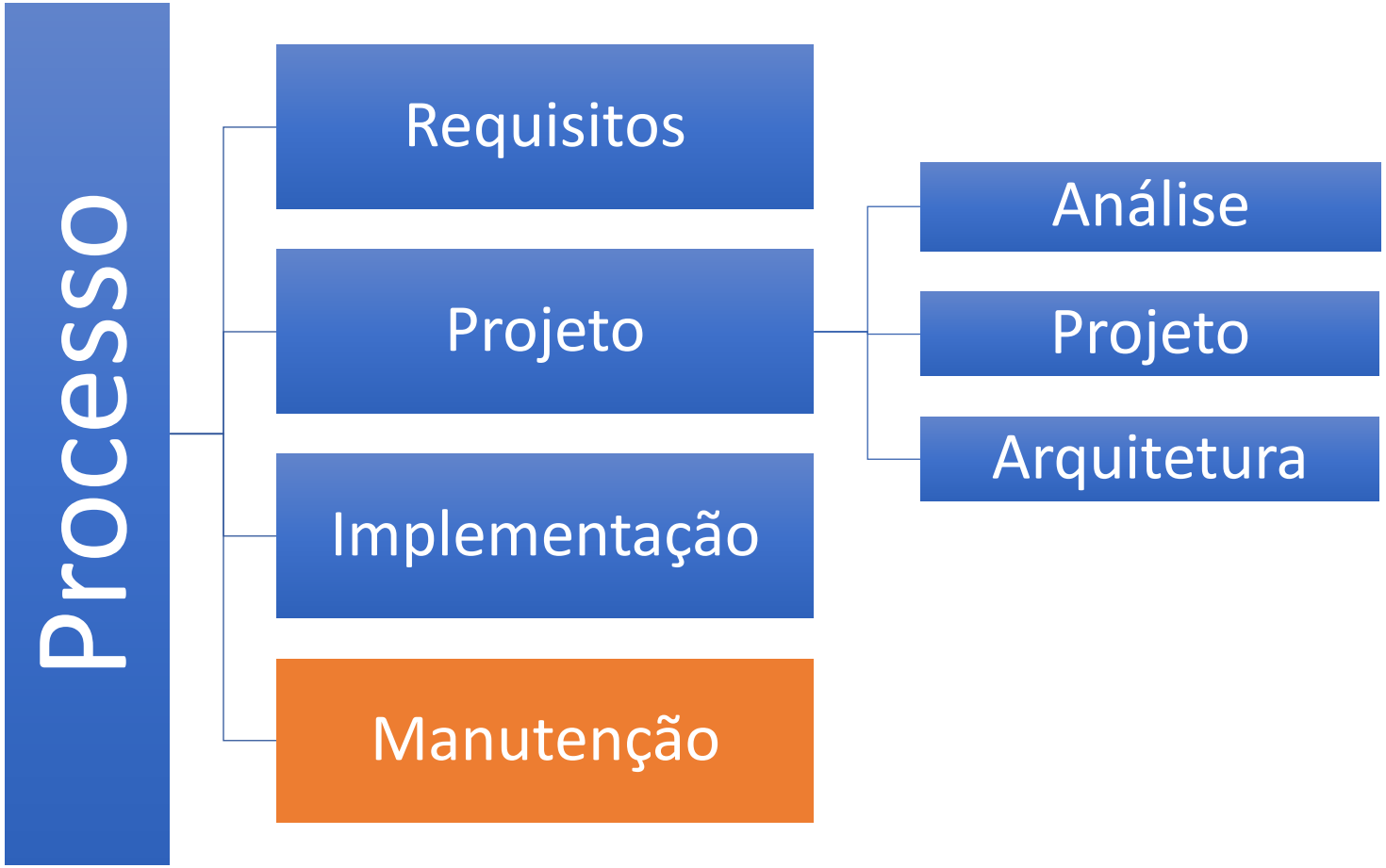
Suporte de Software

Elton Morais

Introdução

Engenharia de Software



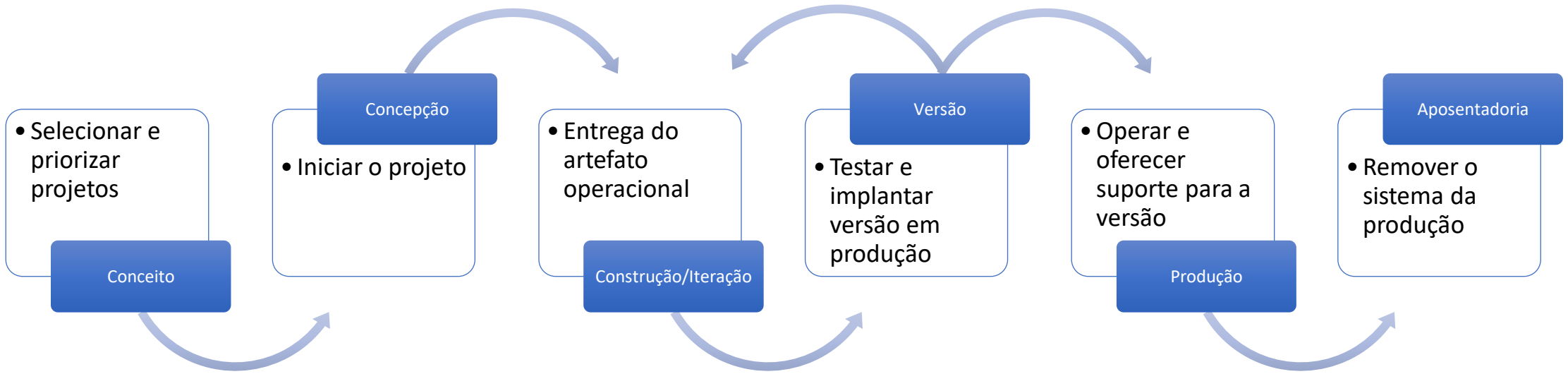
Introdução

A **manutenção** de software é um processo crucial para garantir a **qualidade** e a **confiabilidade** de um software ao longo do tempo.

Ela envolve atividades como correção de bugs, adaptação a novas funcionalidades e otimização de desempenho.

“É o processo de modificar um sistema de software após sua implementação para corrigir defeitos, melhorar o desempenho ou adaptar-se a uma mudança ambiental”.

Introdução



O suporte de software começa quando os desenvolvedores convidam os envolvidos para participar do processo de coleta de requisitos e evolução do protótipo e termina com a decisão de aposentar o sistema e retirá-lo do uso ativo.

Teoria unificada para a evolução do software

As leis de Lehman.

Teoria unificada para a evolução do software

- **Lei da Mudança Contínua:**
  - Software implementado em um contexto de computação real e que, portanto, vai evoluir com o tempo. Deve ser adaptado continuamente ou se tornará cada vez menos satisfatório.
- **Lei da Complexidade Crescente:**
  - A medida que um sistema evolui, sua complexidade aumenta, a menos que seja feito um trabalho para mantê-la ou reduzi-la.

Teoria unificada para a evolução do software

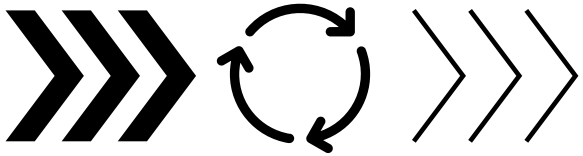
- **Lei da Conservação da Familiaridade:**
  - Conforme um sistema evolui, todos que estão associados a ele – desenvolvedores, pessoal de vendas, usuários – devem manter um conhecimento sobre o seu conteúdo e comportamento para uma evolução satisfatória. Um crescimento excessivo diminui esse conhecimento. Portanto, o crescimento incremental médio permanece invariante à medida que o sistema evolui.

