UFSC
Araranguá

Reengenharia de Software

Engenharia de Software II

Profa. Andréa Sabedra Bordin

Engenharia Direta x Reengenharia

Fonte: Sommerville (2007, p. 332)

Introdução

- Reengenharia de Software é uma forma de **modernização** que melhora as capacidades e/ou manutenibilidade de um sistema legado através da utilização de tecnologias e práticas modernas.
- Reengenharia de Software oferece uma abordagem disciplinada para migrar um **sistema legado** para sistema que possa ser evoluído.
- Reengenharia é uma transformação sistemática de um sistema existente para uma nova forma, procurando realizar melhorias de qualidade, capacidade, funcionalidade, performance ou *evolvability* a um custo baixo, cronograma curto e risco baixo para o cliente. *Software Engineering Institute (SEI*).

Por que reengenhar ?

- Reengenharia é mais cara que uma simples manutenção.
- Menos atrativa - de uma perspectiva técnica - do que substituir um sistema legado.
  - É geralmente a mais pragmática.
  - Como existem muitos sistemas legados, substituir completamente é inconcebível do ponto de vista financeiro.

Vantagens

- As vantagens da reengenharia sobre a substituição (desenvolvimento de novo código) são:
  - Redução de riscos: erros de especificação, problemas de desenvolvimento, etc.
  - Redução de custos: custos de reengenharia são inferiores ao um novo desenvolvimento.
    - Os custos dependem da extensão (nível) do trabalho de reengenharia (Sommerville, 2009)

Formas de Reengenharia

- *Retargeting*
  - *Retargeting* é a migração de um sistema legado para uma nova plataforma de hardware.
  - Migrar para uma plataforma de hardware moderna frequentemente reduz custos operacionais e de manutenção, permite a introdução de computadores de alta performance e provê uma plataforma evoluível para outros esforços de manutenção.

Formas de Reengenharia

Revamping

A interface do usuário (IU) é a parte mais visível do sistema. Substituir somente a IU é uma forma de reengenharia de software conhecida como *revamping.*

Reformar uma IU deve melhorar a usabilidade e é sempre notada pelos usuários finais do sistema.

Formas de Reengenharia

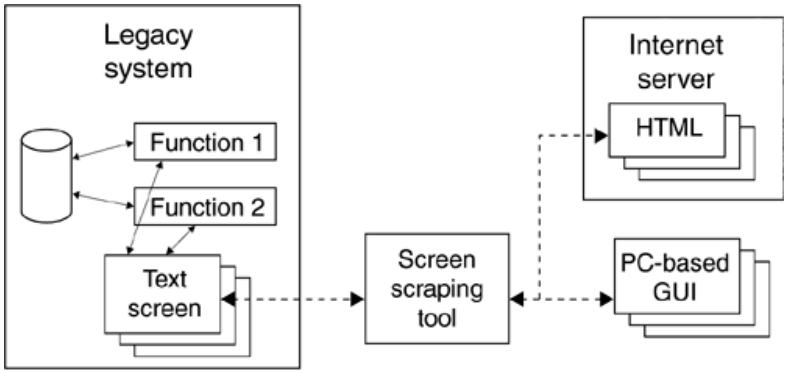
- *Revamping*

– Uma técnica *black-box* para *revamping* é ***screen scraping***.

– Consiste em envolver interfaces antigas, baseadas em texto, em interfaces gráficas novas.

  – A interface antiga tipicamente é um conjunto de telas textuais rodando em um terminal.

  – A nova interface pode ser uma GUI que roda em um PC ou um HTML rodando em um browser.



Formas de Reengenharia

- Uso de componentes comerciais
  - Projetos de reegenharia de software podem substituir código legado com componentes comerciais (*commercial-off-the-shelf -COTS*).
  - Normalmente reduz a quantidade de código-fonte que deve ser mantido.
  - É importante considerar não somente a economia de código a ser mantido, mas também os custos de extrair código legado e a quantidade de “code glue” que deve ser desenvolvido e mantido, e também os custos de licenças e treinamento.

