UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE CIÊNCIAS DA COMPUTAÇÃO – BACHARELADO

# GERADOR DE CÓDIGO JAVA A PARTIR DE ARQUIVOS DO ORACLE FORMS 6I

CLAUDIO SCHVEPE

BLUMENAU
2006

CLAUDIO SCHVEPE

GERADOR DE CÓDIGO JAVA A PARTIR DE ARQUIVOS DO ORACLE FORMS 6I

Proposta de Trabalho de Conclusão de Curso submetida à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso I do curso de Ciências da Computação — Bacharelado.

Profa. Joyce Martins - Orientadora

BLUMENAU
2006

# 1 INTRODUÇÃO

Os sistemas legados são antigos sistemas de software que são necessários para o apoio nas mais variadas atividades executadas dentro de uma organização. Como estes sistemas são indispensáveis, as organizações precisam mantê-los em operação (SOMMERVILLE, 2003, p. 533). Usa-se a reengenharia de software para reimplementar os sistemas legados, visando facilitar sua manutenção e adaptação às tecnologias adotadas pela organização mantenedora do software. As duas principais vantagens da reengenharia em relação a abordagens mais radicais para a evolução de sistemas são: risco reduzido e custo reduzido. Mas, segundo Sommerville (2003, p. 537), somente é viável a migração de um sistema da linguagem em que foi originalmente implementado para uma outra linguagem se for possível automatizar parte do processo de tradução do código.

Para automatizar a etapa de tradução de código, por exemplo, constroem-se geradores de código. No entanto, o processo de desenvolvimento de um gerador de código é considerado uma tarefa complexa. Para diminuir a complexidade e agilizar o processo de desenvolvimento do gerador, pode-se implementar analisadores léxicos e sintáticos para a linguagem na qual o código fonte foi desenvolvido, bem como utilizar motores de *templates*[1].

Diante do exposto, neste trabalho será desenvolvido um gerador de código para permitir automatizar o processo de migração de aplicações desenvolvidas em Oracle Forms 6i para aplicações *desktop* em Java. A entrada do gerador serão arquivos fontes do Oracle Forms 6i, enquanto a saída, construída a partir de *templates*, serão classes Java. Observa-se que os arquivos do Oracle Forms 6i estão em um padrão binário composto de *Forms Objects*[2] e de código *Procedural Language / Structured Query Language*[3] (PL/SQL). Desta forma, será necessário utilizar a biblioteca *Forms Application Programming Interface* (Forms API) para a manipulação dos arquivos fontes.

Com o desenvolvimento da ferramenta proposta será possível automatizar parte do processo de migração de aplicações Oracle Forms para Java, permitindo que projetos que até

[1] Motores de *templates* são mecanismos que permitem separar código dinâmico e o código estático, possibilitando criar moldes – *templates* – para o código a ser gerado. *Templates* são arquivos que contêm variáveis e blocos especiais, que são dinamicamente transformados a partir de dados enviados pelo motor de *templates*, gerando o código-alvo nos moldes do arquivo de especificação do *template* (ROCHA, 2005).
[2] *Forms Objects* são objetos de interface de uma aplicação Oracle Forms 6i.
[3] PL/SQL estende o SQL adicionando construções encontradas em linguagens procedurais, resultando em uma linguagem estrutural que é mais poderosa do que SQL. A unidade básica em PL/SQL é o bloco. Todos os programas são compostos por blocos, que podem estar aninhados uns dentro dos outros. Geralmente, cada bloco efetua uma ação lógica no programa.



então utilizam uma solução proprietária, possam partir para uma solução de desenvolvimento *open source*.

## 1.1 OBJETIVOS DO TRABALHO

O objetivo deste trabalho é desenvolver um gerador de código para migração de aplicações desenvolvidas em Oracle Forms 6i para Java.

