DDE – Draco Domain Editor

Vinicius Cardoso Garcia
Valdirene Fontanette
Adriano Bossonaro
Ângela de Britto Perez
Antonio Francisco do Prado

Universidade Federal de São Carlos – Departamento Computação
{vinicius, valdirene, bossonaro, angela, prado}@dc.ufscar.br

Resumo

Este artigo apresenta uma ferramenta para edição de domínios do Sistema Transformacional Draco, denominada Draco Domain Editor. No ST Draco, um domínio é definido por uma Linguagem e esta por sua vez é formada por: uma gramática, um parser e Prettyprinter, ou unparser, definidos a partir dessa gramática. A ferramenta DDE facilita a edição destas partes do domínio, através de recursos textuais e gráficos. Outros recursos do DDE visam a edição de transformações e de scripts que orientam o processo de aplicação das transformações, e o gerenciamento de projetos que utilizam o ST Draco.

Palavras-Chaves: *Domínios, Gramáticas, Regras de Produção, Transformações de Software.*

Abstract

This article presents a domain edition tool for Draco Transformation System, called Draco Domain Editor. In the ST Draco, a domain is defined by a Language composed by: a grammar, a parser and Prettyprinter, or unparser, defined from this grammar. DDE facilitates the domain edition, through textual and graphic resources. Other DDE resources aims the edition of transformation and scripts that guide the transformation application process, and the management of projects that use the ST Draco.

Key words: *Domains, Grammars, Production Rules, Software Transformation.*

1. Introdução

Devido à constante evolução tecnológica, muitos sistemas de software, apesar de atenderem a seus requisitos e serem completamente estáveis, têm-se tornados obsoletos. Isto ocorre em virtude principalmente das mudanças das linguagens de implementação e das plataformas de software e hardware para a qual foram projetados, aumentando assim a quantidade de sistemas legados que não são reaproveitados. A reengenharia destes sistemas pode ser uma solução para construí-los reutilizando os conhecimentos embutidos no seu código e nas documentações disponíveis. Dessa forma, economiza-se tempo e recursos por meio da reutilização de software.

Prevendo que a evolução tecnológica exige a reconstrução do software e que a reutilização de artefatos de software diminui os custos e o tempo de desenvolvimento, foi proposto o paradigma Draco [1]. Baseado nas idéias do paradigma Draco foi construído o Sistema Transformacional Draco, que vem sendo usado na reengenharia de software.

Orientado a domínios, o ST Draco suporta a construção de software através de transformações. Descrições de um domínio são mapeadas para descrições de outros domínios usando transformações. Na reengenharia, parte-se de descrições no domínio do código legado para se obter descrições no domínio alvo da reengenharia, permitindo recuperar o projeto do sistema legado e gerar sua implementação em uma nova linguagem. Na construção de um domínio, o ST Draco gera o *parser* e o *Prettyprinter* a partir das definições da gramática do domínio. As transformações são geradas a partir de definições, considerando a sintaxe e a semântica das linguagens de origem e alvo das transformações.

Foram desenvolvidas diversas pesquisas para comprovar a viabilidade do ST Draco na transformação de software de diferentes domínios [3, 4, 5, 6, 7].

Estas pesquisas também serviram para testar o ST Draco e identificar as dificuldades na construção dos domínios. Assim, os erros foram corrigidos, tornando o ST Draco mais estável e

confiável. Atualmente, dando continuidade às idéias de Reengenharia de Software usando Transformações, estão sendo desenvolvidos dois novos projetos, apoiados pelo RHAE/CNPq e DC/UFSCar [8, 9], com objetivo de definir uma metodologia para Reengenharia de Software Orientada a Componentes Distribuídos.

Na realização dos projetos, observou-se que uma das grandes dificuldades para construção de domínios no ST Draco relacionam-se com a falta de ferramentas adequadas para a edição e depuração de gramáticas, de transformadores e de *scripts* para a aplicação das transformações.

Dada esta necessidade foi construída a ferramenta *Draco Domain Editor* (DDE), integrada ao Sistema Transformacional Draco, que suporta a edição de gramáticas, transformadores e *scripts* de execução das transformações. Dessa forma tem-se no mesmo ambiente do ST Draco uma ferramenta que auxilia o Engenheiro de Software na construção e utilização de um domínio. Além disso, o DDE também suporta o gerenciamento dos projetos desenvolvidos no ST Draco.