Formas de Reengenharia

- Tradução de código-fonte
  - Consiste em converter código de uma linguagem de programação antiga para uma linguagem mais moderna.
  - Esse tipo de reegenharia tem um alto risco porque frequentemente os benefícios são superestimados e os custos são subestimados.
  - Pode requerer **ferramentas de tradução automática**.
  - Ferramentas de tradução automática podem oferecer produtividade, mas elas não podem mudar significativamente a estrutura do código.
- *"For example, automatically translating COBOL to Java often results in code that looks like COBOL written in the Java programming language. For this and other reasons, transformed code is often difficult to maintain: in this case, for either a COBOL or a Java programmer."*

Formas de Reengenharia

- Transformação funcional
  - Inclui melhoria na estrutura do programa, modularização do programa e reengenharia de dados.
  - A modularização do programa agrupa partes relacionadas do programa em módulos.
    - Modularização do programa é um passo preliminar em um esforço de modernização incremental porque módulos bem definidos são mais fáceis de substituir por novos componentes.
  - A **reengenharia de dados** envolve modificar a armazenagem, organização e formato dos dados processados pelo sistema legado.
    - Ela é necessária porque os dados podem ter sido guardados inconsistemente em múltiplos locais.

Análise da relação custo x benefício

Business Case (Caso de Negócio)

O que é?

- Um caso de negócio pode determinar se um projeto é rentável.
- Um caso de negócio geralmente é realizado no início do projeto, normalmente após a **análise de portfólio** ter priorizado os sistemas que devem ser considerados para a **modernização**.
  - Muitas vezes o número de sistemas legados direcionados para a modernização é maior do que o volume dos recursos disponíveis.
- O *business case* deve apresentar uma análise custo – benefício, vantagem financeira ou uma vantagem competitiva para justificar o **esforço de modernização** e **riscos de implementação** associados.

O que é?

- Mesmo se os recursos são suficientes para modernizar um sistema legado, isso não significa que ele deve ser feito.
- Mais uma vez, um caso de negócio deve ser desenvolvido.
  - Justificar um esforço de modernização é mais difícil do que justificar um novo esforço de desenvolvimento de aplicação, já que a modernização é muitas vezes vista como um último recurso.

Processo do Caso de Negócio

1. Identificação das partes interessadas;
2. Entendimento dos requisitos do sistema;
3. Criação de um caso de negócio;
4. Avaliação do caso pela administração;
    1. A gestão poderá decidir:
        1. iniciar o esforço de modernização,
        2. solicitar que o caso de negócio seja atualizado ou
        3. repetir o processo para um novo sistema candidato a modernização.

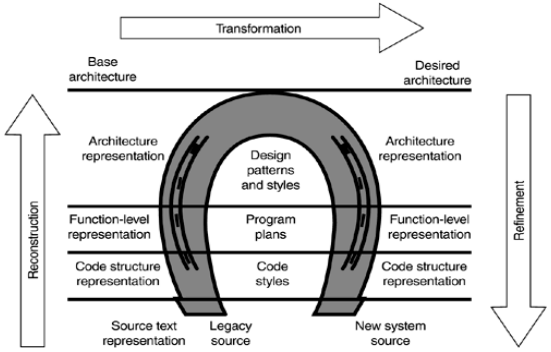
Modelo Ferradura (*Horseshoe Model*)

Um esforço de modernização de um sistema consiste em **três processos** básicos:

