Engenharia Reversa  
EENGENHARIA  
NGENHARIA  
R  
REVERSA  
EVERSA  
UFF – Universidade Federal Fluminense  
Graduação em Ciência da Computação  
Informática I – 2005/2  
Antonio Jorge Sapage da Canhota Junior  
Diego Alves de Souza  
Diogo dos Santos Moutinho  
Felipe Paixão Lohnefink  
1

Engenharia Reversa  
2

Engenharia Reversa  
Índice  
1. Introduçao  
 4  
Por que a engenharia reversa na informática?  
 4  
Exemplos de engenharia reversa  
 5  
Reengenharia  
 6  
Definição de engenharia reversa de software  
 7  
2. Técnicas de engenharia reversa sem código-fonte  
 7  
3. Técnicas de engenharia reversa com código-fonte  
 8  
Extração dos fatos  
 8  
Tratamento dos fatos  
 9  
4. Aspectos Legais  
12  
Leis e acordos  
12  
Casos famosos  
13  
5. Bibliografia  
14  
3

Engenharia Reversa  
1. INTRODUÇÃO  
   
A Engenharia Reversa é uma atividade que trabalha com um produto existente (um software, uma   
peça mecânica, uma placa de computador, etc.) tentando entender como este produto funciona, o   
que ele faz exatamente e como ele se comporta em todas as circunstâncias. Fazemos engenharia   
reversa quando queremos trocar, modificar uma peça (ou um software) por outro, com as mesmas   
características ou entender como esta funciona e não temos acesso a sua documentação.  
   
Por exemplo, numa fábrica, uma bomba falhou e tem que ser trocada por uma nova. A bomba foi   
instalada há 25 anos e as pessoas que fizeram o trabalho se aposentaram há muito tempo. A empresa   
que vendia essas bombas faliu. A fábrica tem que achar uma nova bomba, com exatamente as   
mesmas características, ou seja, ela tem que ser montada sobre a tubulação existente (dimensões   
definidas, como a bomba está fixada, volume ocupado pela bomba, etc.) que são características   
fáceis de descobrir, mas podem também existir outras menos evidentes (a bomba tem que fornecer   
um débito definido, ela precisa respeitar algumas restrições desconhecidas). Todas essas   
características da bomba podem ser importante ou não, a fábrica tem que descobrir isso antes de   
comprar uma nova.  
   
Por que a Engenharia Reversa na Informática?  
Organizações trabalham com sistemas apresentando problemas tais como:  
   
 O sistema foi iniciado há muitos anos (até 20 anos atrás).   
   
 O sistema tem pouca documentação e ela não foi atualizada. O que quer dizer que a   
documentação descreve um estado anterior do sistema, mas não a configuração atual.   
   
 As pessoas que criaram o sistema deixaram a empresa, ninguém pode explicar muitas   
decisões que foram tomadas.   
   
 Algumas partes do sistema foram implementadas com métodos “estranhos” ou sem método   
nenhum.   
   
 Muitos programadores diferentes implementaram pequenas partes do sistema. Cada um   
usava um método e um estilo particular de programação.   
   
 O sistema é implementado numa linguagem de programação antiga (Cobol, Fortran, APL,   
etc.) para a qual existem poucas ferramentas.   
   
4

Engenharia Reversa  
Mas o sistema tem que evoluir:  
   
 Para ser adaptado a novos computadores (mais barato, mais rápido ou porque ninguém   
mantém mais os velhos).   
   
 Para ser adaptado a novos softwares (novas bibliotecas, novas linguagem de programação,   
novas ferramentas).   
   
 Para ser adaptado a novas regras (troca de moeda em todos os países da Europa).   
   
 Para disponibilizar novas funcionalidades que outras empresas usam.   
   
 Para corrigir bugs (bug do ano 2000).   
   
A engenharia reversa pode ser de programas como nos exemplos acima ou de dados. Por exemplo,   
se queremos construir um editor de texto compatível com o MS-Word, vamos ter que entender a   
representação que ele usa para ler os documentos MS-Word ou poder salvar documentos nesse   
formato. Outro exemplo é de banco de dados, ou seja, para passar de um banco de dados relacional   
a um banco de dados orientado a objeto.  
   
Para software, restrições físicas como as dimensões da tubulação sobre a qual a bomba tem que ser   
montada, são restrições de interface. O velho programa tinha uma interface específica, e costumava   
ser chamado de maneira bem definida. O novo programa tem que respeitar a mesma interface. As   
outras características do programa vão se tornar requisitos não funcionais para o novo programa.  
   