Na seção 2 deste artigo apresentam-se as principais características da ferramenta DDE, detalhando o Editor de Gramáticas, de Transformadores e de s*cripts* de execução das transformações, e o gerenciamento de projetos do ST Draco. Finalmente, na seção 3, apresentam-se as conclusões deste artigo.

2. A Ferramenta DDE

A ferramenta DDE tem dois módulos para suportar a edição das partes de um domínio, conforme as opções do menu da sua interface principal, mostrada na Figura 1. A opção (1) "*Grammar"* suporta a edição da gramática, e a opção (2) "*Transformer*" suporta a edição dos transformadores.

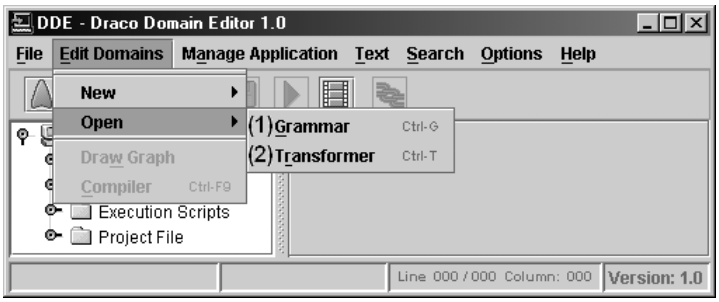

Figura 1 - Interface principal do DDE

Inicialmente o Engenheiro de Software deve definir o domínio de trabalho, selecionando-o no *browser*, carregado quando a ferramenta DDE é iniciada. Caso o domínio não exista então este deve ser criado e em seguida têm-se as áreas para a edição textual e gráfica de sua gramática, conforme mostra a Figura 2, para o domínio Dataflex.

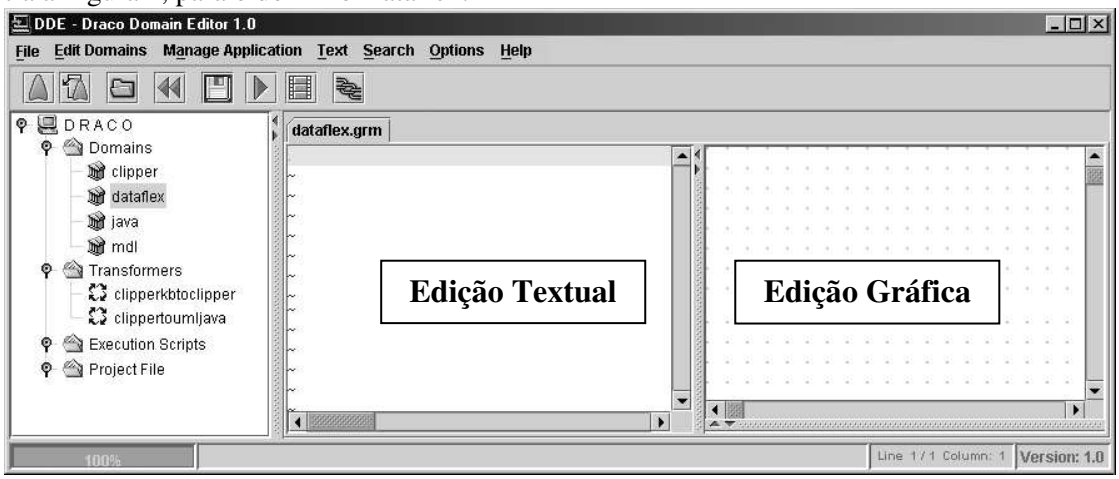

Figura 2 - Ambiente de trabalho do DDE

2.1. Edição de Domínio

O Engenheiro de Software deve editar a gramática da linguagem do domínio e submetê-la aos subsistemas **PARGEN** [1] e **PPGEN** [1] do ST Draco para gerar o *parser* e o *Prettyprinter*

respectivamente. O conjunto de transformações é definido com base no conhecimento da sintaxe e da semântica das linguagens do domínio de origem e alvo das transformações. Segue-se uma apresentação mais detalhada sobre a edição das gramáticas e das transformações.

2.1.1. Editor de Gramáticas

O Editor de Gramáticas suporta a edição de gramáticas livres de contexto, com recursos textuais e gráficos, que ajudam o usuário na definição de suas regras de produção. Baseado na descrição textual da gramática o DDE gera uma representação gráfica da sua árvore gramatical, que possibilita a navegação nas regras de produção da gramática e a visualização dos seus símbolos terminais e não-terminais.

