Capítulo 9

Evolução de Software

Tópicos abordados

- Processos de evolução
  - ✓ Processos de mudança de sistemas de software
- Dinâmica da evolução de programas
  - ✓ Compreensão da evolução de softwares
- Manutenção de software
  - ✓ Mudanças nos sistemas de software em operação.
- Gerenciamento de sistemas legados
  - ✓ Tomadas de decisão sobre mudanças no software

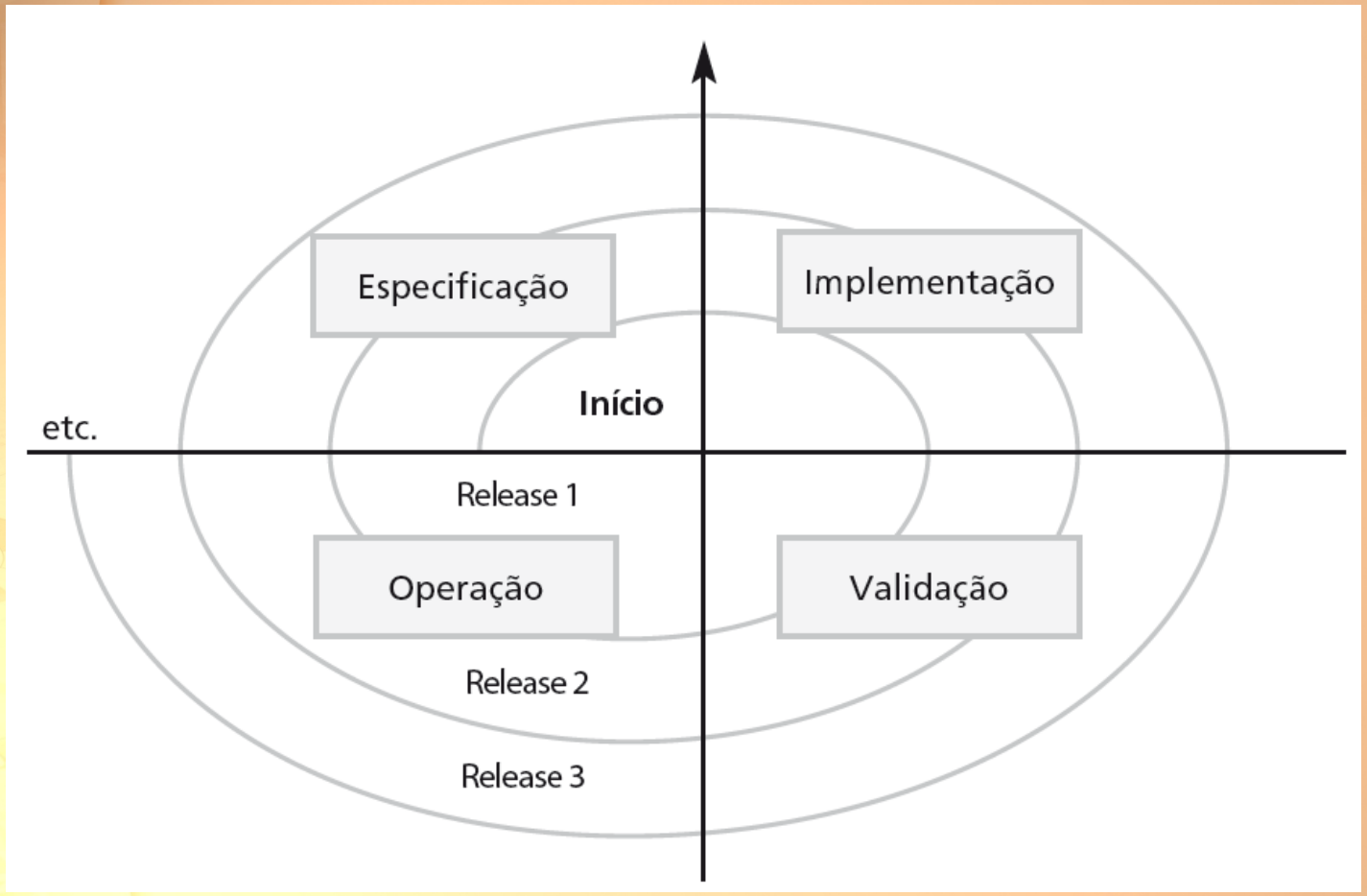
Mudanças no software

- Mudanças no software são inevitáveis
  - ✓ Novos requisitos emergem quando o software é usado;
  - ✓ Mudanças no ambiente de negócios;
  - ✓ Erros devem ser reparados;
  - ✓ Computadores e equipamentos novos são adicionados ao sistema;
  - ✓ O desempenho ou a confiabilidade do sistema pode demandar melhorias.
- Um problema-chave de todas as organizações está em implementar e gerenciar as mudanças em seus sistemas de software já existentes.

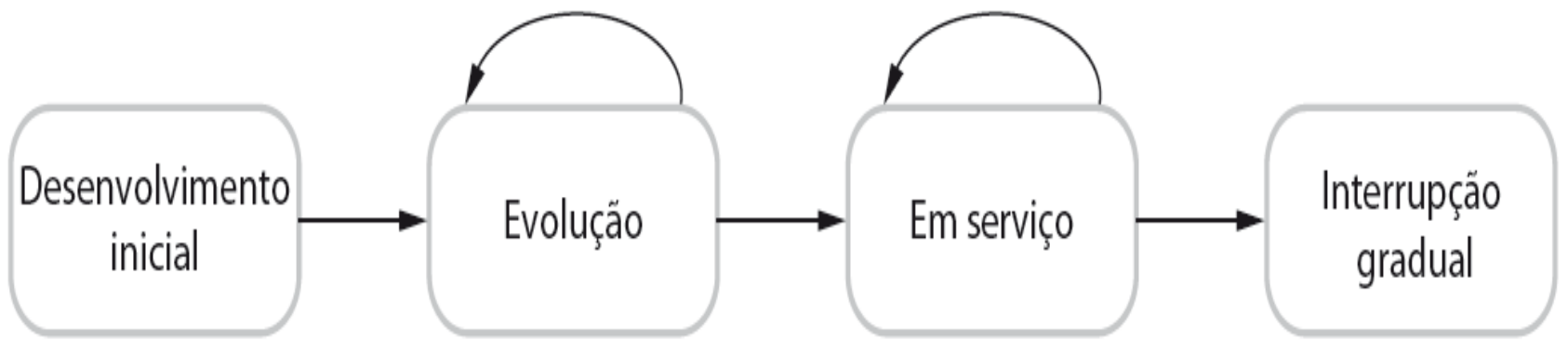
Importância da evolução

- As organizações fazem grandes investimentos em seus sistemas de software – esses são ativos críticos de negócios.
- Para manter o valor desses ativos para o negócio, esses devem ser alterados e atualizados.
- A maior parte do orçamento de softwares em grandes empresas é dedicado à mudança e evolução de softwares existentes e não no desenvolvimento de softwares novos.

Um modelo espiral de desenvolvimento e evolução



Evolução e em serviço



Evolução e em serviço

- Evolução
  - ✓ Fase do ciclo de vida de um sistema de software em que esse está em uso operacional e está evoluindo na medida em que novos requisitos são propostos e implementados no sistema.