É importante observar que esses problemas não são raros, pelo contrário, são muito freqüentes.  
   
Exemplos de Engenharia Reversa  
Fora da Computação  
Tupolev Tu-4: Em 1945, durante a segunda guerra mundial, três bombardeiros americanos modelo   
B-29 foram forçados a aterrissar em território russo. Os soviéticos os desmontaram e estudaram.   
Usaram a engenharia reversa para copiar o bombardeiro nos mínimos detalhes. O resultado foi o   
bombardeiro Tupolev Tu-4 que voou pela primeira vez em 19 de maio de 1947. A produção em   
série do bombardeiro começou neste mesmo ano.  
5

Engenharia Reversa  
Na Computação  
   
IBM-PC compatível: A IBM abriu mão da patente de sua plataforma, deixando o caminho livre   
para qualquer um produzir uma maquina que fosse compatível com o IBM-PC. Assim surgiram   
vários clones do IBM-PC.  
   
Samba: Software que permite sistemas que não estão rodando o Microsoft Windows a compartilhar   
arquivos com sistemas que estão. A engenharia reversa foi utilizada para descobrir como o   
compartilhamento de arquivos do Windows funcionava, para que então computadores que não   
estivessem com a plataforma Windows pudessem emula a mesma.  
   
Wine: Programa que funciona como a API do Windows. Permite executar aplicativos   
desenvolvidos para Windows 3.1X, 9X, NT e 200x no GNU/Linux.  
   
OpenOffice.org: É um conjunto de aplicativos em OpenSource (código aberto). Está disponível   
para diferentes plataformas: incluindo Microsoft Windows, Unix, Solaris, Linux e Mac OS X. A   
Suite é compatível com o Microsoft Office.  
Reengenharia  
A engenharia reversa consiste em apenas analisar o sistema ou a ferramenta para criar uma   
representação dela. Já a Reengenharia vai além. Analisa-se o projeto, cria-se uma representação do   
mesmo e, através dessa representação, monta-se uma nova estrutura que funcione Exatamente como   
a primeira, mas que não seja meramente uma cópia dela.   
   
6

Engenharia Reversa  
Definição de engenharia reversa de software  
A engenharia reversa de software consiste em analisar um determinado sistema para criar   
representações do próprio em um nível mais alto de abstração.  
Também pode ser encarada como “Voltar atrás no ciclo de desenvolvimento do software”.  
   
Na prática, existem dois tipos de engenharia reversa de software:  
•  
No primeiro caso, o código-fonte já está disponível, mas os aspectos mais globais, talvez   
documentação escassa ou não válida, têm que ser descobertos.  
•  
No segundo caso o código-fonte do software não está disponível, e todos os esforços para   
descobrir uma possível fonte do código para o software são considerados como engenharia   
reversa.  
OBS: As pessoas que trabalham com engenharia reversa de software estão mais familiarizadas com   
o segundo caso, chegando até a desconsiderar o primeiro.  
2. TÉCNICAS DE ENGENHARIA REVERSA SEM O   
CÓDIGO-FONTE  
   
Engenharia reversa de software pode ser efetuada por vários métodos.Três grupos principais da   
engenharia de software são:  
•  
Análise de fluxo de dados: Análise através da observação da troca de informações que   
envolvem “analisadores de bus” e “pacotes de sniffers” por exemplo, para "ouvir" dentro do   
bus de um computador ou uma conecção de rede, revelando o tráfico de dados "escondidos".   
O comportamento dos dados no bus ou na rede podem então ser analisados para produzir   
uma nova implementação do software que imita o mesmo comportamento. Isto é   
especialmente utilizado na engenharia reversa de drivers de dispositivos.   
•  
Desassemblar: Usando um desassembler, conseguimos obter a linguagem de máquina   
diretamente do programa. Este código é lido e entendido nos seus próprios termos, apenas   
com a ajuda de “mneminics” da linguagem de máquina. Isto funciona em qualquer programa   
de computador, mas pode levar um bom tempo, especialmente para alguém que não esteja   
acostumado ao código de máquina.  
•  
Decompilação: Neste método utiliza-se um decompilador, um programa que tenta recriar o   
código-fonte em uma linguagem de alto nível, tendo disponível apenas o código de   
máquina.  
7

Engenharia Reversa  
3. TÉCNICAS DE ENGENHARIA REVERSA COM O   
CÓDIGO-FONTE  
   
Extração das Informações  
O primeiro trabalho que se deve fazer é coletar informações sobre o sistema a ser estudado. As   
atividades da engenharia reversa se fazem sobre essas informações extraídas, mais do que sobre o   
próprio sistema.   
As informações podem ser extraídas de várias fontes: o código fonte, a execução, os dados (por   
exemplo, em banco de dados), a “documentação”, ou outras fontes.  
Código (análise estática)  
A primeira fonte, o código, é a mais usada. A análise do código é chamada também de análise   
estática (por oposição à análise dinâmica que é a execução do sistema).  
   
