Engenharia de Software

Seção 9 – Engenharia Reversa e Reengenharia

Objetivos


O aluno deverá reconhecer os conceitos e aplicações e Engenharia Reversa e Reengenharia de Software

Era uma vez...

- ✓ Suponha um software que tenha servido a uma empresa por 10, 15 anos.
- ✓ Durante este período ele foi corrigido, adaptado e melhorado diversas vezes,
- ✓ Apesar das “boas intenções” dos mantenedores, boas práticas de Engenharia de Software foram deixadas de lado durante as modificações.
- ✓ Agora a aplicação está instável. Ainda funciona, mas qualquer tentativa de alteração leva a vários efeitos colaterais indesejáveis.
- ✓ Ainda assim, o software precisa evoluir.
- ✓ O que fazer?

Manutenção de Software

- ✓ Software Não-manutenível não é um problema novo.
- ✓ Já há quase 40 anos a manutenção de software era vista como um “iceberg”.
- ✓ Espera-se que o que está imediatamente visível seja tudo, mas sabemos que uma enorme massa de problemas e custos potenciais esconde-se sob a superfície.

Imagem: by Uwe Kills via Wikimedia Commons

Manutenção de Software (continuação)

O custo da manutenção de software pode chegar a 60% de todo o esforço despendido por uma organização de desenvolvimento de software.

Algumas das causas para isso:

- ✓ Alguns dos programas legados foram criados quando havia muitas restrições de memória e disco, e malabarismos eram feitos para fazer o programa “caber” no espaço que lhe era destinado.

Manutenção de Software (continuação)

- ✓ Pessoas deixam projetos e empresas, e muitas vezes o conhecimento do software está só com elas.

- ✓ Às vezes é difícil achar quem programa em determinada linguagem ou plataforma (há 200 bilhões de LOCs em COBOL ATIVAS no mundo – alguém se habilita?)

Manutenção de Software (continuação)

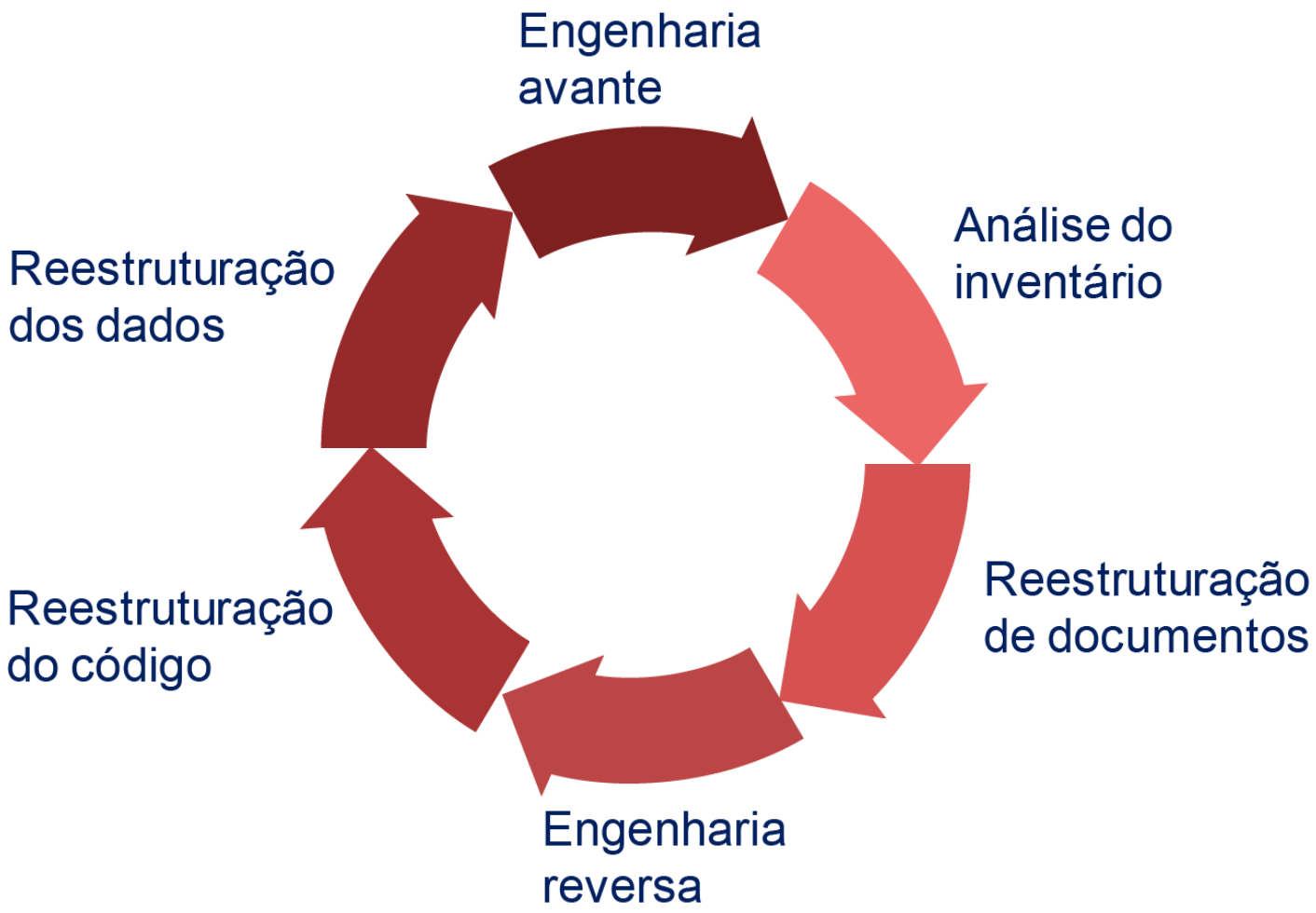
- A manutenção de software é dividida em:
  - Manutenção corretiva: Correção de defeitos de software, responsável por 20% do esforço de manutenção.
  - Manutenção adaptativa: Responsável pela adaptação do software a modificações no ambiente.
  - Melhoria: Aperfeiçoamento do software com foco em usabilidade ou desempenho.
  - Reengenharia: Reconstrução da aplicação para uso futuro.