Os objetivos específicos do trabalho são:

a) traduzir um subconjunto de código PL/SQL, incluindo apenas as construções procedurais;
b) permitir a conversão de alguns objetos visuais (`Canvas`, `CheckBox`, `Frame`, `Image`, `Line`, `ListItem`, `PushButton`, `RadioGroup`, `Tab`, `Text`, `TextItem`, `Window`);
c) mapear eventos (gatilhos ou *triggers*) de controle de objetos visuais para eventos Java;
d) utilizar *templates* para configurar o formato do código de saída.

## 1.2 RELEVÂNCIA DO TRABALHO

A construção de geradores de código do tipo proposto pode ser considerada uma tarefa complexa pois, apesar de as técnicas utilizadas já serem amplamente estudadas, envolve tanto o mapeamento entre as construções das linguagens fonte (procedural) e destino (orientada a objetos) como o projeto e a implementação de tradutores de código, mais especificamente analisadores léxico e sintático. Além disso, diferente da maioria dos tradutores de linguagens existentes, serão usados arquivos binários como entrada do gerador.

Por fim, considera-se que o uso de *templates* para configurar o formato do código a ser gerado torna o processo de migração de software bastante flexível.

## 1.3 METODOLOGIA

O trabalho será desenvolvido observando as seguintes etapas:

a) levantamento bibliográfico: realizar levantamento de bibliografia sobre construção de geradores de código, tradução de código, uso de *templates,* Oracle Forms 6i e Java;

b) elicitação dos requisitos: detalhar e reavaliar os requisitos, observando as necessidades levantadas durante a revisão bibliográfica;

c) especificação da linguagem PL/SQL: especificar o subconjunto da linguagem PL/SQL a ser traduzido, usando expressões regulares e a notação *Backus-Naur Form* (BNF) para especificar, respectivamente, os símbolos léxicos e as regras sintáticas;

d) análise das linguagens PL/SQL e Java: identificar as diferenças entre as linguagens fonte e destino. O mapeamento entre as construções PL/SQL e as construções Java será especificado usando esquemas de tradução;

e) especificação do gerador: especificar o gerador com análise orientada a objeto utilizando a *Unified Modeling Language* (UML). Será usada a ferramenta Enterprise Architect para o desenvolvimento dos diagramas de caso de uso, de classes e de seqüência;

f) implementação da interface do gerador: implementar a *Graphic User Interface* (GUI) utilizando o ambiente de desenvolvimento Microsoft Visual C++ 6.0, com o apoio da biblioteca wxWidgets-2.6.2;

g) implementação dos analisadores: implementar os analisadores utilizando Flex para gerar o analisador léxico, Bison para gerar o analisador sintático e a biblioteca Forms API para auxiliar na leitura dos arquivos fontes do Oracle Forms 6i. Gerar estes analisadores para a linguagem C++, e integrá-los usando o ambiente de desenvolvimento Microsoft Visual C++ 6.0;

h) estudo de motores de *templates* para C++: pesquisar motores de *templates* disponíveis para C++, como por exemplo eNITL (JOU, 2000), avaliando qual será o mais indicado para a utilização na implementação do gerador de código;

i) integração do motor de *templates*: integrar o motor de *templates,* definido na etapa anterior, aos analisadores léxico e sintático de forma a permitir a tradução do código fonte Oracle Forms para Java. Será usado o ambiente de desenvolvimento Microsoft Visual C++ 6.0;



j) testes: criar alguns exemplos de aplicações Oracle Forms 6i que contenham somente objetos válidos para a tradução. Verificar se o código gerado pela ferramenta preserva as funcionalidades da aplicação. Na implementação dos exemplos será usado o Oracle Forms 6i.