Ela permite extrair as informações mais básicas do sistema:  
   
 Quais são os componentes básicos do sistema: arquivos, rotinas, tipos, variáveis, classes,   
etc;   
   
 Relações de definição conectam um componente com seu conteúdo (onde ele se encontra);   
   
 Relações de referência conectam um componente com aqueles que o usam (se uma rotina A   
chamar uma outra rotina B. A depende de B porque uma modificação na definição de B   
pode ter conseqüências sobre a execução de A).   
   
Ferramentas para fazer essa análise são chamadas de “parser”. O “parsing” é a primeira etapa da   
compilação do código fonte. Para fazer isso, é preciso conhecer a sintaxe da linguagem de   
programação usada.  
Dependendo das necessidades, pode-se usar parsers específicos que vão procurar só um tipo   
particular de informação. Por exemplo, num programa Pascal, podemos extrair o nome de todas as   
funções definidas.   
Trace de execução (análise dinâmica)  
 A análise estática pode extrair muitas informações de um programa, mas nem todas. Por exemplo,   
qual parte de uma instrução IF é realmente usada pode depender dos dados com que o programa foi   
chamado. Para descobrir esse tipo de informação, precisamos da análise dinâmica, que consiste em   
executar o programa e monitorar os valores das variáveis, quais funções são chamadas, etc.  
8

Engenharia Reversa  
Dados   
Os bancos de dados podem ser usados como fonte de informação para ajudar na engenharia reversa   
de um sistema. Mas a engenharia reversa de dados é também um trabalho específico que pode ser   
feito independentemente de qualquer sistema que possa manipular esses dados. Por exemplo,   
poderíamos querer converter um velho banco de dados sobre um “main frame” para um banco de   
dados relacional e distribuído sobre vários PCs.  
Documentação   
 Chamamos de documentação tudo o que não está usado pelo computador para fazer funcionar o   
sistema, mas se destina aos engenheiros que usam o código: relatórios, comentários no código,   
diagramas da análise ou do projeto, etc.   
   
Como ela se destina aos seres humanos ela é de difícil de analisar automaticamente. A abordagem   
mais usada é usar as palavras da documentação. Isso pode permitir extrair os conceitos importantes   
do domínio de aplicação do sistema. Esta abordagem está baseada sobre a suposição que os   
conceitos importantes aparecem com mais freqüência na documentação.   
Outras fontes de informação   
Finalmente, é possível usar outras fontes de informação. Por exemplo, para definir sub-sistemas,   
poderíamos procurar quem escreveu cada porção do código. É razoável pensar que se duas partes do   
código foram desenvolvidas pela mesma pessoa elas tem maior probabilidade de pertencer ao   
mesmo sub-sistema  
   
Tratamento dos Fatos  
Essas são algumas das principais atividades envolvidas na engenharia reversa. Essas atividades   
usam as informações que foram extraídas do sistema. Podemos resumir o objetivo geral dessas   
atividades da seguinte forma: elas tentam abstrair informações de mais alto nível de abstração dos   
fatos do sistema. O que quer dizer que elas vão eliminar os inumeráveis detalhes. O problema é   
decidir nessa massa de informações o que é importante ou não.   
Anomalias no código   
Num software legado, o código pode conter várias anomalias, como partes do programa que nunca   
podem ser executadas (código morto) e trechos de código que foram copiados e levemente   
modificados. Essas anomalias complicam o código inutilmente, fazendo ele mais longo do que   
deveria ser e multiplicando as coisas que um programador tem que estudar e entender.  
Para resolver esse problema podemos deletar o código morto, e no caso dos clones podemos   
modificar o código para suprimir-los ou então comentar o código dizendo que existem clones.  
9

Engenharia Reversa  
Encapsulamento   
O encapsulamento é mais uma técnica de re-engenharia do que de engenharia reversa. Em vez de   
reestruturar um sistema, ela propõe esconder o velho código dentro de uma nova camada.  
“Slicing”  
“Slicing” (fatiar) é uma técnica de decomposição do código de acordo com a utilização das   
variáveis. O slicing de uma parte do código consiste em extrair dela todas as instruções que têm   
uma influência sobre o valor de uma variável definida a um ponto definido do código.   
   