Reengenharia de Software

- ✓ Realizar a reengenharia (ou “reengenhariar”) de um software significa reconstruir este software.
- ✓ Esta atividade é usualmente custosa e longa.
- ✓ Para que ela seja realizada adequadamente, uma série de passos deve ser seguida, acompanhando uma estratégia bem definida.
- ✓ Uma estratégia viável é incluída em um modelo de processo de reengenharia.

Reengenharia de Software

(continuação)



Análise do Inventário

- A organização precisa saber quais são os itens de software que precisam de reengenharia.
- Para tal, o mínimo que se precisa é uma tabela, contendo, para cada software da empresa:
  - Tamanho
  - Idade
  - Criticalidade para o negócio
  - Manutenibilidade
- Avaliando-se esta planilha, é possível identificar quais são os principais candidatos a reengenharia.

Reestruturação de Documentos

Documentação fraca é a marca registrada de um sistema legado. As alternativas para lidar com isso são:

- ✓ Documentar leva tempo demais: Se o sistema está no final de sua vida útil, não sofrerá modificações até lá e está funcionando, não faça nada.
- ✓ A documentação precisa ser atualizada, mas temos poucos recursos: Documentar apenas o que for tocado.
- ✓ O software é crítico para o negócio e precisa ser todo redocumentado: Gerar apenas documentação essencial.

Engenharia Reversa

- ✓ Como parte do processo de Reengenharia, a Engenharia Reversa atua no sentido de obter informações sobre as aplicações da própria organização (criadas às vezes muitos anos antes).
- ✓ A Engenharia Reversa é um processo de recuperação do design do software.
- ✓ Ferramentas de Engenharia Reversa extraem dados, design arquitetural e procedural de um programa existente.

Reestruturação de Código

- Às vezes se conhece a arquitetura do software (eventualmente obtida a partir de Engenharia Reversa) e esta estrutura é bastante sólida, mas o código dos módulos que são endereçados por ela não é.

- Os módulos individuais são pouco inteligíveis, difíceis de testar e manter.

Reestruturação de Código

(continuação)

- ✓ O código é então reestruturado (eventualmente de forma automática, com ferramentas específicas), revisado e testado para assegurar que nenhuma anomalia foi introduzida.

- ✓ A documentação do módulo é atualizada de acordo com a nova estrutura.

Reestruturação de Dados

- ✓ Programas com uma arquitetura de dados fraca apresentarão dificuldades para melhorar e adaptar. A arquitetura de dados pode ter mais a ver com a viabilidade de longo prazo do software do que o próprio código.

- ✓ A reestruturação do código ocorre a um nível relativamente baixo de abstração, mas a reestruturação de dados não, pois usualmente é necessário obter modelos de dados de alto nível.