As etapas serão realizadas nos períodos relacionados no Quadro 1.

| | 2006 | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | maio | | jun. | | jul. | | ago. | | set. | | out. | | nov. | |
| etapas / quinzenas | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 | 1 | 2 |
| levantamento bibliográfico | | | | | | | | | | | | | | |
| elicitação dos requisitos | | | | | | | | | | | | | | |
| especificação da linguagem PL/SQL | | | | | | | | | | | | | | |
| análise das linguagens PL/SQL e Java | | | | | | | | | | | | | | |
| especificação do gerador | | | | | | | | | | | | | | |
| implementação da interface do gerador | | | | | | | | | | | | | | |
| implementação dos analisadores | | | | | | | | | | | | | | |
| estudo de motores de *templates* para C++ | | | | | | | | | | | | | | |
| integração do motor de *templates* | | | | | | | | | | | | | | |
| testes | | | | | | | | | | | | | | |

Quadro 1 – Cronograma

# 2 REVISÃO BIBLIOGRÁFICA

Nas seções seguintes são detalhados os motivos para traduzir sistemas legados para outras linguagens, bem como as etapas necessárias para construção de geradores para apoiar este processo da reengenharia de software e quais as vantagens de se utilizar deste tipo de ferramenta. Também é explanado sobre o uso de motores de *templates* no processo de geração de código. Na seqüência é dada uma visão geral de Oracle Forms 6i e de Java. Por fim, são descritas algumas ferramentas existentes no mercado com funcionalidades similares a da ferramenta proposta.

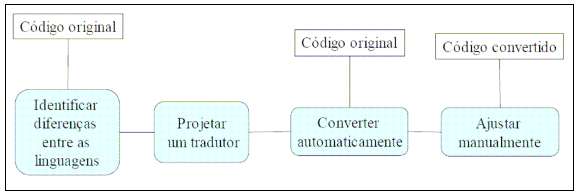
## 2.1 TRADUÇÃO DE CÓDIGO FONTE

Segundo Sommerville (2003, p. 536), a tradução de código é um processo da reengenharia no qual o código fonte é traduzido automaticamente para outra linguagem. Neste processo, a estrutura e a organização do programa não são alteradas. A linguagem alvo pode ser uma versão mais atual da linguagem fonte ou pode ser outra linguagem totalmente diferente. Segundo Sommerville (2003, p. 536), os motivos que levam à tradução de um software para uma nova linguagem são:

a) atualização da plataforma de hardware ou software adotada pela empresa;
b) escassez de profissionais qualificados para manter sistemas desenvolvidos em linguagens incomuns ou que já tenham caído em desuso;
c) mudanças na política organizacional devido à padronização do ambiente de desenvolvimento para reduzir custos;
d) falta de suporte ao software por parte dos fornecedores.

A tradução de código só é economicamente viável se tiver um tradutor automatizado disponível para fazer grande parte da tradução. O tradutor pode ser um programa escrito especialmente para este fim ou pode ser uma ferramenta comprada para converter da linguagem fonte para a linguagem alvo. A figura 1 mostra o processo de tradução de código.






Fonte: Sommerville (2003, p. 537).

Figura 1 – Processo de tradução de código fonte

No processo de tradução é necessário observar as diferenças entre as linguagens envolvidas a fim de verificar quanto do código pode ser traduzido. Após definido o que será realmente traduzido com o apoio do gerador, parte-se para a etapa de especificação do tradutor.

Na grande maioria dos casos, é impossível a tradução completa por um processo automatizado. Isto é, pode não ser possível mapear algumas construções da linguagem fonte para construções da linguagem alvo. Nestes casos, é necessário fazer ajustes no código manualmente (SOMMERVILLE, 2003, p. 537).

## 2.2 GERADORES DE CÓDIGO

Geradores de código são basicamente programas que geram programas. Podem ser desde simples formatadores de código até ferramentas que geram aplicações complexas a partir de modelos abstratos (*templates*). De acordo com Herrington (2003, p. 3), entre as vantagens de se utilizar geradores de código para o desenvolvimento de software estão:

a) qualidade: código escrito manualmente tende a ter um nível de qualidade muito irregular visto que, durante o desenvolvimento da aplicação, podem ser propostas melhores abordagens para solucionar os problemas;
b) produtividade: quando são usados geradores de código, o volume de código digitado manualmente é bem menor se comparado ao volume de código escrito para aplicações desenvolvidas sem o uso dessas ferramentas. Sendo assim, tem-se mais tempo para outras etapas do projeto como, por exemplo, a fase de testes;
c) abstração: a definição de *templates* é bem mais simplificada que o código alvo. Com o uso de *templates* pode-se corrigir erros do projeto ou incluir novas funcionalidades apenas reescrevendo os *templates*. Além disso, o gerador pode ser

facilmente reprojetado para outras linguagens ou tecnologias.