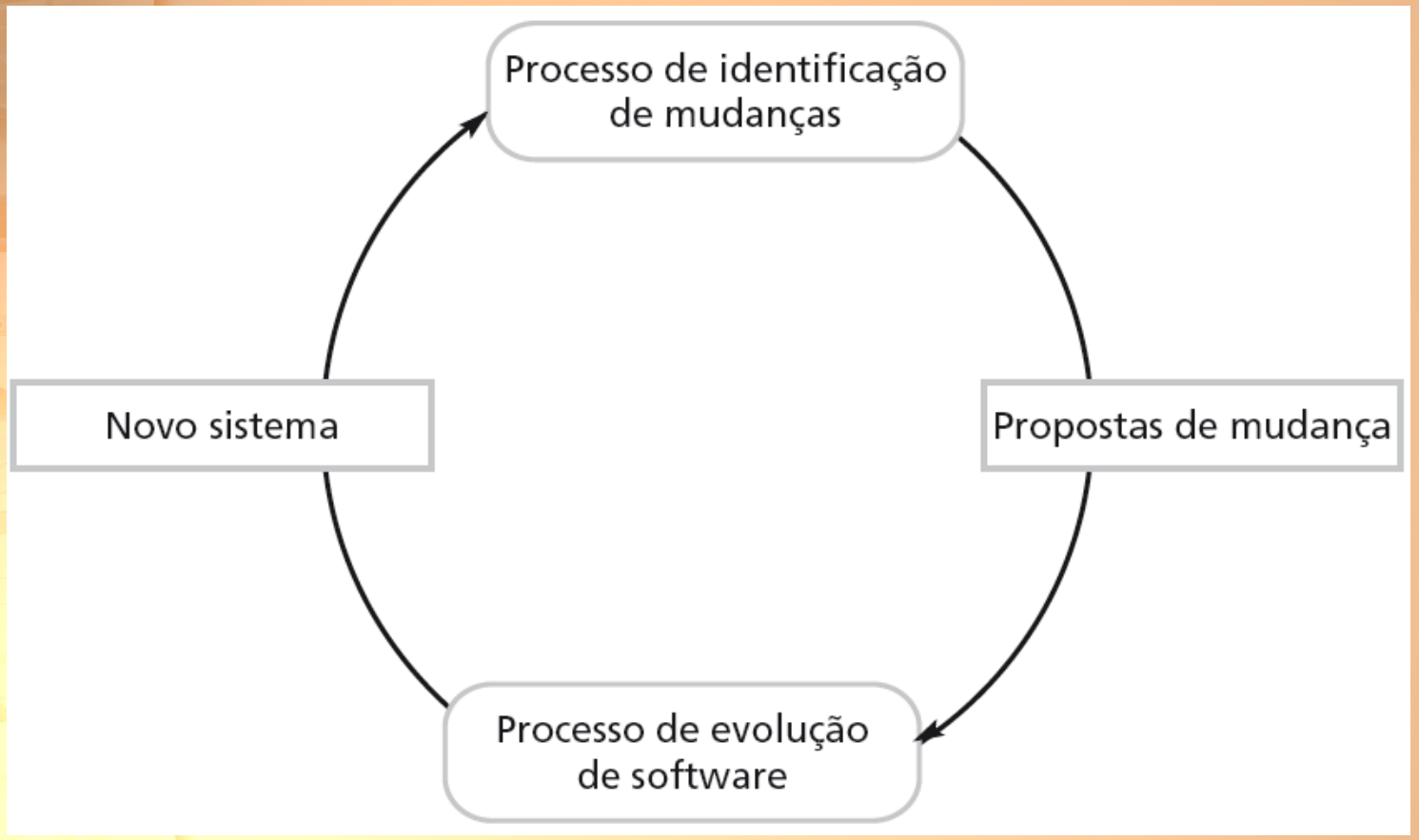
- Em serviço
  - ✓ Nesta fase, o software continua a ser útil, mas as únicas mudanças feitas são aquelas necessárias para mantê-lo operacional, isso é, correções de bugs e mudanças para refletir as mudanças no ambiente do software. Nenhuma funcionalidade nova é adicionada.

- Interrupção gradual
  - ✓ O software ainda pode ser usado, mas nenhuma outra alteração é realizada nele.

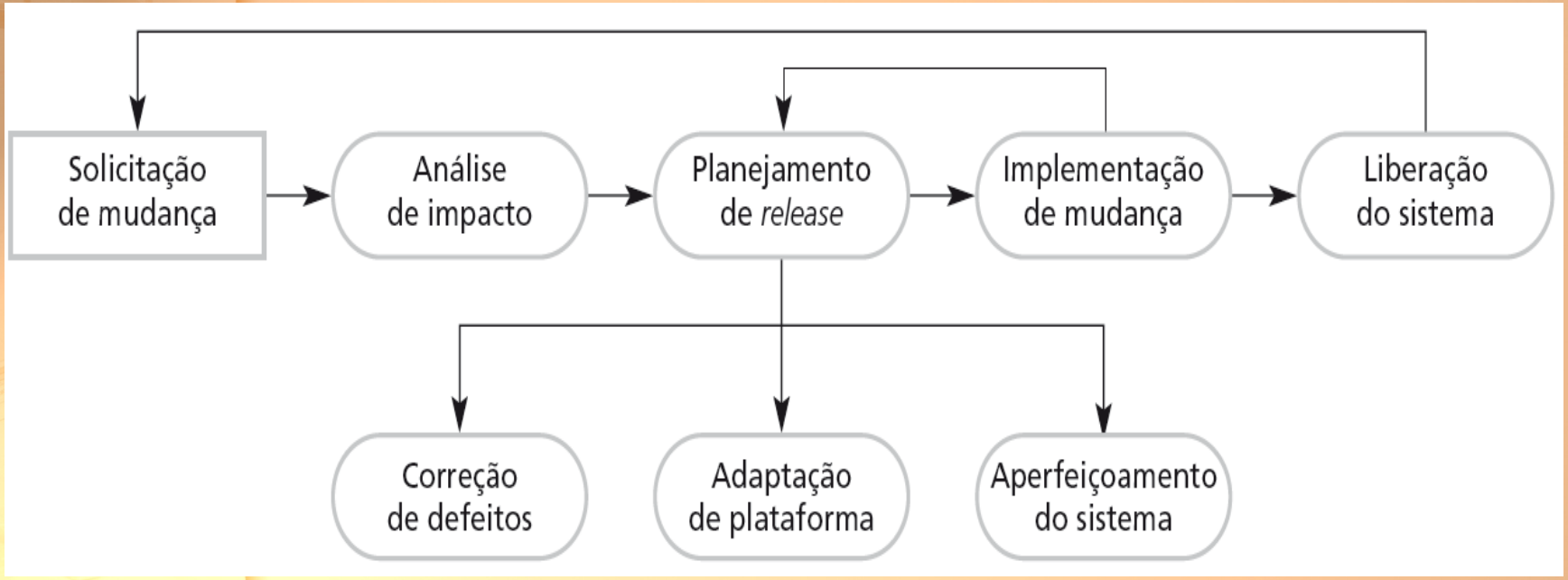
Processos de evolução

- Processos de evolução do software dependem de:
  - ✓ O tipo de software que está sendo mantida;
  - ✓ O desenvolvimento dos processos usados;
  - ✓ As habilidades e a experiência das pessoas envolvidas.
- Propostas de mudança são os acionadores para a evolução do sistema.
  - ✓ Devem estar ligados com componentes que são afetados pela mudança, permitindo assim que sejam estimados os custos e o impacto da mudança.
- A evolução e a identificação de mudanças continuam durante toda a vida do sistema.