Teoria unificada para a evolução do software

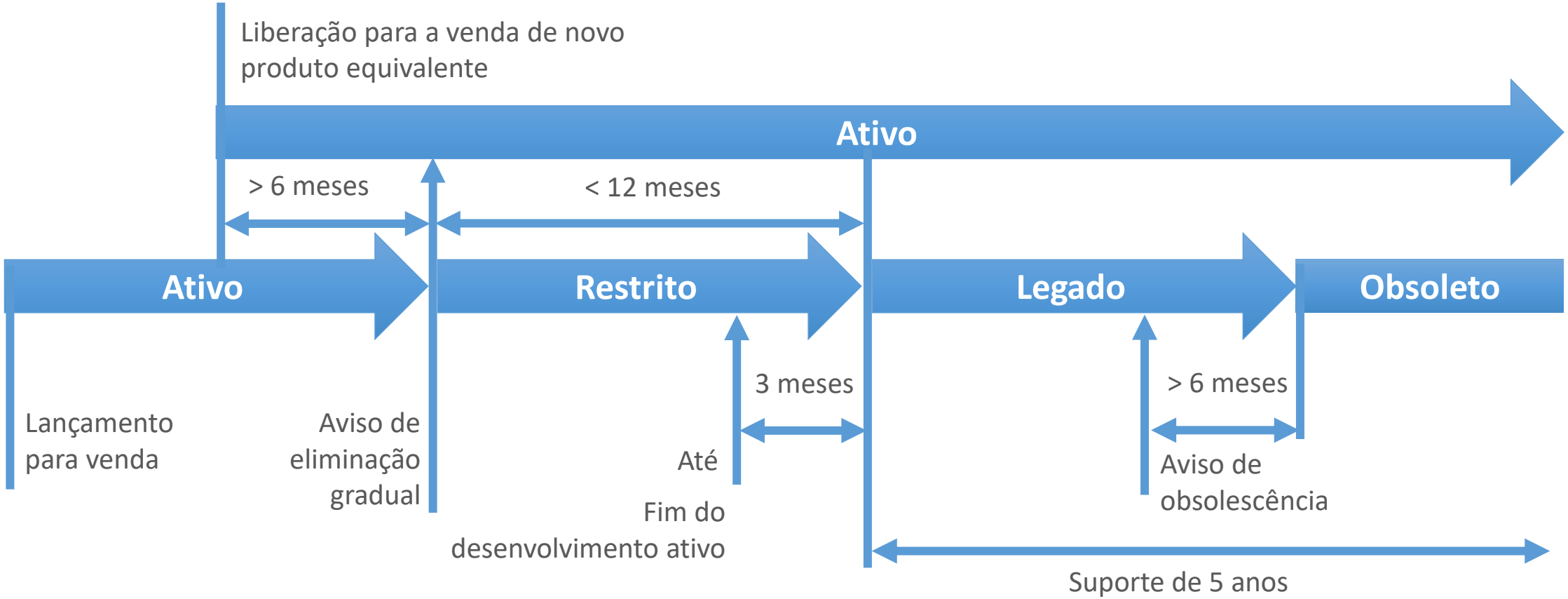
- **Lei do Crescimento Contínuo:**
  - O conteúdo funcional dos sistemas deve ser continuamente ampliado durante toda a sua existência para manter a satisfação do usuário.
- **Lei da Qualidade em Declínio:**
  - A qualidade dos sistemas vai parecer diminuir a menos que eles sejam rigorosamente mantidos e adaptados às mudanças do ambiente operacional.

Suporte de Software

O suporte de software pode ser considerado uma atividade de apoio que inclui gestão de alterações, gestão de risco proativa, gestão de processo, gestão de configuração, garantia da qualidade e gerencimento de versões.