No processo de construção de um gerador de código, a primeira coisa a fazer é verificar se o problema em que se propõe uma solução pode ser resolvido aplicando as técnicas de geração de código. Muitos arquivos de códigos ou códigos repetitivos são sinais de que a geração de código pode ser uma opção a ser usada no desenvolvimento.

Após verificar que a geração de código é a solução mais adequada para solucionar o problema, parte-se para a etapa de definição do processo de desenvolvimento. O processo de desenvolvimento do gerador, de acordo com Herrington (2003, p. 77), segue os passos abaixo:

a) escrever o código de saída manualmente: o primeiro passo na construção do gerador consiste em escrever manualmente o código alvo. Isso facilita a identificação do que é necessário extrair dos arquivos de entrada;
b) projetar o gerador: em seguida deve-se determinar a estrutura do gerador, indicando como a entrada será analisada, como a saída será gerada, quais as rotinas para manipulação de arquivos e diretórios;
c) analisar a entrada: nesta etapa constrói-se os analisadores léxico[4] e sintático[5] para analisar os arquivos de entrada e extrair as informações necessárias para gerar os arquivos de saída;
d) gerar a saída usando *templates*: a última etapa do processo de construção do gerador consiste em implementar a geração do código alvo de acordo com o modelo especificado nos *templates*.

## 2.3 MOTOR DE *TEMPLATES*

Os motores de *templates*, segundo Rocha (2005), são mecanismos que permitem desenvolver geradores de código independentes do código alvo já que este está externo à aplicação. Nesse caso , deve existir um programa responsável por instanciar o motor, carregar os valores das variáveis e blocos especiais do *template* e apresentar o resultado da união do

[4] Segundo Aho, Sethi e Ullman (1995, p. 38), o analisador léxico lê o programa fonte da esquerda para a direita, caractere a caractere, identificando seqüências de caracteres que formam unidades léxicas (*tokens*).
[5] O analisador sintático agrupa os *tokens* recebidos do analisador léxico em frases gramaticais, que são usadas pelo compilador para sintetizar a saída (AHO; SETHI; ULLMAN, 1995, p. 72).



código dinâmico[6] com o código estático[7]. Motores de *templates,* tais como eNITL (JOU, 2000), Velocity (APACHE SOFTWARE FOUNDATION, 2005) e TinyButStrong (SKROL, 2006), podem ser utilizados em diversas áreas da computação como, por exemplo, geradores de interfaces web, geradores de aplicações e ferramentas de documentação de código fonte.

Cada motor de *templates* implementa sua própria linguagem através da qual os *templates* deverão ser escritos. Esta linguagem define os tipos de blocos especiais e como referenciar variáveis nos *templates*. Varia bastante em nível de complexidade, podendo ser definida apenas com estruturas básicas para substituição de valores de variáveis ou até ter estruturas de controle mais complexas como *loops* e comandos condicionais (ROCHA, 2005).

## 2.4 ORACLE FORMS DEVELOPER 6I E JAVA

Segundo Fernandes (2002, p. 670), Oracle Forms Developer (ou Oracle Forms) é um ambiente de produtividade *Rapid Application Development* (RAD), capaz de construir rapidamente aplicações a partir das definições do banco de dados. As aplicações, compostas de *Forms Objects* e de código PL/SQL, podem ser implementadas para arquiteturas cliente-servidor, de três camadas ou para web.