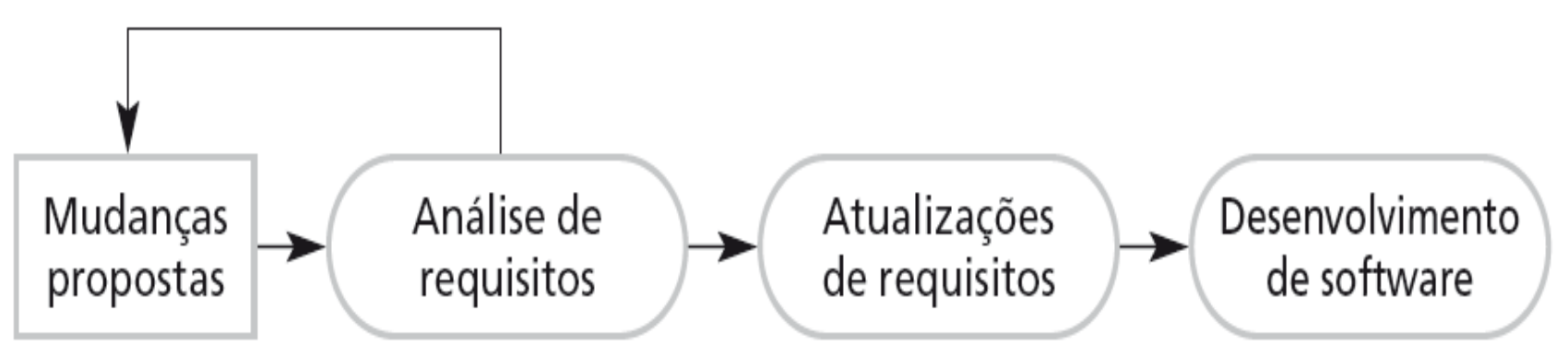
Processos de identificação de mudanças e de evolução



O processo de evolução de software



Implementação de mudança



Implementação de mudança

- Iteração do processo de desenvolvimento, no qual as revisões do sistema são projetadas, implementadas e testadas.

- Uma diferença fundamental é que a primeira fase de implementação da mudança pode envolver a compreensão do programa, especialmente se os desenvolvedores do sistema original não são responsáveis pela implementação da mudança.

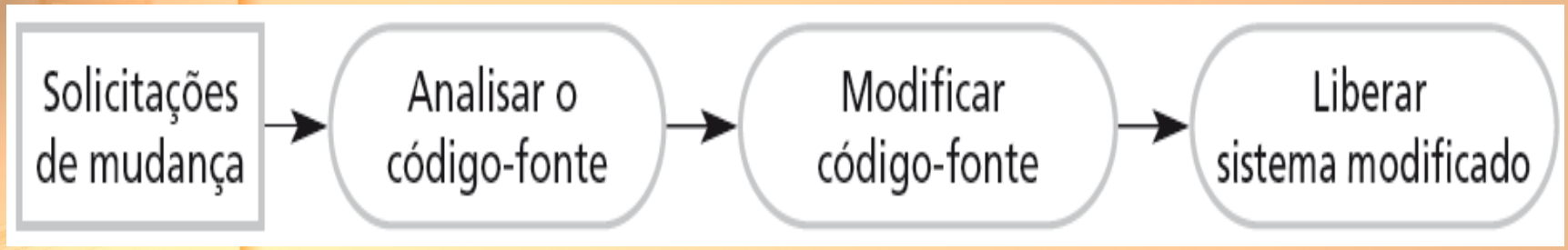
- Durante a fase de compreensão do programa, você precisa entender como o programa está estruturado, como ele implementa a funcionalidade e como a mudança proposta pode afetar o programa.

Solicitações de mudança

- Mudanças urgentes podem precisar ser implementadas sem passar por todas as fases do processo de engenharia de software.

  - ✓ Se um defeito grave do sistema precisa ser reparado para permitir a continuidade da operação normal;
  - ✓ Se as mudanças ao ambiente do sistema (por exemplo, uma atualização do sistema operacional) tem efeitos inesperados;
  - ✓ Se houver mudanças de negócios que exigem uma resposta muito rápida (por exemplo, o surgimento de um produto concorrente).

O processo de correção de emergência



Métodos ágeis e evolução

- Métodos ágeis são baseados no desenvolvimento incremental, assim, a transição do desenvolvimento para a evolução é imperceptível.
  - ✓ A evolução é simplesmente uma continuação do processo de desenvolvimento baseada em versões frequentes do sistema.
- Testes de regressão automatizados são particularmente valiosos quando são feitas mudanças em um sistema.
- Mudanças podem ser expressas como estórias adicionais de usuários.

Problemas de transferência

- No qual a equipe de desenvolvimento usa uma abordagem ágil, mas a equipe de evolução não está familiarizado com os métodos ágeis e preferem uma abordagem baseada em planos.

  - ✓ A equipe de evolução pode esperar uma documentação detalhada para apoiar a evolução, e essa não é produzida em processos ágeis.

- Em que uma abordagem baseada em planos tem sido usada para o desenvolvimento, mas a equipe de evolução prefere usar métodos ágeis.

  - ✓ A equipe de evolução pode ter de começar do zero, desenvolvendo testes automatizados e os códigos no sistema podem não ter sido refatorados e simplificados como é esperado em desenvolvimento ágil.

Dinâmica da evolução de programas

- Dinâmica da evolução de programas é o estudo dos processos de mudanças de sistema.

- Depois de vários importantes estudos empíricos, Lehman e Belady propuseram que houvesse uma série de "leis" que se aplicassem a todos os sistemas, na medida em que eles evoluíssem.

- Existem observações ao invés de leis. Elas são aplicáveis a sistemas de grande porte desenvolvidos por grandes organizações.

  - ✓ Não está claro se essas são aplicáveis a outros tipos de sistemas de software.

A mudança é inevitável

- Os requisitos do sistema são suscetíveis de mudar enquanto o sistema está sendo desenvolvido, pois o ambiente está mudando.
- Portanto, um sistema entregue não atenderá a seus requisitos!
- Sistemas são fortemente acoplados com seu ambiente.
- Quando um sistema é instalado em um ambiente, ele muda esse ambiente e, portanto, altera os requisitos do sistema.
- Os sistemas devem mudar, caso queiram permanecer úteis em um ambiente.

Leis de Lehman

Lei	Descrição
Mudança contínua	Um programa usado em um ambiente do mundo real deve necessariamente mudar, ou se torna progressivamente menos útil nesse ambiente.
Aumento da complexidade	Como um programa em evolução muda, sua estrutura tende a tornar-se mais complexa. Recursos extras devem ser dedicados a preservar e simplicar a estrutura.
Evolução de programa de grande porte	A evolução de programa é um processo de autorregulação. Atributos de sistema como tamanho, tempo entre *releases* e número de erros relatados são aproximadamente invariáveis para cada *release* do sistema.
Estabilidade organizacional	Ao longo da vida de um programa, sua taxa de desenvolvimento é aproximadamente constante e independente dos recursos destinados ao desenvolvimento do sistema.

Leis de Lehman

Lei	Descrição
Conservação da familiaridade	Durante a vigência de um sistema, a mudança incremental em cada *release* é aproximadamente constante.
Crescimento contínuo	A funcionalidade oferecida pelos sistemas tem de aumentar continuamente para manter a satisfação do usuário.
Declínio de qualidade	A qualidade dos sistemas cairá, a menos que eles sejam modificados para refletir mudanças em seu ambiente operacional.
Sistema de *feedback*	Os processos de evolução incorporam sistemas de *feedback* multiagentes, *multiloop*, e você deve tratá-los como sistemas de *feedback* para alcançar significativa melhoria do produto.