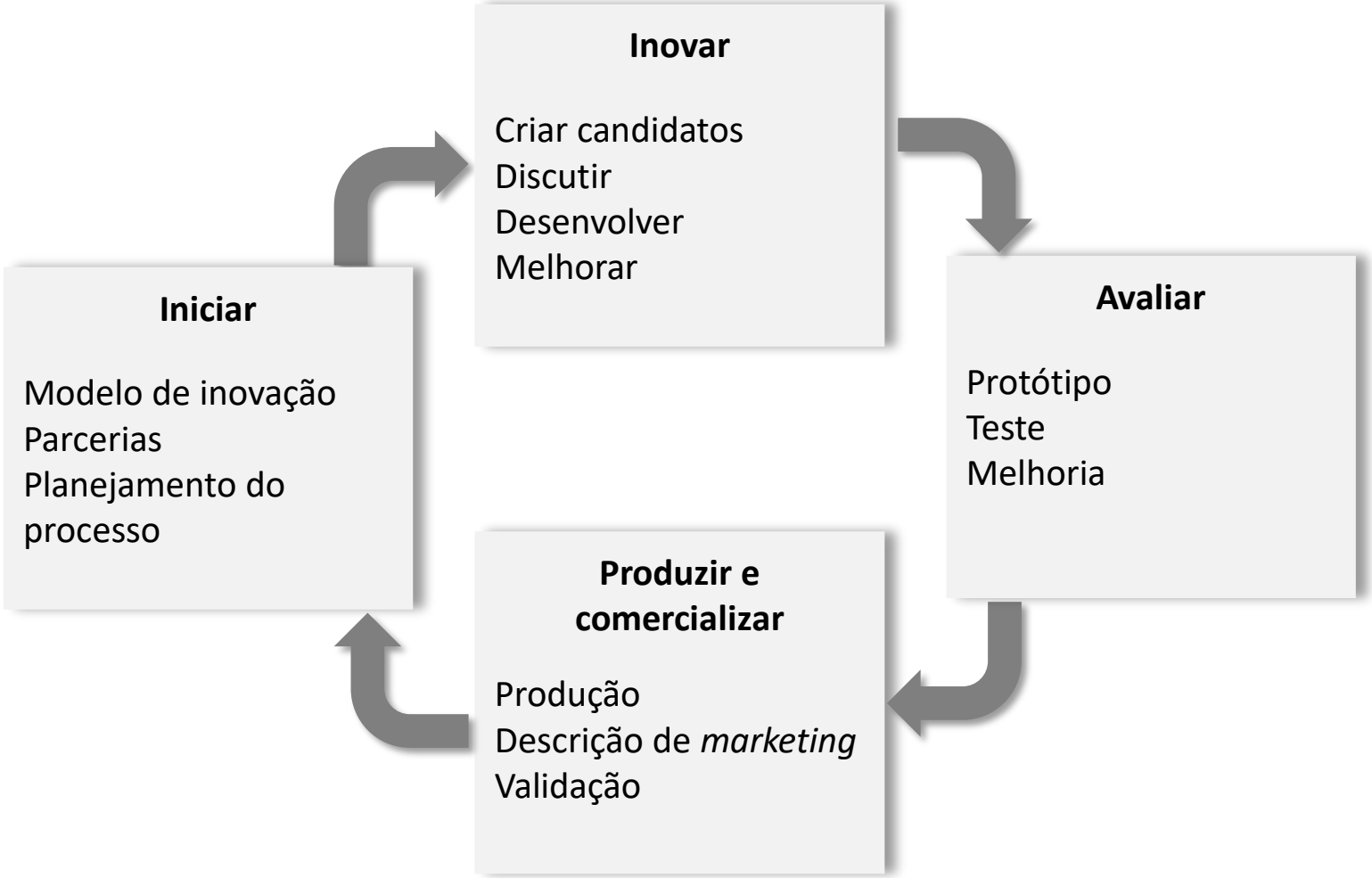


Linha do tempo de um software



Linha do tempo que vai do lançamento à aposentadoria de um produto de software.
Fonte:

Modelo de suporte de software



Fonte:

Suportabilidade

A ***suportabilidade*** é a capacidade de fornecer suporte a um sistema de software durante toda a vida útil do produto.