A ferramenta constrói três tipos de módulos:

a) `Form`: é o formulário da aplicação, composto de itens de controle e blocos PL/SQL. Seu fonte possui a extensão FMB;
b) `Menu`: consiste de um conjunto de sub-menus e código para acionamento dos diversos módulos, `Forms` ou sub-menus. Os arquivos do módulo `Menu` possuem extensão MMB;
c) `PL/SQL Library`: são programas PL/SQL que podem ser compartilhados por toda a aplicação. Possui a extensão PLL.

Segundo Oracle Corporation (2000, p. 4), Oracle Forms inclui uma API, a Forms API, que possibilita que programas desenvolvidos utilizando a linguagem C possam criar, carregar, editar e salvar os arquivos com extensão FMB, MMB e PLL. Nesta API encontram-se todas as funcionalidades para acessar as propriedades dos objetos do Oracle Forms.

[6] Código dinâmico é o código definido no *template* de forma simbólica, que passará por tranformações no processo de geração do código de saída. Dessa forma, símbolos como variáveis ou macros serão subtituídos por informações enviadas ao motor de *templates*.

Por sua vez, Java é uma linguagem de programação orientada a objetos desenvolvida pela Sun Microsystems, com sintaxe similar à da linguagem C++ (DEITEL; DEITEL, 2003, p. 59). É independente de arquitetura e pode ser usada no desenvolvimento de vários tipos de aplicações. Comparando Java com Oracle Forms, tem-se que:

a) ao contrário de Oracle Forms, Java possui apenas um tipo de arquivo fonte (extensão JAVA) composto de classes, que depois de compilado é transformado em *bytecode* (extensão CLASS);
b) em Oracle Forms não é possível criar dinamicamente componentes gráficos (formulários da aplicação e objetos visuais, definidos em `Form` ou `Menu`). Já em Java, para a programação de interfaces é fornecida a biblioteca Swing que é mais flexível, possuindo um componente gráfico equivalente para cada componente gráfico existente em Oracle Forms;
c) as estruturas sintáticas encontradas em PL/SQL, como laços de repetições, testes condicionais e expressões, também possuem estruturas equivalentes em Java.

No entanto, para migrar uma aplicação implementada em PL/SQL para Java, Klösch (1996) afirma que, devido às diferenças entre os paradigmas procedural e orientado a objeto, não é possível mapear todos os elementos diretamente, sendo necessárias algumas adaptações no código. A estrutura dos objetos, por exemplo, deve ser formatada. Isso quer dizer que as variáveis e os procedimentos devem ser mapeados para atributos e métodos, sendo encapsulados em objetos. Suas chamadas também serão modificadas para acessar os serviços oferecidos pelos objetos.

## 2.5 TRABALHOS CORRELATOS

Existem várias ferramentas, comerciais ou não, que oferecem suporte para migração de sistemas entre plataformas distintas, tais como Delphi2Java-II (FONSECA, 2005), VB.Net to C# Converter (VBCONVERSIONS, 2006) e VB2Java.COM (TAULLI; JUNG, 1997). Especificamente sobre migração de aplicações Oracle, pode-se citar: Forms2Net (ATX SOFTWARE, 2006) e SwisSQL (ADVENTNET, 2006).

Com a ferramenta Forms2Net é possível migrar aplicações de Oracle Forms para

[7] Código estático é o código definido no *template* de forma literal que não será alterado pelo motor de *templates*.



aplicações equivalentes em Microsoft ASP.NET. Segundo ATX Software (2006), os motivos que podem levar a utilizar Forms2Net são:

a) padronizar a plataforma de desenvolvimento;
b) aumentar a produtividade;
c) minimizar tempo, custo e risco do processo de migração;
d) facilitar a manutenção da aplicação desenvolvida;
e) preservar a semântica (funcionalidade) do sistema original.

Com Forms2Net é possível continuar usando um banco de dados da Oracle ou migrar para SQL Server. A conectividade com a camada de dados está baseada em componentes de acesso a dados através da tecnologia de Microsoft ADO.NET, que permite ao cliente trabalhar com bancos de dados diferentes de um modo independente. Observa-se que a migração do *schema* do banco de dados e dos próprios dados não é feita pelo Forms2Net, sendo necessário, nesse caso, o uso de outras soluções complementares.