O Editor sinaliza com cores (*sintaxy hightligth*) seus símbolos terminais e não-terminais. Diferentes cores sinalizam cada nó da árvore gramatical, associado a um símbolo terminal ou não-terminal. O amarelo denota um nó terminal (token). O cinza denota um nó cuja derivação leva a uma regra de expressão regular no analisador léxico. O branco identifica um nó não-terminal sem problemas de derivação. O vermelho identifica um nó não-terminal com problemas de derivação. O ícone SA (*Semantic Action*) denota a presença de ações semânticas associadas ao símbolo, e o metasímbolo "|" (*pipe)* é representado pelo rótulo "**OR**".

Alguns objetos, como comentários e comandos de formatação do *Prettyprinter*, são ignorados na representação gráfica da árvore gramatical. Dessa forma tem-se uma representação com os símbolos essenciais da gramática, facilitando sua visualização.

A Figura 3 mostra, por exemplo, a tela de edição da gramática Dataflex [6]. À esquerda tem-se sua descrição textual e à direita a respectiva árvore gramatical. Cada símbolo terminal e não-terminal aparece como um nó da árvore. Assim por exemplo, a regra *"screen"*, leva aos símbolos *"id_tela"*, *"screen_cnt"* e *"end_tela"*.

Navegando na árvore gramatical o Engenheiro de Software pode editar as regras de produção da gramática. Da mesma forma procede-se com as ações semânticas colocadas ao lado das regras de produção. Por exemplo, a regra *"id_tela"* tem associada a ação semântica *"init_screen()"* e a regra "*end_tela*" tem a ação "*end_screen()*".

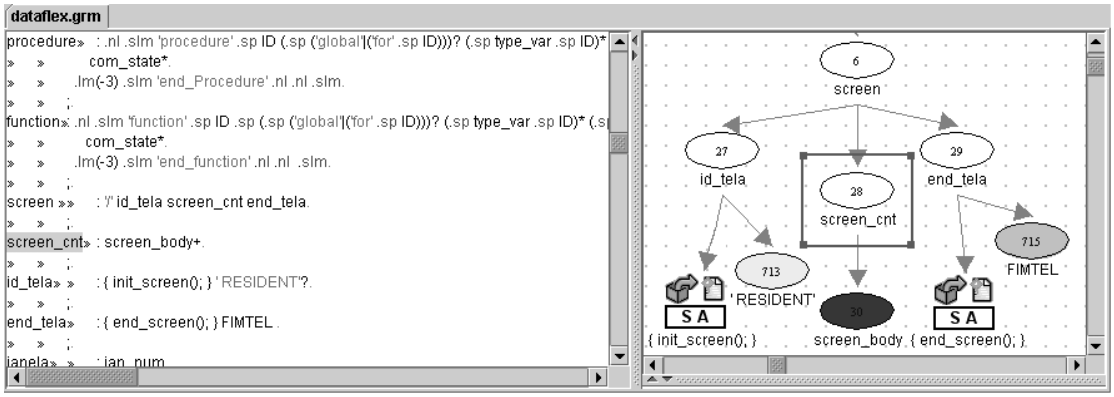

Figura 3 - Editor de Gramáticas

Depois de criada, a gramática, esta deve ser submetida ao componente **PARGEN** do ST Draco para a geração do *parser*, e ao **PPGEN** para a geração do *Prettyprinter*. Isto pode ser feito através do comando "*Compiler*", no menu "*Edit Domain*" do DDE. Os conflitos do tipo "*shift reduce*" e "*reduce reduce*", são detectados pelo componente **PARGEN** do ST Draco, porém ainda não são tratados pelo DDE.

2.1.2. Editor de Transformadores

As transformações são escritas com comandos da meta-linguagem do subsistema **TFMGEN** [1] do ST Draco. Esta meta-linguagem possui comandos que atendem a quase todas as necessidades de um sistema de transformação. Usando comandos do **TFMGEN** o Engenheiro de Software descreve as transformações orientadas pelas sintaxes e semânticas das linguagens de origem e alvo das transformações. Caso a linguagem de origem e alvo seja a mesma, os transformadores são chamados intradomínios, caso contrário são chamados interdomínios. Assim, por exemplo, as transformações que mapeiam um código Dataflex para Visual Dataflex são denominadas interdomínios e as que

organizam o código Dataflex, segundo os princípios da Orientação a Objeto, são denominadas intradomínios.