Reestruturação de Dados

(continuação)

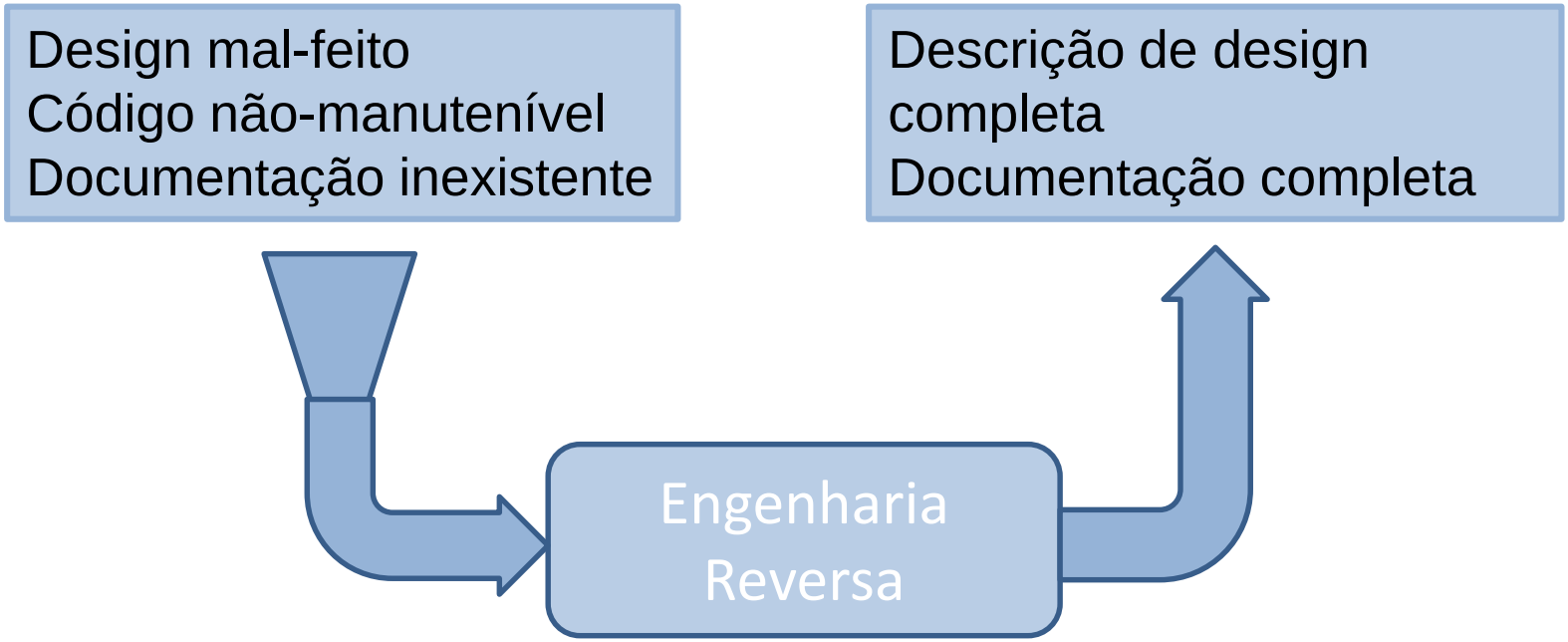
- Uma vez que a arquitetura de dados tem uma forte influência sobre a arquitetura do programa e dos módulos que a populam, usualmente modificações nesta arquitetura levam a alterações arquiteturais ou mudanças de código.

Engenharia avante

- Engenharia avante é o termo para identificar o processo de engenharia de software aplicado aos artefatos obtidos a partir das análises e avaliações do software a ser modificado.
- De posse de informações originais de design, os analistas tem condições de recriar o software, aplicando melhores práticas de arquitetura, design e qualidade, e utilizando técnicas mais avançadas.
- O design original serve, neste caso, como especificação de nível detalhado.

Engenharia Reversa

- ✓ O termo “Engenharia Reversa” traz à mente algo como:



Engenharia Reversa (continuação)

- Engenharia Reversa pode extrair informações de design a partir de código-fonte, mas:
  - O nível de abstração
  - A completeza da informação
  - O nível em que o analista e ferramentas devem trabalhar juntos, e
  - A direcionalidade do processo
- São altamente variáveis

Nível de Abstração

- O Nível de Abstração refere-se à sofisticação da informação de design que pode ser obtida do código-fonte.
- Em níveis crescentes de abstração, teríamos:
  - Representações de design procedural;
  - Informação de estruturas de dados e programas;
  - Modelos de objetos
  - Classes (em UML)
  - Diagramas de estado e distribuição