Aplicabilidade das leis de Lehman

- Geralmente, as leis de Lehman parecem ser aplicáveis a sistemas customizados, grandes, desenvolvidos por grandes organizações.
  - ✓ Confirmado no início de 2000 pelo trabalho de Lehman sobre o projeto FEAST.
- Não está claro como devem ser modificadas para:
  - ✓ **Produtos de software devidamente embalados (rever tradução ???);**
  - ✓ Sistemas que incorporam um número significativo de componentes COTS;
  - ✓ Pequenas organizações;
  - ✓ Sistemas de médio porte.

Pontos importantes

- O desenvolvimento e a evolução de software podem ser pensados como processo iterativo e integrado, que pode ser representado usando um modelo em espiral.
- Para sistemas customizados, os custos de manutenção de software geralmente excedem os custos de desenvolvimento de software.
- O processo de evolução do software é impulsionado pelas solicitações de mudança e inclui a análise do impacto da mudança, planejamento de release e implementação da mudança.
- As leis de Lehman, assim como a noção de que a mudança é contínua, descreve uma série de considerações derivadas de estudos de longo prazo, de evolução de sistema.

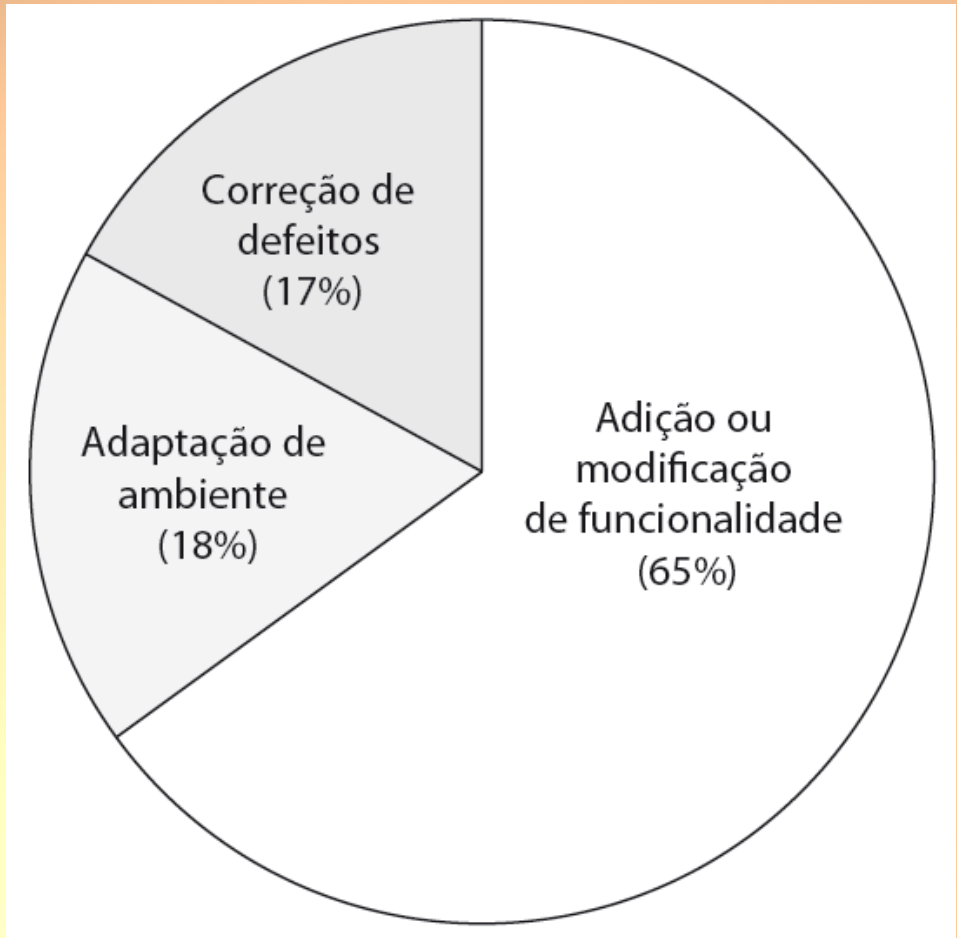
Manutenção de software

- Modificar um programa depois desse ter sido liberado para uso.
- O termo é usado principalmente para mudar software customizado.
- Produtos de software genéricos são ditos para evoluir para criar novas versões.
- Geralmente, a manutenção não envolve mudanças importantes para a arquitetura do sistema.
- As mudanças são implementadas por meio da modificação dos componentes existentes e pela adição de novos componentes ao sistema.

Tipos de manutenção

- Manutenção para corrigir defeitos de software
  - ✓ Mudar um sistema para corrigir deficiências que não correspondem aos seus requisitos.
- Manutenção para adaptar o software a um ambiente operacional diferente.
  - ✓ Mudar um sistema para que ele opere em um ambiente diferente (computador, OS, etc.) da sua implementação inicial.
- Manutenção para adicionar ou modificar a funcionalidade do sistema
  - ✓ Modificando o sistema para satisfazer novos requisitos.