No processo de tradução, de 75% a 95% do código final são gerados automaticamente pelo *wizard* do Forms2Net - o valor exato depende da complexidade e da estrutura da aplicação original (ATX SOFTWARE, 2006).

Conforme AdventNet (2006), SwisSQL é uma ferramenta para ajudar a converter para a linguagem Java procedimentos PL/SQL armazenados no Sistema Gerenciador de Banco de Dados (SGDB). Suas principais características são:

a) suporte a todas as construções PL/SQL, bem como pacotes, procedimentos e funções;
b) padrão de desenvolvimento baseado em API *open source,* isto é, as funções utilizadas no código gerado são amplamente difundidas e de uso gratuito;
c) código gerado é facilmente portável para outras arquiteturas.

## 3 REQUISITOS DO SISTEMA A SER DESENVOLVIDO

O gerador de código deverá:

a) processar os seguintes arquivos fontes da ferramenta Oracle Forms: `Form` – formulário da aplicação (arquivo com extensão FMB), `Menu` – conjunto de menus (arquivo com extensão MMB) e `PL/SQL Libray` – conjunto de programas PL/SQL (arquivo com extensão PLL) (requisito funcional - RF);
b) analisar léxica e sintaticamente os arquivos de entrada (RF);
c) produzir, para cada arquivo de entrada, a tradução especificada no *template* (RF);
d) traduzir o subconjunto de código PL/SQL composto de procedimentos, funções e blocos, que contenham atribuições, testes condicionais, declarações de variáveis, laços de repetição, blocos de exceções e expressões, sendo que não será traduzido nenhum código SQL (RF);
e) traduzir os objetos visuais `Canvas`, `CheckBox`, `Frame`, `Image`, `Line`, `ListItem`, `PushButton`, `RadioGroup`, `Tab`, `Text`, `TextItem` e `Window`, implementando suas principais funcionalidades (RF);
f) mapear gatilhos de controle de objetos visuais para eventos (RF);
g) ser implementado utilizando o ambiente de desenvolvimento Microsoft Visual C++ 6.0 (requisito não-funcional – RNF);
h) ser implementado utilizando a linguagem de programação C++ (RNF);
i) utilizar a ferramenta Flex para gerar o analisador léxico (RNF);
j) utilizar a ferramenta Bison para gerar o analisador sintático (RNF);
k) utilizar Forms API para manipulação de arquivos do Oracle Forms 6i (RNF);
l) utilizar a biblioteca wxWidgets-2.6.2 para construção da ferramenta (RNF).



## 4 CONSIDERAÇÕES FINAIS

Verificou-se que a reengenharia fornece métodos menos agressivos à evolução de sistemas legados, como a tradução de código. Como este método é muito demorado e seu custo elevado, fica desaconselhada qualquer tentativa de tradução de sistemas legados sem auxílio de ferramentas para automatizar o processo. Portanto, para automatizar o processo de tradução, deve-se construir geradores de código como o proposto nesse trabalho.

O desenvolvimento da ferramenta proposta envolverá primeiramente a análise das diferenças entre as linguagens envolvidas. Posteriormente será gerado o código de saída manualmente e após isso terá início a etapa de construção do tradutor. Serão implementados os analisadores léxico e sintático para analisar os arquivos fontes, extrair as informações relevantes para a tradução e enviá-las para o motor de *templates*, a ser definido durante o desenvolvimento do trabalho, que fará a união entre o código estático, definido no *template,* e o código dinâmico, extraído do arquivo fonte. Nesse caso, o arquivo de *template* nada mais é do que um arquivo com o molde do código alvo desejado.

Observa-se que no Oracle Forms os arquivos fontes estão especificados em um padrão binário. Por este motivo, será utilizada a Forms API para realizar um pré-processamento dos arquivos fontes antes de enviar para o tradutor de código.