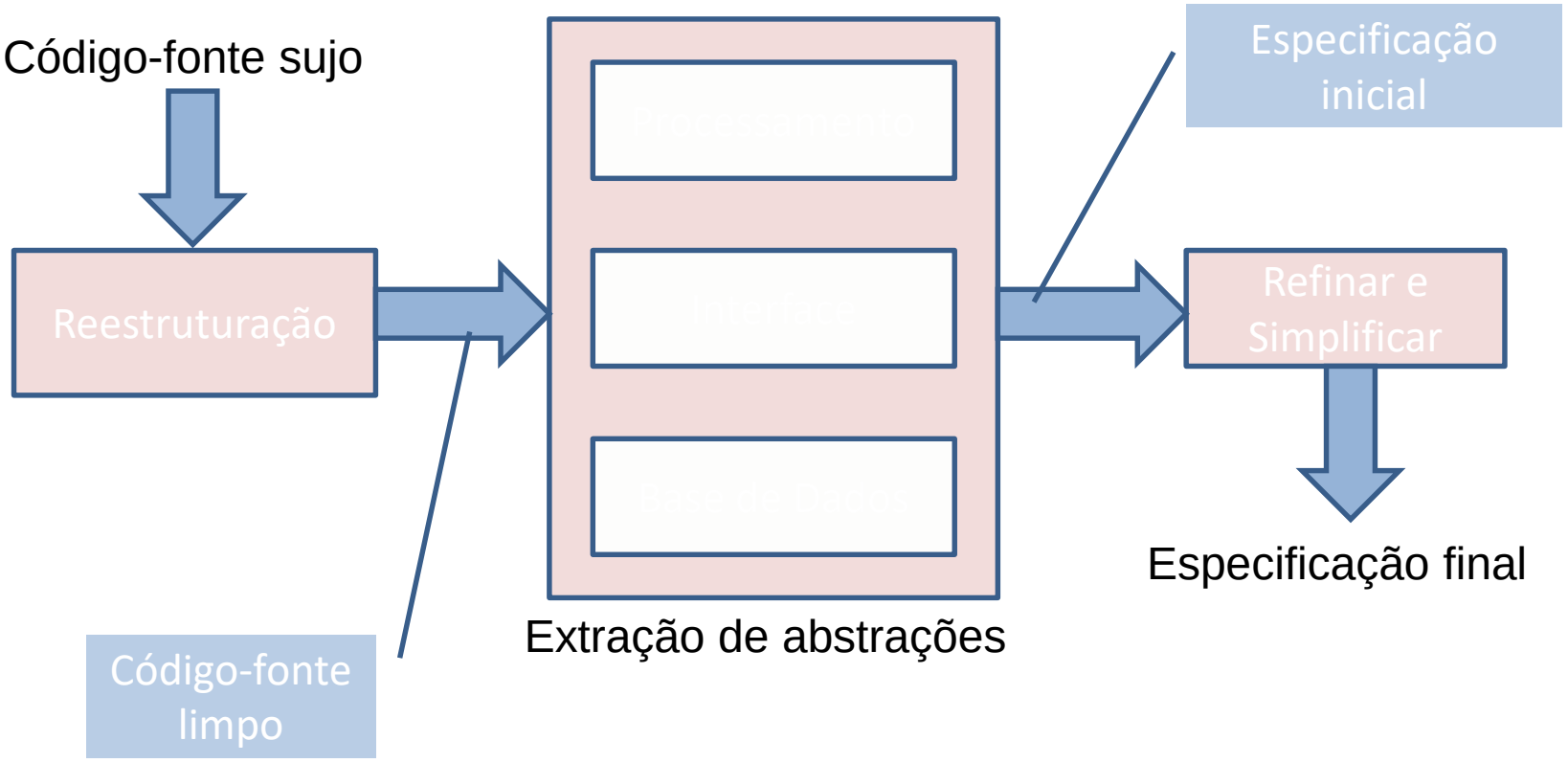
Completeza

- A completeza de documentação refere-se ao nível de detalhe que é fornecido a cada nível de abstração.
- Por exemplo, dado um conjunto de arquivos de código-fonte, é relativamente fácil obter o design procedural, mas é mais complicado obter um conjunto completo de diagramas UML.
- A completeza aumenta na proporção direta da quantidade de análise realizada pela pessoa realizando a engenharia reversa.

Interatividade e Direcionalidade

- ✓ A interatividade diz respeito ao grau em que um humano e integrado com as ferramentas automatizadas para criar um processo de engenharia reversa efetivo.
- ✓ A direcionalidade refere-se à possibilidade de se realizar a engenharia reversa unidirecional, isto é, apenas obtendo a informação de design, ou bidirecional, em que esta informação pode ser utilizada para gerar automaticamente o código da qual foi originada.

Processo de Engenharia Reversa



Fonte: PRESSMAN, Roger S. Software Engineering: A practicioner's Approach. 6th. Ed. McGraw-Hill