Isso pode ajudar na localização de um bug (erro no valor da variável), limitando a pesquisa nas   
únicas instruções realmente necessárias.  
(Re-)Modularização   
A (re-)modularização é a decomposição de um conjunto de componentes de software em sub-partes   
(os módulos). Em engenharia de software, normalmente, se espera que os módulos tenham uma   
forte coesão interna e um pequeno acoplamento com o exterior.   
   
Essa atividade de agrupar arquivos em sub-sistema não é trivial. Um sistema legado pode ser   
composto de milhares de arquivos, em várias linguagens de programação (por exemplo: uma   
linguagem procedural, assembler, dados e alguns “scripts” para compilações e execuções).  
   
Para agrupar os componentes são utilizados algoritmos que “medem as distancias” entre os   
componentes com base em informações extraídas do sistema.  
Definição de módulos a partir de informações de inclusão entre arquivos.  
10

Engenharia Reversa  
Porém eles apresentam dois problemas principais:  
   
 O resultado pode ser difícil de entender;   
   
 Não há nada mágico neles. Eles agrupam todos os componentes segundo algumas restrições.   
Se as restrições forem erradas, o resultado também será.   
   
O resultado desta técnica é globalmente bom. Mas sempre há detalhes errados em alguns módulos.   
Esses algoritmos são bons quando existem muitos componentes para tratar e como uma primeira   
abordagem para desbravar o sistema.  
Reconhecimento de Clichés   
Um “clichê” é um padrão que descreve uma maneira geral de implementar um conceito de   
programação. A atividade de reconhecimento de clichês trabalha com um banco de clichês e   
procura esses clichês no código. A idéia é que um clichê é implementado com várias linhas de   
código. Reconhecer um clichê pode simplificar o código porque isso vai substituir um só conceito a   
essas linhas.  
   
O reconhecimento de clichê sofre de várias dificuldades:  
   
 Construção do banco de clichês;   
   
 Dualidade, precisão e cobertura (a descrição do clichê deve ser geral porém corre-se o risco   
de ocorrerem falso-positivos);   
   
 Tempo de execução;   
   
 Clichês deslocalizados ou interligados (podem existir outras instruções entre as que   
implementam o clichê ou dois clichês cujas instruções estão misturadas).   
   
11

Engenharia Reversa  
4. ASPECTOS LEGAIS  
Leis e Acordos  
A engenharia reversa pode também gerar problemas de legalidade, como uma empresa querendo   
criar uma cópia de um produto que vende bem. No entanto a questão legal depende das leis de cada   
país. E, mesmo assim, ainda existem países que não possuem leis específicas sobre o assunto.   
Uma das leis mais conhecidas é o “Digital Millenium Copyright Act” dos Estados Unidos,   
aprovado em 1998 que, entre várias medidas para proteger direitos autorais na informática, também   
faz restrições em relação à engenharia reversa. Só é permitida para fins de analisar compatibilidade   
com outros softwares e/ou hardware.  
   
Na União Européia, o “EU Copyright Directive”, de 2001, é similar ao “Digital Millenium   
Copyright Acts”, porém não é tão restritiva. Só são feitas restrições caso o objetivo final da   
engenharia reversa seja a cópia de algum programa ou quebra de patente com objetivo de lucro.   
Caso seja pra fins acadêmicos ou de compatibilidade, à princípio não existem restrições.  
Na suíça, a lei a respeito do assunto é bastante curiosa e, de certo modo, polemica. A Lei Suíça de   
Concorrência Desleal de 1986 exige dos competidores a realização de investimentos em engenharia   
reversa mesmo quando a tecnologia não seja secreta. Os tribunais suíços, porém, têm rejeitado ou   
limitado severamente a aplicação de tal norma, pela inexistência de prazo e limites.  
   
No Japão, a Lei Japonesa de Concorrência Desleal de 1993 proíbe a imitação servil, mesmo no caso   
de produtos não patenteados, nem protegidos por direitos autorais. A lei japonesa impõe limites   
claros à aplicação da norma de apropriação ilícita: o “lead time” vigora apenas por três anos; não se   
protegem as idéias e os conceitos técnicos; e ressalva-se o caso de modificações ou aperfeiçoamento   
técnico efetuado pelo competidor com base no item copiado; a necessidade de padronização e   
compatibilização de produtos e o uso de elementos de caráter estritamente funcional. Ou seja, a   
proibição de imitação não impede o progresso técnico, ressalva o domínio das patentes para   
proteger idéias e conceitos, e o interesse social na padronização e compatibilização industrial.  
   