Opcionalmente, à semelhança do que existe nos geradores de compiladores, como o **YACC** [10] e o **BISON** [11], o Engenheiro de Software pode adicionar comandos da linguagem C++, na definição das transformações, para resolver problemas relacionados com as diferenças de sintaxe e semântica das linguagens transformadas.

Para representação gráfica do transformador, foi adotada uma notação baseada na proposta de Sant'anna [2] para Circuitos Transformacionais, onde um conjunto de transformações é representado por um ícone denominado *SOT* (*Set Of Transformers*), uma transformação é representada por um ícone denominado *Transform* e um padrão formatado de uma dada categoria semântica, é representado por um ícone denominado *Template*. Da mesma forma que na edição das gramáticas, o Engenheiro de Software pode navegar nestas representações para editar as transformações.

A Figura 4 mostra um exemplo de transformador (DfpToDfpOO.tfm) [6], que organiza o código Dataflex Procedural segundo os princípios da Orientação a Objetos. O conjunto de transformações (*SOT*) "*LocalizaLinhasComandoAntesEnter*" possui duas transformações: "*LocalizaComparacoesdoBloco*" e "*LocalizaComandos*". A transformação "*LocalizaComparacoesdoBloco*" utiliza dois *Templates*: "*T_Grava_Bloco*" e "*T_Bloco*". Selecionando uma transformação o Engenheiro de Software pode editar seus pontos de controle(LHS, POST-MATCH e outros), baseado na sintaxe e semântica das linguagens de origem e alvo das transformações.

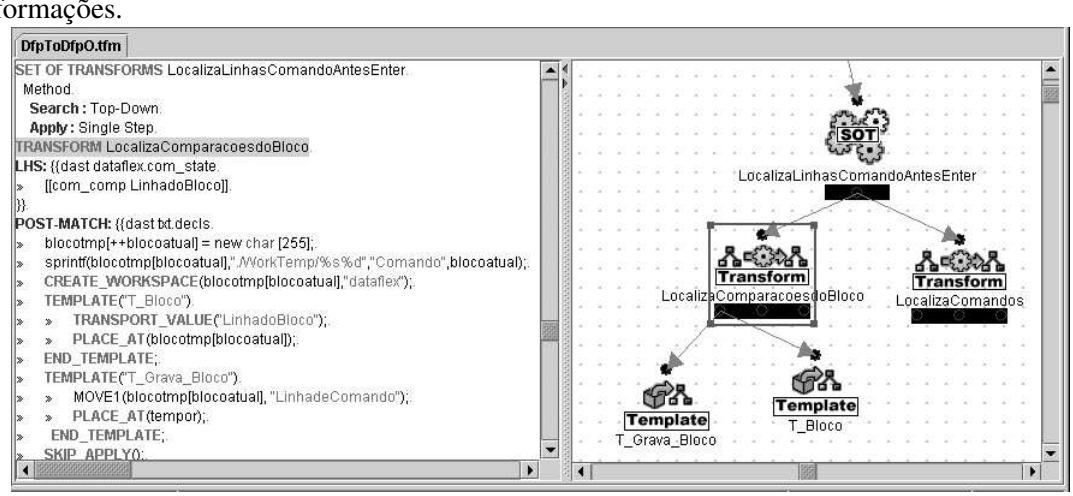

Figura 4 - Editor de Transformadores

2.2. Gerenciamento de Aplicações

O DDE auxilia o Engenheiro de Software na gerência dos projetos de transformação de software, através da edição de arquivos "*.prj*", executados através do comando "*Execute*" no menu "*Manage Application*" do DDE. Um arquivo "*.prj*" define os arquivos do projeto e os *scripts* de execução das transformações. Os arquivos de um projeto, submetidos às transformações, são definidos no bloco de comandos "*BEGIN_SOURCES*" e "*END_SOURCES*", conforme mostra a Figura 5. Os *scripts* de execução (*.est*), que gerenciam a aplicação das transformações, são definidos no bloco de comandos "*BEGIN_EXECUTE*" e "*END_EXECUTE*".