Isso implica em satisfazer qualquer *necessidade* ou *requisito*, assim como *equipamento*, *infraestrutura* de suporte, *software* adicional, *serviços* de conveniências, *mão de obra* ou qualquer outro recurso para manter o produto operacional e satisfatório.

É um dos muitos fatores de qualidade que devem ser considerados durante as ações de análise e projeto que são parte do processo de software.

Manutenção de Software

A Manutenção é definida como as atividades necessárias para manter o software operacional após ele ser aceito e entregue no ambiente do usuário.

Manutenção de software

- O **software é liberado** para os usuários, e a *manutenção começa quase imediatamente*.
  - Em alguns dias começam os relatos de erros;
  - Em algumas semanas um grupo de usuários indica mudanças no software para melhor atender suas necessidades.
  - Em meses, outro grupo, que ainda não estava interessado no software, mas reconhece suas vantagens, solicita melhorias para adequar ao seu mundo.

Manutenção de software

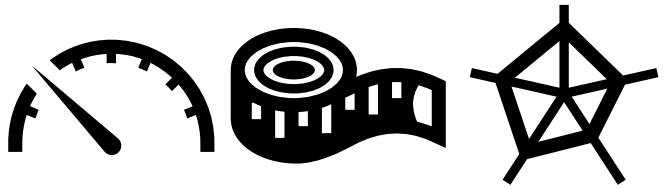
- Com a liberação do software, a **fila** de **correções de erros**, **solicitações de adaptação** e **melhorias** que devem ser planejadas, programadas e, por fim executadas se torna cada vez maior.
- O crescimento continuo da fila ameaça consumir os recursos disponíveis, e pode fazer com que a empresa gaste **mais tempo e dinheiro** com manutenção do que com a criação de novas aplicações.

Manutenção de software

- Mudanças são inevitáveis, portanto, é imprescindível desenvolver mecanismos para avaliar, controlar e fazer modificações.
- Tais mecanismos partem da análise e do projeto, que iram fornecer características essenciais de manutenibilidade.
- A manutenibilidade é um indicativo de qualitativo da facilidade de corrigir, adaptar ou melhorar o software.

Métricas Chidamber-Kemerer (CK)

As métricas CK são utilizadas na análise da testabilidade para softwares orientado a objetos.



Métricas para POO

- Métricas de Chidamber-Kemerer (CK)
  - Profundidade da Herança (DIT);
  - Número de Filhos (NOC);
  - Acoplamento entre Objetos (CBO);
  - Resposta para uma Classe (RFC);
  - Falta de Coesão em Métodos (LCOM);
  - Métodos Ponderados por Classes (WMC);

Métricas para POO

- Profundidade da Herança (DIT):
  - É definida como o comprimento máximo do nó até a raiz da árvore;
  - Quanto maior a profundidade, maior a complexidade do projeto;
- Número de Filhos (NOC):
  - Conta o número de subclasses diretas;
  - Quanto maior o valor, maior indicativo de reuso;
- Acoplamento entre Objetos (CBO):
  - Conta o número de classes chamadas por uma classe;
  - Quanto maior o número, mais difícil de entender e manter;

Métricas para POO

- Resposta para uma Classe (RFC):
  - Conta o número de métodos que podem ser executados em resposta a uma mensagem recebida de outro objeto;
  - Quanto maior o número, mais complexa é a classe;
- Falta de Coesão em Métodos (LCOM):
  - Conta o número de métodos de uma classe acessam atributos em comum;
  - Maior número de atributos, maior coesão, menor perda de coesão;
- Métodos Ponderados por Classes (WMC):
  - Atribuição de pesos aos métodos de uma classe. Pode ser por linha de código, número de parâmetros etc.;
  - Maior valor, indica maior complexidade.