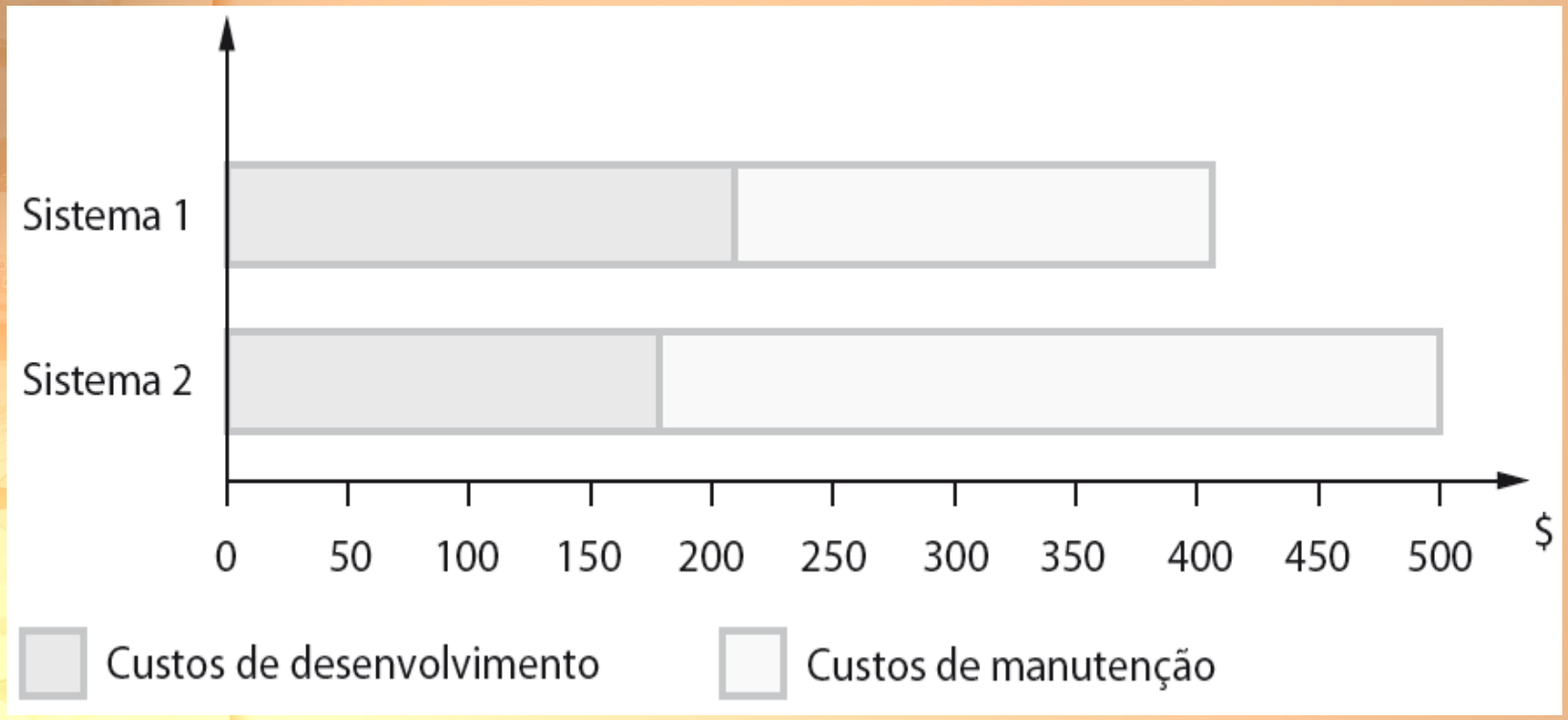
Distribuição do esforço de manutenção



Os custos de manutenção

- Geralmente, maior do que os custos de desenvolvimento (2 * a 100 *, dependendo da aplicação).
- Afetados por fatores técnicos e não técnicos.
- Aumentam na medida em que o software é mantido. A manutenção corrompe a estrutura do software, dificultando futuras manutenções.
- Softwares envelhecidos podem ter altos custos de suporte (por exemplo, linguagens, compiladores antigos etc.)

Custos de desenvolvimento e manutenção



Fatores de custos de manutenção

- Estabilidade da equipe
  - ✓ Custos de manutenção são reduzidos se a mesma equipe estiver envolvida com eles por algum tempo.
- Responsabilidade contratual
  - ✓ Os desenvolvedores de um sistema podem não ter responsabilidade contratual para a manutenção, assim não há incentivo para projetar mudanças futuras.

Fatores de custos de manutenção

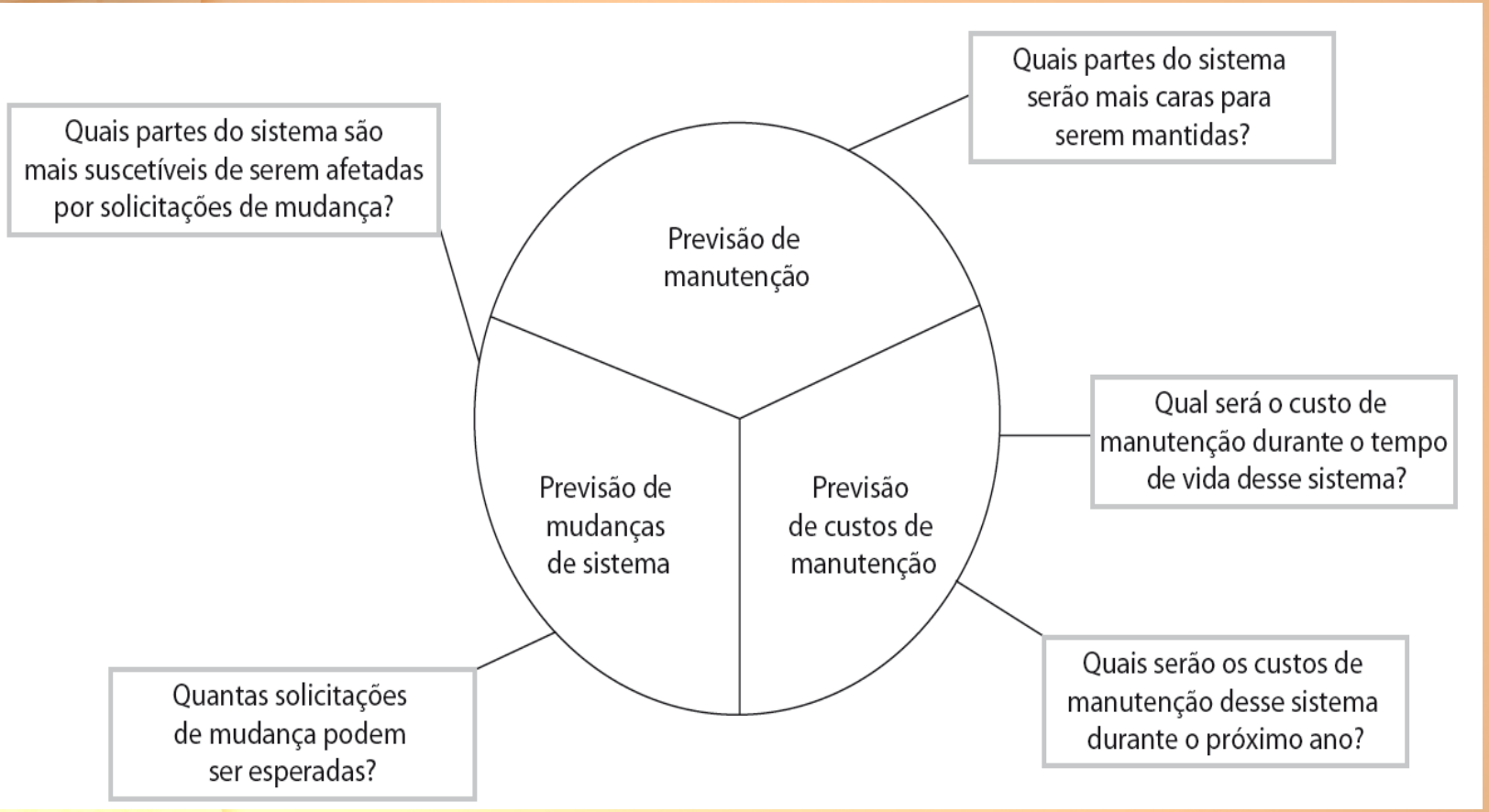
- Qualificações de pessoal
  - ✓ Muitas vezes a equipe de manutenção é inexperiente e tem conhecimentos do domínio limitados.
- Idade e estrutura do programa
  - ✓ Na medida em que os programas envelhecem, sua estrutura se degrada e tornam-se mais difíceis de entender e mudar.

Previsão de manutenção

- A previsão da manutenção está preocupada em avaliar quais partes do sistema podem causar problemas e terão altos custos de manutenção.

  - ✓ A aceitação de mudanças depende da capacidade de manutenção dos componentes afetados pela mudança;
  - ✓ A implementação de mudanças degrada o sistema e reduz sua capacidade de manutenção;
  - ✓ Os custos de manutenção dependem do número de mudanças e os custos de mudança dependem da capacidade de manutenção.

Previsão de manutenção



Previsão de mudanças

- Prever o número de mudanças exige a compreensão do relacionamento entre um sistema e seu ambiente.
- Sistemas fortemente acoplados exigem mudanças sempre que ocorrem alterações no sistema.
- São fatores que influenciam essa relação:
  - ✓ Número e complexidade das interfaces de sistema;
  - ✓ Número de requisitos de sistema inerentemente voláteis;
  - ✓ Os processos de negócio nos quais o sistema é usado.

Métricas de complexidade

- As previsões de manutenção podem ser feitas por meio da avaliação da complexidade dos componentes do sistema.

- Estudos têm demonstrado que a maioria dos esforços de manutenção são gastos em um número relativamente pequeno de componentes do sistema.