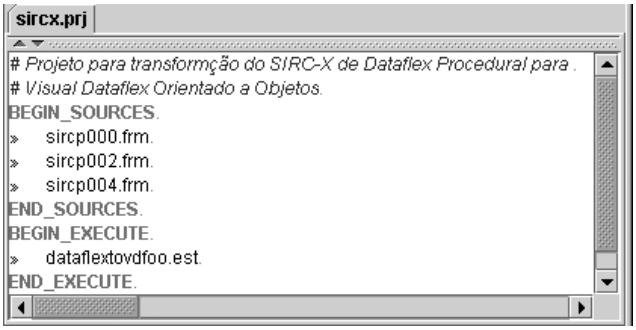

Figura 5 - Projeto de transformação de software

No caso da Figura 5, tem-se que os arquivos "*sircp000.frm"*, "*sircp002.frm"* e "*sircp004.frm"*, do projeto "*sircx.prj*" do domínio Dataflex [6], serão submetidos ao *script* "*dataflextovdfoo.est*". Conforme comentários, indicados com "#", o script "*dataflextovdfoo.est*" do projeto "*sircx.prj*", chama as transformações que mapeiam o código Dataflex para Visual Dataflex Orientado a Objeto.

Os *scripts* de execução das transformações (*.est*) suportam a definição da seqüência de aplicação dos transformadores nas descrições de um domínio. Contêm comandos que geram outros *scripts*: do sistema operacional (arquivos ".bat") e do ST Draco (arquivos ".dsf").

A Figura 6 mostra, por exemplo, o *script* de execução "*dataflextovdfoo.est*", onde o comando "*CALL PREPARA.BAT*" dispara o *script* "*PREPARA.BAT*" do Sistema Operacional que prepara uma área de trabalho para a aplicação das transformações. Seus comandos são gerados a partir das definições contidas no bloco "*BEGIN_BAT*", seguido do nome do *script,* e "*END_BAT*", do *script ".est"*.

