio dele é possível recuperar cada classe de um determinado sistema desenvolvido em Delphi 6.0, ficando a cargo do usuário apenas o relacionamento entre essas classes. Com essa funcionalidade, o usuário ganha tempo, e não precisa analisar cada método e atributo de determinada classe para desenhá-la manualmente em uma ferramenta tal como Rational Rose 2000. Essa última, foi escolhida por ser uma das ferramentas mais utilizadas para o paradigma OO, a qual contém uma série de métodos visuais que facilitam a produção de documentações de aplicações desse paradigma. A geração do arquivo Petal, contendo a classe a ser importada, independe da ordem de disponibilidade dos atributos e métodos no arquivo, desde que, organizados de acordo com a gramática de declaração de classes no Delphi (Figura 2). Para cada declaração lida de uma classe, são levadas em conta propriedades como tipos de retorno, tipo do atributo, argumentos de um método e modo de acesso (*public*, *private*, *protected*). Um arquivo Petal tem seu formato em ASCII e possui uma estrutura sintática semelhante a linguagem Lisp (GRAHAM, 1995), formando uma árvore de objetos. Cada objeto tem uma identificação e opcionalmente uma lista de outros objetos agregados, como exemplos, atributos no caso de uma classe ou argumentos no caso de um método (DAHM, 2001). Cada objeto tem um código único que o identifica, sendo utilizado para referenciar objetos em arquivos Petal. A principal finalidade da geração de arquivos Petal é facilitar o desenho das classes, utilizadas em um sistema desenvolvido em Delphi 6.0, na ferramenta Rational Rose 2000, a qual possui mecanismos para importar esses arquivos. A ferramenta Rational Rose 2000 também permite que classes sejam importadas por meio de Engenharia Reversa, porém o Delphi não está entre as linguagens aceitas por essa ferramenta. Existem outras ferramentas com o mesmo intuito, como por exemplo: JVision (OBJECT INSIGHT, 2003), Structure Builder (WEBGAIN, 2003) e Together (BORLAND TOGETHERSOFT, 2003). Nenhuma dessas ferramentas citadas recuperam classes a partir de códigos fontes em Delphi, no entanto são capazes de adquirir diagramas de classes inteiros incluindo características como associações, dependências, detecção e exibição da estrutura de pacotes, para as linguagens Java, C++ e Visual Basic. Todas as funcionalidades implementadas no FAD foram aplicadas ao sistema Androsys e, a partir da interpretação de seu código fonte e banco de dados, os resultados das documentações, como projeto de interfaces e auxílio na recuperação do diagrama de classes do sistema

foram alcançados. Esses resultados, associados às carências da Engenharia Reversa, motivam a implementação de outras funcionalidades tais como: extração de diagramas de classes com seus relacionamentos, aceitação de maior número de linguagens orientadas a objetos e a associação de figuras de telas aos campos de formulários.

CONCLUSÕES: As ferramentas com o propósito de Engenharia Reversa, absorvem uma significativa porcentagem de todo o esforço relacionado à produção de software (PRESSMAN, 2002). Levando em consideração as dificuldades encontradas na Engenharia Reversa, o FAD alcançou um bom desempenho em função dos resultados obtidos quando foi aplicado no sistema Androsys. Além disso, esse aplicativo poderá ter suas funções ampliadas e empregadas em outros sistemas similares.

AGRADECIMENTOS: À Richardson Floriani Voltolini, Jean Metz, Rafael Mendes Pereira e Daniel de Faveri Honorato, estagiários do LABI, por terem ajudado no projeto do FAD.

REFERÊNCIAS:

BORLAND DELPHI. **Delphi**. Disponível em <http://www.borland.com/delphi>. Acesso em: 4 abril 2003.

BORLAND TOGETHERSOFT. **TogetherSof**. Disponível em <http://www.togethersoft.com>. Acesso em: 4 abril 2003.

CHIKOFSKY, Elliot J., SELFRIDGE, Peter G., WATERS, Richard C. **Challenges to the Field of Reverse Engineering**, Computer Society Press, IEEE, pp. 144-150, Baltimore, Maryland, USA, 1993.

DAHM, M., **Grammar and API for Rational Rose Petal Files**, Julho, 2001. Disponível em <http://crazybeans.sourceforge.net/CrazyBeans/doc/grammar.pdf >. Acesso em 10 abril 2003.

GRAHAM, P., **ANSI Common Lisp**, Prentice Hall, 1995.

OBJECT INSIGHT. **Object Insight "UML for the rest of us".** Disponível em <http://www.o-bjectinsight.com>. Acesso em: 4 abril 2003.

PRESSMAN, Roger S. **Engenharia de Software**. 5. Ed. Rio de Janeiro: McGraw-Hill, 2002.

RATIONAL. **Visual Modeling with Rational Rose Home**. Disponível em <http://www.ra-tional.com/rose> . Acesso em: 4 abril 2003.

TROIANO, Bruno A., BECKER, Adriana, WU, Feng C., ESTEVES, Sandro C., MACHADO, Renato B., LEE, Huei D. **Sistema de Gerenciamento de Dados Para Reprodução Humana**, XI EAIC - Encontro Anual de Iniciação Científica, 2002, Maringá - PR. Disponível em <http://www.foz.unioeste.br/labi/documentos/análise_eaic.pdf>. Acesso em: 16 abril 2003.

VERONESE, Gustavo O., NETTO, Felipe J., WERNER, Claudia M. L., CORREA, Alexandre L. **Uma Ferramenta de Auxílio à Recuperação de Modelos UML de Projeto a partir de Código Java**, XVI Simpósio Brasileiro de Engenharia de Software, Sessão de Ferramentas, Gramado, Outubro, 2002. Disponível em <http://www.sbc.org.br/reic/edicoes/2002e4/cien-tificos/UmaFerramentaDeAuxilioARecuperacaoDeModelosUMLdeProjeto.pdf>. Acesso em: 14 abril 2003.

WEBGAIN. **StructureBuilder.** Disponível em <http://www.webgain.com/products/structure_ builder>. Acesso em: 4 abril 2003.