- A complexidade depende de:

  - ✓ Complexidade das estruturas de controle;
  - ✓ Complexidade das estruturas de dados;
  - ✓ Método (procedimento) de objetos e tamanho de módulo.

Métricas de processo

- Métricas de processo podem ser usadas para avaliar a capacidade de manutenção
  - ✓ Número de solicitações de manutenção corretiva;
  - ✓ Tempo médio necessário para a análise de impacto;
  - ✓ Tempo médio necessário para implementar uma solicitação de mudança;
  - ✓ Número de solicitações de mudança pendentes.
- Se algum ou todos esses estão aumentando, isso pode indicar um declínio na manutenibilidade.

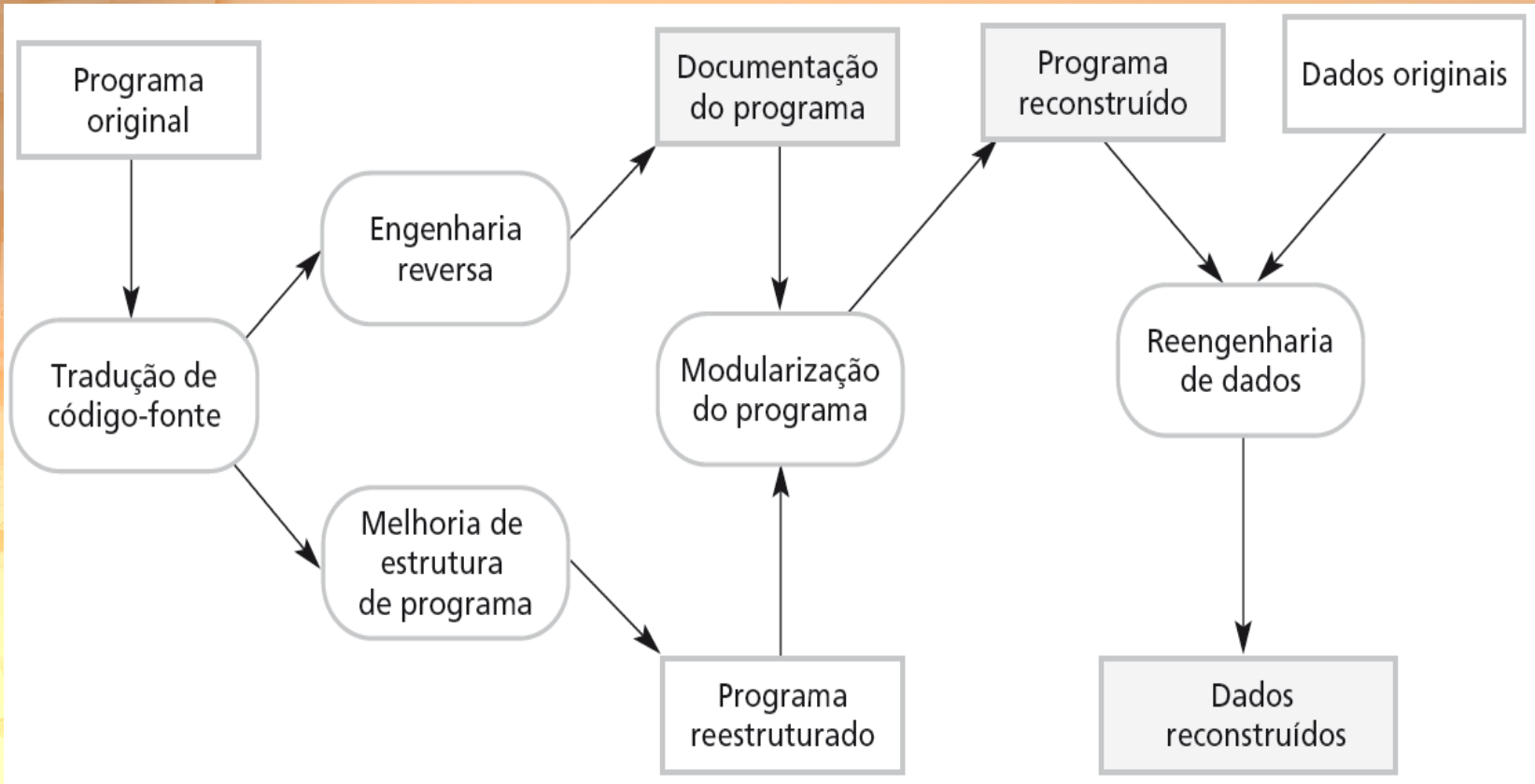
Reengenharia de software

- A reestruturação ou reescrita de parte ou da totalidade de um sistema legado, sem alterar sua funcionalidade.
- Aplicável em casos em que alguns, mas não todos os subsistemas de um sistema maior exigem manutenção frequente.
- A reengenharia envolve esforços adicionais para torná-los mais fáceis de se manter.
- O sistema pode ser reestruturado e redocumentado.

Vantagens da reengenharia

- Risco reduzido
  - ✓ Existe um alto risco no desenvolvimento de software novo. Pode haver problemas de desenvolvimento, problemas com a equipe e problemas de especificação.
- Custo reduzido
  - ✓ Muitas vezes, o custo da reengenharia é significativamente menor do que os custos de desenvolvimento de software novo.