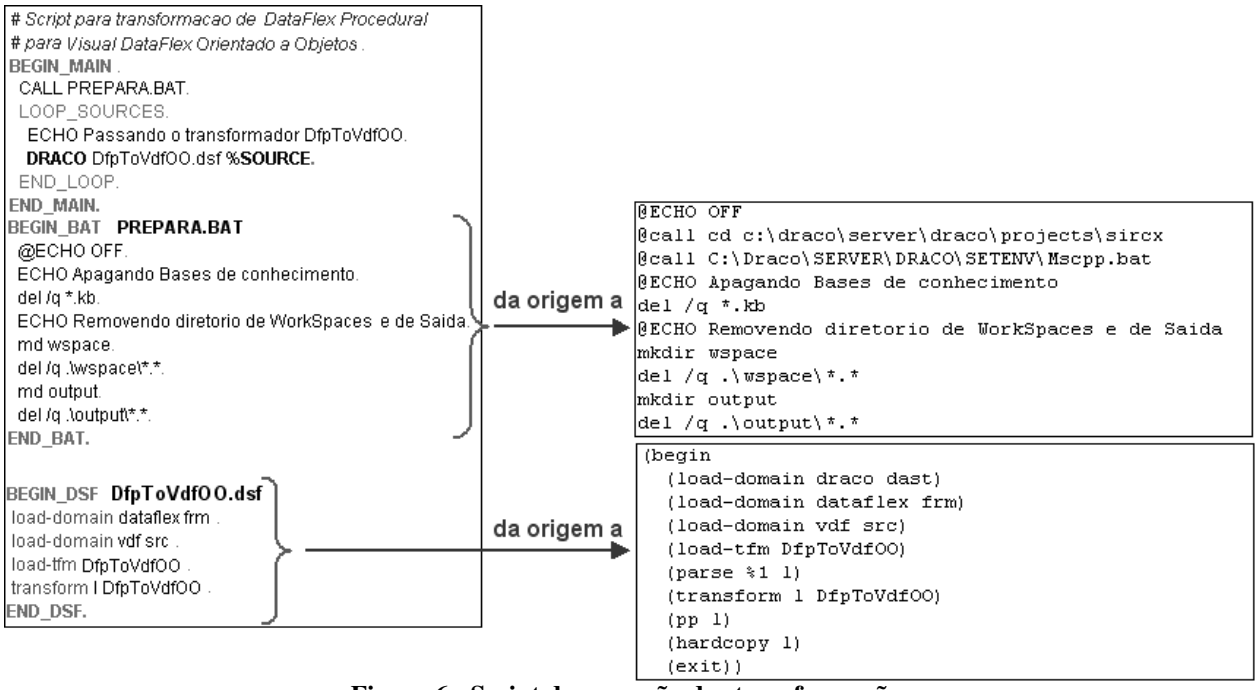

Figura 6 - Script de execução das transformações

De forma semelhante o comando "*DRACO DfpOOToVdf.dsf %SOURCE*" chama um *script* do ST Draco, denominado "*DfpOOToVdf.dsf*" que inicialmente carrega os domínios (*load-domain*) e os transformadores (*load-tfm*). Em seguida analisa (*parse*) a descrição do domínio a ser transformado (Dataflex). O comando "*transform*" aplica a transformação "*DfpOOToVdf*" na descrição de entrada do domínio, no caso Dataflex (*%SOURCE*). Finalmente o comando "*pp*" (*Prettyprinter*) cria a nova descrição transformada, orientada pela sintaxe da linguagem do domínio alvo da transformação (Visual Dataflex), e o comando "*hardcopy*" grava esta descrição. O *script* "*DfpOOToVdfp.dsf*" é gerado a partir das definições contidas no bloco "*BEGIN_DSF*" seguido pelo nome do *script* ".dsf" e "*END_DSF*".

3. Conclusão

O uso do DDE na construção de Domínios e no processo de aplicação das transformações facilita o trabalho do Engenheiro de Software nos projetos de reengenharia usando transformações. Utilizando a representação gráfica da árvore gramatical da gramática do domínio é possível navegar pelas suas regras de produção e editar seus símbolos terminais e não-terminais, e suas respectivas ações semânticas. A geração de uma representação gráfica para os transformadores facilita a edição e o entendimento da sua seqüência de aplicação, conforme a sintaxe e semântica das linguagens dos domínios. O s*cript* de execução ("*.dsf*") reunindo comandos do Sistema Operacional e do ST Draco, facilita o gerenciamento de aplicação das transformações, que antes eram realizados em vários *scripts* separados, do Sistema Operacional e do ST Draco. Novos estudos estão sendo realizados para estender

o DDE com recursos que incluem a visualização e gerenciamento das transformações associadas às regras de produção das gramáticas dos domínios de origem e alvo das transformações.

4. Referências bibliográficas

[1] PRADO, A.F.; **"Estratégia de Engenharia de Software Orientada a Domínios".** Rio de Janeiro, 92. Tese de Doutorado. Pontifica Universidade Católica, p.333.

[2] SANT´ANNA, M.; "**Circuitos Transformacionais"**. Rio de Janeiro, 99. Tese (Doutorado) – Departamento de Informática, Pontifica Universidade Católica.

[3] PRADO, A. F., PENTEADO, R.A.D., Abrahão, S. M. e Fukuda, A. P.; "**Reengenharia de Programas Clipper para Java"**, CLEI, 98. p. 383-394.

[4] FUKUDA, A. P.; "**Refinamento Automático de Sistemas Orientados a Objetos Distribuídos"**. São Carlos/SP, 2000. Dissertação de Mestrado. Universidade Federal de São Carlos.

[5] NOVAIS, E.R.A.; "**Reengenharia de Software Orientada a Componentes Distribuídos"**. São Carlos/SP, 2002. Dissertação de Mestrado. Universidade Federal de São Carlos.

[6] NOGUEIRA, A. R.; **"Transformação de DataFlex Procedural para Visual DataFlex Orientado a Objetos reusando um Framework"**. São Carlos/SP, 2002. Dissertação de Mestrado. Universidade Federal de São Carlos.

[7] LEITE, J.C.S, SANT'ANNA, M., PRADO, A.F.; "**Porting COBOL Programs Using a Transformacional Approach"**. Journal of Software Maintenance: Reseach and Practice, vol 9, 3-31 , Out 1996 John Wiley&Sons Ltd

[8] RHAE/CNPQ, Projeto: **"Reengenharia de Software Usando Transformações (RST)"**, NRO: 610.069/01-2.

[9] Bossonaro, A. A., **"Estratégia de Reengenharia de Software usando Transformações"**. URL: http://www.recope.dc.ufscar.br. Acessado em 02/08/2002.

[10] Free Software Foundation, Inc., 59 Temple Place - Suite 330, Boston, MA 02111, USA, 1998. URL: http://dinosaur.compilertools.net/bison/index.html. Acessado em 10/06/2002.

[11] Yet Another Compiler-Compiler. URL: http://dinosaur.compilertools.net/yacc/index.html. Acessado em 10/06/2002.