Verificou-se que existem aplicações comerciais similares à ferramenta proposta. No entanto, as ferramentas descritas não permitem a personalização do código alvo. A ferramenta Forms2Net permite a conversão tanto de *Forms Objects* como de PL/SQL para a plataforma Microsoft ADO.NET; já a ferramenta SwisSQL apenas permite a conversão de PL/SQL para Java. Com a ferramenta proposta serão convertidos para Java tanto objetos visuais definidos no *Forms Objects* quanto as construções sintáticas PL.

## REFERÊNCIAS BIBLIOGRÁFICAS


ADVENTNET. **SwisSQL**: Oracle to Java migration tool. [S.l.], 2006. Disponível em: <http://www.swissql.com/products/oracle-to-java/oracle-to-java.html>. Acesso em: 15 abr. 2006.

AHO, Alfred V.; SETHI, Ravi; ULLMAN, Jeffrey D. **Compiladores**: princípios, técnicas e ferramentas. Tradução Daniel de Ariosto Pinto. Rio de Janeiro: LTC, 1995.

APACHE SOFWARE FOUNDATION. **Velocity.** [S.l.], 2005. Disponível em: <http://jakarta.apache.org/velocity/>. Acesso em: 03 jun. 2006.

ATX SOFTWARE. **Forms2Net**: from Oracle Forms to Microsoft .NET. [S.l.], 2006. Disponível em: <http://forms2net.atxsoftware.com/site/Welcome.do>. Acesso em: 01 abr. 2006.

DEITEL, Harvey M.; DEITEL, Paul J. **Java:** como programar**.** 4. ed. Tradução Carlos Arthur Lang Lisboa. Porto Alegre: Bookmam, 2003.

FERNANDES, Lúcia. **Oracle 9i para desenvolvedores:** curso completo. Rio de Janeiro: Axcel Books, 2002.

FONSECA, Fabricio. **Ferramenta conversora de interfaces gráficas**: Delphi2Java-II. 2005. 59 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) – Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

HERRINGTON, Jack. **Code generation in action**. Greenwich: Manning Puplications, 2003.

JOU, Herng-Jeng. **eNITL**. [S.l.], 2000. Disponível em: <http://ceu.fi.udc.es/SAL/F/1/ENITL.html>. Acesso em: 03 jun. 2006.

KLÖSCH, René R. **Reverse engineering:** why and how to reverse engineer software. [Vienna], [1996?]. Disponível em: <http://www.infosys.tuwien.ac.**at**/staff/hg/papers/css96.pdf>. Acesso em: 06 jun. 2006.

ORACLE CORPORATION. **Using the Oracle Forms Application Programming Interface (API)**. [S.l.], 2000. Documento eletrônico disponibilizado com o Ambiente Oracle Forms Developer 6i.

ROCHA, Lucas. **APE:** plataforma para o desenvolvimento de aplicações web com PHP. [Salvador], 2005. Disponível em: <http://twiki.im.ufba.br/bin/view/Aside/ProjetoConclusaoDeCursoAPEMonografia#4_2_Motores_de_templates>. Acesso em: 01 abr. 2006.





SKROL, Jean B. **TinyButStrong**: template engine for pro and beginners for PHP. [S.l.], 2006. Disponível em: <http://www.tinybutstrong.com/>. Acesso em: 03 jun. 2006.

SOMMERVILLE, Ian. **Engenharia de software**. 6. ed. Tradução Maurício de Andrade. São Paulo: Addison Wesley, 2003.

TAULLI, Tom; JUNG, David. **VB2Java.COM**: moving from Visual Basic to Java. [S.l.], 1997. Disponível em: <http://www.vb2java.com/about.html>. Acesso em: 04 jun. 2006.

VBCONVERSIONS. **VB.NET to C# converter**: Visual Basic.NET to C# converter. [S.l.], 2006. Disponível em: <http://www.filesrepository.com/preview/vbconversions/vb-net-to-c-converter.html>. Acesso em: 03 jun. 2006.