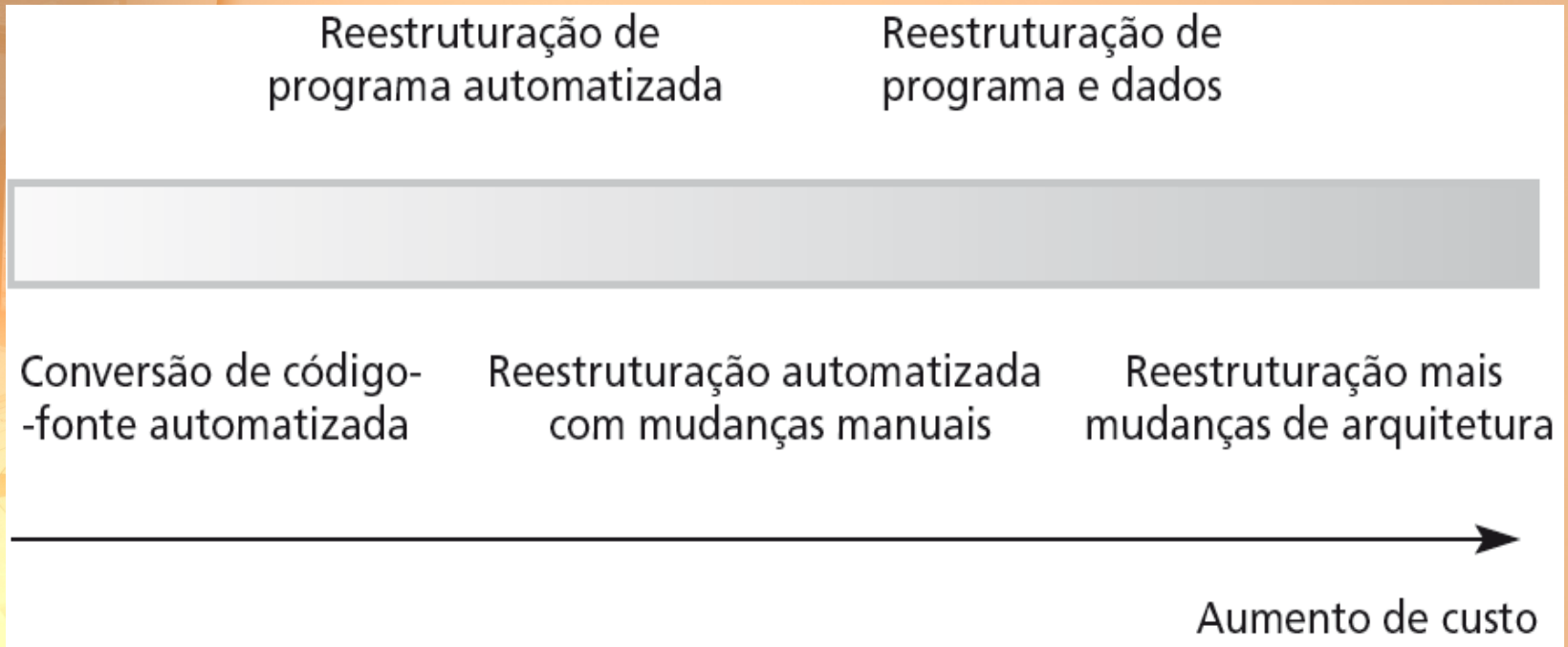
O processo de reengenharia



Atividades do processo de reengenharia

- Tradução de código-fonte
  - ✓ Conversão do código para uma nova linguagem.
- Engenharia reversa
  - ✓ Análise do programa para compreensão desse;
- Melhoria de estrutura de programa
  - ✓ Reestruturação automática para capacidade de compreensão;
- Modularização de programa
  - ✓ Reorganização da estrutura de programa;
- Reengenharia de dados
  - ✓ Limpeza e reestruturação de dados de sistema.

Abordagens de reengenharia



Fatores de custo de reengenharia

- A qualidade do software a ser reestruturado.
- A ferramenta de apoio disponível para a reengenharia.
- A extensão da conversão de dados necessária.
- A disponibilidade de pessoal especializado para a reengenharia.
  - ✓ O que pode ser um problema com velhos sistemas baseados em tecnologias que já não são amplamente usadas.

Manutenção preventiva por refatoração

- A refatoração é o processo de fazer melhorias em um programa para diminuir a degradação por meio da mudança.
- Você pode pensar na refatoração como "manutenção preventiva", o que reduz os problemas de mudanças futuras.
- A refatoração envolve a modificação de um programa para melhoria da sua estrutura, redução da sua complexidade ou facilitar o entendimento.
- Ao refatorar um programa, você não deve adicionar funcionalidade, mas sim concentrar-se na melhoria do programa.

Refatoração e reengenharia

- A reengenharia ocorre depois que um sistema é mantido por algum tempo e os custos de manutenção são crescentes.
- São necessárias ferramentas automatizadas para processar e fazer a reengenharia de um sistema legado para criar um sistema novo, mais manutenível.
- A refatoração é um processo contínuo de melhoria por meio do processo de evolução e desenvolvimento.
- Pretende-se evitar a degradação da estrutura e código que aumenta os custos e as dificuldades de manutenção de um sistema.

“Maus cheiros” no código de programa

- Código duplicado
  - ✓ O mesmo código ou códigos muito semelhantes podem ser incluídos em diferentes lugares de um programa. Estes podem ser removidos e implementados como um único método ou função a qual será chamada quando necessária.

- Métodos longos
  - ✓ Se um método for muito longo, ele deve ser redefinido como uma série de métodos mais curtos.

- Declarações switch (case)
  - ✓ Frequentemente envolve a duplicação, na qual o switch depende do tipo de um valor. As declarações de switch podem ser espalhadas em torno de um programa. Muitas vezes, em linguagens orientadas a objetos, é possível usar polimorfismo para conseguir a mesma coisa.

“Maus cheiros” no código de programa

- Aglutinação de dados
  - ✓ A aglutinação de dados ocorre quando o mesmo grupo de itens de dados (campos em classes, parâmetros em métodos) voltam a ocorrer em vários lugares em um programa. Muitas vezes, esses podem ser substituídos por um objeto que encapsule todos os dados.
- Generalidade especulativa
  - ✓ Ocorre quando os desenvolvedores incluem generalidades em um programa no caso dessas serem necessárias no futuro. Muitas vezes podem, simplesmente, ser removidas.

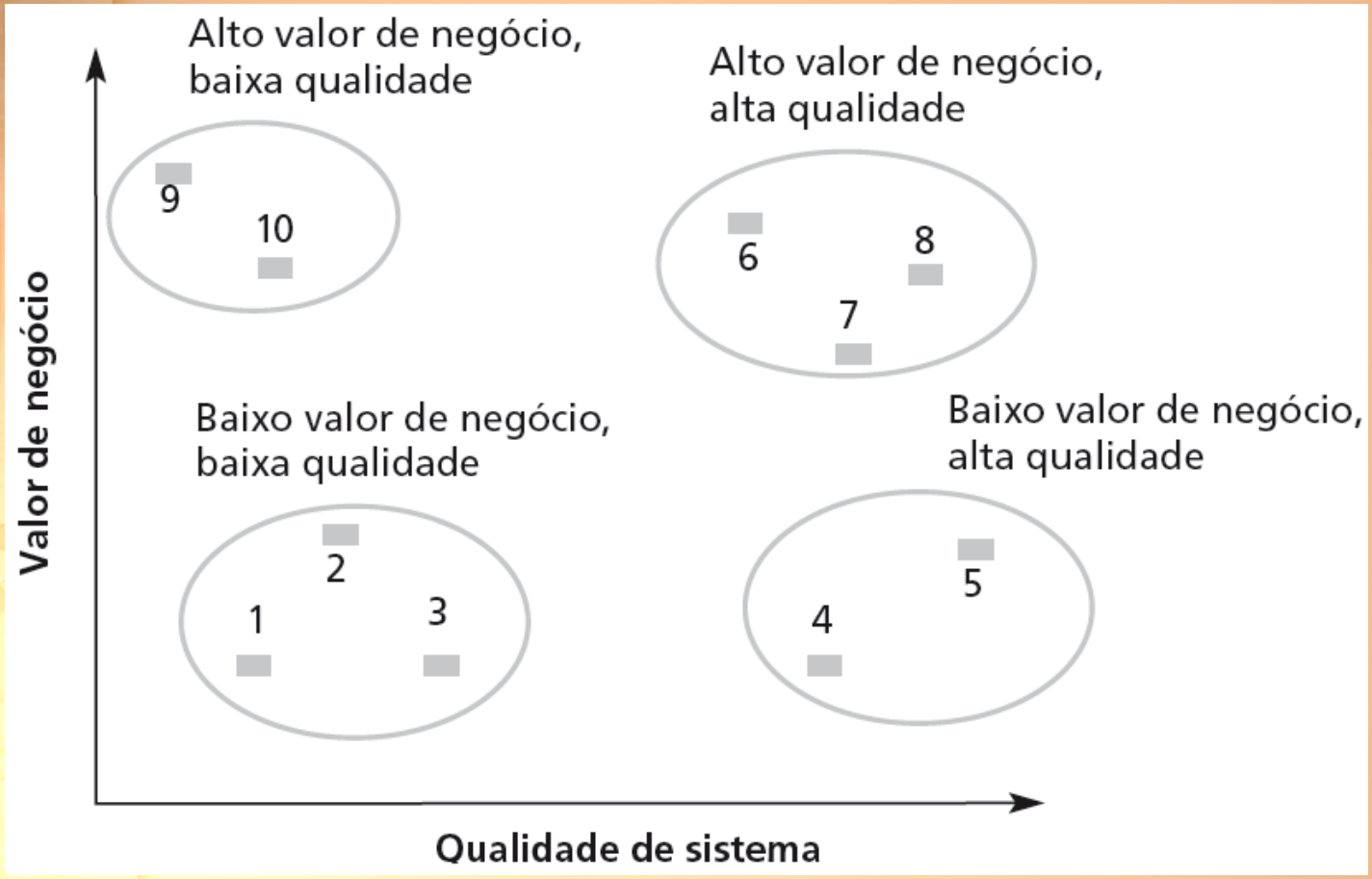
Gerenciamento de sistemas legados

- Organizações que dependem de sistemas legados devem escolher uma estratégia para a evolução desses sistemas

  - ✓ Fazer o descarte completo do sistema e modificar os processos de negócio para que esse não seja mais necessário;
  - ✓ Continuar dando suporte para o sistema;
  - ✓ Reestruturar o sistema por meio da reengenharia para melhoria de sua manutenibilidade;
  - ✓ Substituir o sistema com um novo sistema.