No Brasil, não existe uma lei específica sobre Engenharia Reversa. Apesar disso, quando ocorre   
engenharia reversa, costuma-se proceder de duas maneiras: caso a engenharia reversa não tenha   
como objetivo a pirataria ou infração de algum direito autoral, não é considerado crime; porém caso   
contrário, a Lei de Software e também de Direitos Autorais protege seus autores.  
   
12

Engenharia Reversa  
Alguns Casos Famosos  
Apesar dessas leis, os problemas legais relacionados à engenharia reversa são bastante comuns e   
difíceis de serem resolvidos.  
   
Na Noruega (um dos países que permitem engenharia reversa) Jon Johansen, um jovem de 16 anos,   
desenvolveu um programa capaz de fazer um computador com Linux exibir um filme em DVD.   
   
O que Jon e alguns colaboradores fizeram foi descobrir que uma das empresas, a Xing   
Technologies, subsidiária da RealNetworks, não mantinha chaves criptografadas. Eles entraram no   
site da empresa, pegaram as chaves e simplesmente olharam como elas funcionavam. É como uma   
espécie de quebra-cabeças, onde o que precisa ser feito é estabelecer correlações entre símbolos. A   
chave criptográfica do DVD tem a extensão de cinco bytes, o que torna a matemática envolvida   
nisso bem simples. Como as outras chaves das empresas concorrentes também são parecidas, ele   
ainda adivinhou algumas chaves diferentes das da Xing. Isso faz com que o Decoded-CSS, nome   
dado ao seu programa, possa rastrear uma chave adequada entre algumas que ele entende, ou seja,   
mesmo que os discos de DVD daqui para frente não possuam a chave Xing, o programa ainda   
funcionará.  
   
Ele foi acusado de "facilitar a pirataria". Segundo os advogados que o acusam, Jon teria criado um   
software que permite aos usuários fazer e distribuir cópias digitais de filmes em DVD. Seu   
programa permitiria que o arquivo que contém o filme fosse copiado por um microcomputador   
caseiro. Jon está sendo processado por autoridades norueguesas do DOEC (Departamento   
Norueguês de Crimes Econômicos) e pela MPAA (Motion Pictures Association of America),   
entidade que representa as sete maiores distribuidoras de filmes nos EUA: Universal City Studios   
lnc, Paramount Pictures Corporation, Metro-Goldwin-Mayer Studios lnc, Tristar Pictures lnc,   
Columbia Pictures lnc, Time Warner Entertainment Co., Disney Enterprises lnc. e Twentieth   
Century Fox Film Co.  
   
O caso Lotus Development Corp. vs. Borland International foi uma disputa judicial entre duas   
empresas produtoras de Software. A Lotus produzia o Lotus 1-2-3, e a Borland, o Quattro Pro. A   
Borland produziu seu programa de computador com a interface idêntica ao da Lotus, de maneira   
que os usuários da Lotus 1-2-3 pudessem usar o Quattro Pro sem dificuldades. A Lotus entrou com   
ação em face da Borland por infração de Copyright. Grande parte da controvérsia foi a respeito da   
possibilidade de se proteger pelo direto do autor a mera interface do programa. Por fim a decisão   
final achou "absurdo" sugerir que "se alguém faz uso de vários programas diferentes, seja forçado a   
aprender como efetuar cada a mesma operação de maneiras diferentes em cada programa utilizado".   
A corte decidiu que, se uma empresa atinge um monopólio, por conseqüência a maioria do mercado   
fica bem adaptada à interface de seu programa. Desse modo, é justo que um competidor utilize a   
mesma interface como meio de concorrência.   
13

Engenharia Reversa  
5. Bibliografia  
Nicolas Anquetil Homepage  
http://www.ucb.br/ucbtic/mgcti/paginapessoalprof/Nicolas/Disciplinas/RevEng/index.html  
Wikipedia  
www.wikipedia.org  
DVD e Linux: liberdade na era digital  
http://www.comciencia.br/reportagens/softliv/softliv1.html  
Limitações ao direito do autor na lei brasileira, cópia privada e engenharia reversa de   
software  
http://www.direitonet.com.br/artigos/x/19/08/1908/  
Implementing the EU Copyright Directive  
www.fipr.org/copyright/guide/eucd-guide.pdf  
The Digital Milllenium Copyright Act of 1998  
www.copyright.gov/legislation/dmca.pdf  
14