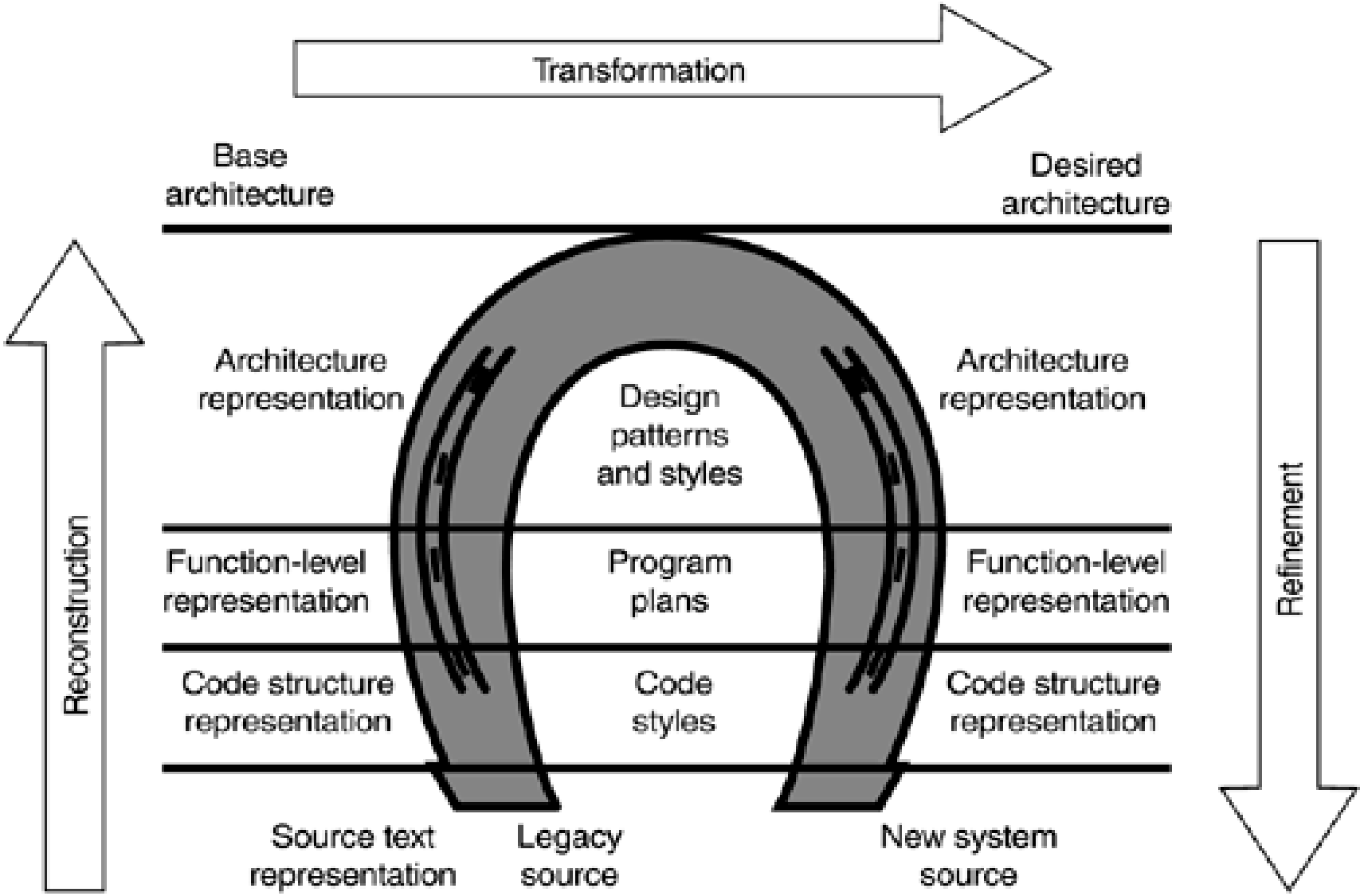
1.Reconstruir uma ou mais descrições lógicas do sistema em alto nível, a partir dos artefatos existentes (análise do sistema existente).

2.Transformar as descrições lógicas em novas e melhoradas descrições lógicas .

3.Refinar estas novas e melhoradas descrições lógicas para código (desenvolvimento de um novo sistema).



O grau com que cada processo é aplicado depende dos objetivos da modernização.


Transformation
Base architecture
Desired architecture
Reconstruction
Refinement
Architecture representation
Design patterns and styles
Architecture representation
Function-level representation
Program plans
Function-level representation
Code structure representation
Code styles
Code structure representation
Source text representation
Legacy source
New system source

Modelo Ferradura

- O modelo também propõe três níveis de abstração que podem ser usados para descrições lógicas, as quais podem ser concretas e simples como o código-fonte do sistema ou abstratas e complexas como a arquitetura do sistema.

1. **Transformações de código:**
   - São formas de evolução "rápidas e sujas". Inclui técnicas de *retargeting* ou *source code translation*.
   - São mais associadas com atividades de manutenção do que com atividades de reengenharia.

     "*Code transformation does little to improve the structure of the legacy code and often results in code that is even more difficult to maintain*."