- A estratégia escolhida deve depender da qualidade do sistema e do seu valor de negócio.

Um exemplo de uma avaliação de sistema legado



Categorias de sistemas legados

- Baixa qualidade, baixo valor de negócio
  - ✓ Esses sistemas devem ser descartados.
- Baixa qualidade, alto valor de negócio
  - ✓ Esses fazem uma importante contribuição para os negócios, mas sua manutenção é cara. Deve passar por uma reengenharia ou ser substituído caso um sistema adequado esteja disponível.
- Alta qualidade, baixo valor de negócio
  - ✓ Substituir com COTS, descartar completamente ou manter.
- Alta qualidade, alto valor de negócio
  - ✓ Continuar em operação com a manutenção normal do sistema.

Avaliação do valor de negócio

- A avaliação deve levar em conta diferentes pontos de vista:
  - ✓ Usuários finais do sistema;
  - ✓ Clientes empresariais;
  - ✓ Gerentes de linha;
  - ✓ Gerentes de TI;
  - ✓ Gerentes seniores.
- Entrevistar diferentes *stakeholders* e coletar os resultados obtidos.

Problemas na avaliação do valor de negócio

- O uso do sistema
  - ✓ Se os sistemas são usados apenas ocasionalmente ou por um pequeno número de pessoas, eles podem ter um valor de negócio de baixo.

- Os processos de negócio que são apoiados
  - ✓ Um sistema pode ter um baixo valor de negócio se força o uso de processos de negócios ineficientes.

- Confiança do sistema
  - ✓ Se um sistema não é confiável e seus problemas afetam diretamente seus clientes, o sistema tem um baixo valor de negócio.

- As saídas do sistema
  - ✓ Se o negócio depende das saídas do sistema, então o sistema tem um alto valor de negócio.

Avaliação da qualidade do sistema

- Avaliação dos processos de negócio
  - ✓ O quão bem o processo de negócios dá suporte para as metas atuais da empresa?
- Avaliação ambiental
  - ✓ Quão eficaz é o ambiente do sistema e quão caro é mantê-lo?
- Aplicação da avaliação
  - ✓ Qual é a qualidade do sistema de aplicação de software ?

Avaliação do processo de negócio

- Use uma abordagem orientada a pontos de vista e procure respostas dos *stakeholders* do sistema.

  - ✓ Existe um modelo de processo definido, esse é seguido?
  - ✓ Diferentes partes da organização usam processos diferentes para a mesma função?
  - ✓ Como o processo foi adaptado?
  - ✓ Quais são os relacionamentos com outros processos de negócios, esses são necessários?
  - ✓ O processo efetivamente recebe suporte do software de aplicação legada?

- Exemplo - um sistema de aquisição de viagens pode ter um baixo valor de negócio, em virtude do uso generalizado de aquisições baseadas em Web.

Fatores usados na avaliação de ambientes

Fator	Questões
Estabilidade do fornecedor	O fornecedor ainda existe? O fornecedor é financeiramente estável e deve continuar existindo? Se o fornecedor não está mais no negócio, existe alguém que mantém os sistemas?
Taxa de falhas	O hardware tem uma grande taxa de falhas reportadas? O software de apoio trava e força o reinício do sistema?
Idade	Quantos anos têm o hardware e o software? Quanto mais velho o hardware e o software de apoio, mais obsoletos serão. Ainda podem funcionar corretamente, mas poderia haver significativos benefícios econômicos e empresariais se migrassem para um sistema mais moderno.
Desempenho	O desempenho do sistema é adequado? Os problemas de desempenho têm um efeito significativo sobre os usuários do sistema?
Requisitos de apoio	Qual apoio local é requisitado pelo hardware e pelo software? Se houver altos custos associados a esse apoio, pode valer a pena considerar a subsituição do sistema.
Custos de manutenção	Quais são os custos de manutenção de hardware e de licenças de software de apoio? Os hardwares mais antigos podem ter custos de manutenção mais elevados do que os sistemas modernos. Os softwares de apoio podem ter altos custos de licenciamento anual.
Interoperabilidade	Existem problemas de interface do sistema com outros sistemas? Compiladores podem, por exemplo, ser usados com as versões atuais do sistema operacional? É necessária a emulação do hardware?

Fatores usados na avaliação de aplicações

Fatores	Questões
Inteligibilidade	Quão difícil é compreender o código-fonte do sistema atual? Quão complexas são as estruturas de controle usadas? As variáveis têm nomes significativos que refletem sua função?
Documentação	Qual documentação do sistema está disponível? A documentação é completa, consistente e atual?
Dados	Existe um modelo de dados explícito para o sistema? Até que ponto os dados nos arquivos estão duplicados? Os dados usados pelo sistema são atuais e consistentes?
Desempenho	O desempenho da aplicação é adequado? Os problemas de desempenho têm um efeito significativo sobre os usuários do sistema?
Linguagem de programação	Compiladores modernos estão disponíveis para a linguagem de programação usada para desenvolver o sistema? A linguagem de programação ainda é usada para o desenvolvimento do novo sistema?
Gerenciamento de configuração	Todas as versões de todas as partes do sistema são gerenciadas por um sistema de gerenciamento de configuração? Existe uma descrição explícita das versões de componentes usadas no sistema atual?
Dados de teste	Existem dados de teste para o sistema? Existem registros dos testes de regressão feitos quando novos recursos forem adicionados ao sistema?
Habilidades de pessoal	Existem pessoas disponíveis com as habilidades necessárias para manter a aplicação? Existem pessoas disponíveis que tenham experiência no sistema?

Medições do sistema

- Você pode coletar dados quantitativos para fazer uma avaliação da qualidade do sistema de aplicação.
  - ✓ O número de solicitações de mudança no sistema;
  - ✓ O número de diferentes interfaces de usuário usadas pelo sistema;
  - ✓ O volume de dados usados pelo sistema.

Pontos Importantes

- Existem três tipos de manutenção de software, correção de bugs, modificação do software para funcionar em um novo ambiente, e implementação de requisitos novos ou alterados.
- A reengenharia de software está preocupada com a reestruturação e a redocumentação do software para torná-lo mais fácil de entender e mudar.
- Refatoração, fazer pequenas alterações no programa, as quais devem preservar a funcionalidade, é uma forma de manutenção preventiva.
- O valor de negócio de um sistema legado e a qualidade do software de aplicação devem ser avaliadas para ajudar a decidir se o sistema deve ser substituído, transformado ou mantido.