Modelo Ferradura

2. **Transformações funcionais:**
   - São requeridas, por exemplo:
     - na mudança de um paradigma funcional para um paradigma orientado a objeto ;
     - na mudança de um processamento *off-line* para um processimento do tipo *e-commerce*;
     - na mudança de um database hierárquico para um database relacional.

Modelo Ferradura

3. Transformações arquiteturais

- Neste caso, o processo de reconstrução resulta na recuperação completa da arquitetura.
- No processo de transformação, a arquitetura recuperada é “reengenhada”.

Modelo Ferradura

1) Processo de Reconstrução

- Está fortemente relacionado com o entendimento do programa (*program understanding*).
  - *Program understanding* permite que o desenvolvedor forme/crie descrições abstratas do sistema.

"**Reverse engineering** is the process of analyzing a subject system to create representations of the system at a higher level of abstraction."

Chikofsky, E. J.; Cross, J. H. (January 1990). "Reverse engineering and design recovery: A taxonomy". *IEEE Software* **7**: 13–17.

Modelo Ferradura

1) **Processo de Reconstrução:** Técnicas nos diversos níveis de abstração:

   a) Representação da estrutura do código

   - **Leitura do código manualmente:** o engenheiro de software lê o código-fonte.
   - **Análise estática:** envolve “parsear” o código-fonte para gerar vários relatórios, incluindo grafo de chamadas, fluxos de controle e de dados e análise da definição/uso de tipos de dados e variáveis de instância.

Modelo Ferradura

b) Representação do nível de função : descreve os relacionamentos entre funções (chamadas) e dados (relacionamentos de dados e funções).

- **Redocumentação** é uma das formas mais antigas de engenharia reversa. É o processo de prover documentação para um software existente.
- **Refatoração**: é o processo de modificar um sistema de forma que o seu comportamento externo não seja alterado, mas a sua estrutura interna seja melhorada.

Processo de Reconstrução
Ferramentas

- Libera os engenheiros de software de tarefas tediosas, manuais e sujeitas a erros, como leitura de código, pesquisa e correspondência de padrões por inspeção.
- Muitas **ferramentas** de **engenharia reversa** se concentram em extrair a estrutura de um sistema legado com o objetivo de transferir essas informações para a mente dos engenheiros de software que estão tentando fazer a reengenharia.
- As ferramentas de engenharia reversa ainda têm um longo caminho a percorrer antes de se tornarem parte integrante e eficaz do conjunto de ferramentas padrão que um típico engenheiro de software usa no dia-a-dia.
  - A maioria dos engenheiros de software tem pouco conhecimento das ferramentas atuais e de seus recursos.

Processo de Reconstrução

Ferramentas

- Plugins Eclipse (Java)

http://marketplace.eclipse.org/category/free-tagging/reverse-engineering

- Understand (C/C++, Java, C#, Fortran, Cobol, Delphi/Pascal)

http://www.scitools.com/

- Imagix4D (C, C++ e Java)

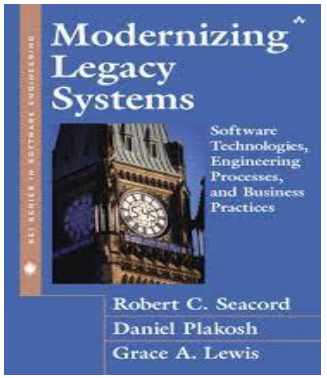
http://www.imagix.com/index.html

Modelo Ferradura

2) **Processo de transformação**: a arquitetura recuperada é reengenhada.

3) **Processo de refinamento**: transforma a arquitetura em projeto e refina o projeto em código-fonte.

Referências



- SEACORD, Robert; PLAKOSH, Daniel; LEWIS, Grace **Modernizing Legacy Systems:** Software Technologies, Engineering Processes, and Business Practices. 1 ed. Addison-Wesley Professional, 2003.
- MULLER, H. A. Reverse Engineering - A roadmap. Proceedings of the Conference on The Future of Software Engineering. 2000.
- CANFORA, Gerard; DI PENTA, Massimiliano; CERULO. Luigi. Achievements and Challenges in Software Reverse Engineering. Communications of the ACM. 2011.