Tipos de Manutenção

Existem diferentes tipos de manutenção de software, cada um com seu próprio objetivo e atividades específicas.

1 2 3 4

Manutenção de software

A **manutenção corretiva** é a modificação reativa do software para consertar problemas descobertos após sua entrega.

É uma a manutenção realizada para corrigir problemas em um software já em produção. Esses problemas podem ser erros lógicos, de programação ou de funcionamento do software.

É o tipo de manutenção mais frequente, podendo chegar a cerca de 60% de todo o esforço de manutenção.

Manutenção de software

A **manutenção adaptativa** é a modificação reativa do software após a entrega para mantê-lo utilizável em um ambiente do usuário em mutação.

É realizada para adaptar um software às mudanças no ambiente em que ele opera. Essas mudanças podem ser alterações em sistemas operacionais, em equipamentos, em protocolos de comunicação, entre outros.

Ela pode representar cerca de 25% de todo o esforço de manutenção.

Manutenção de software

A **manutenção perfectiva** é a modificação proativa do software após a entrega para adicionar novos recursos para o usuário.

É realizada para aprimorar a qualidade ou a eficiência de um software já em produção. Essa manutenção pode incluir atividades como otimização de desempenho, melhorias na interface do usuário, entre outras.

Pode representar cerca de 10% de todo o esforço de manutenção.

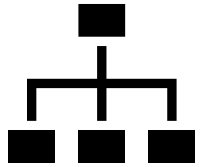
Manutenção de software

A **manutenção preventiva** é a modificação proativa do software após a entrega para detectar e corrigir falhas do artefato antes que sejam descobertas pelos usuários no campo.

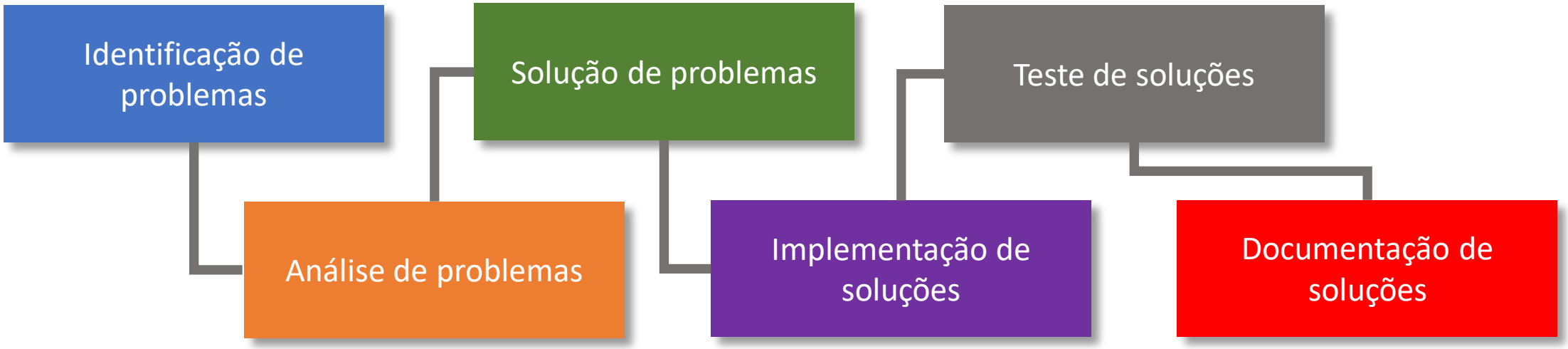
É realizada para evitar problemas futuros no software. Essa manutenção pode incluir atividades como revisão de código, atualização de bibliotecas, entre outras.

Pode representar cerca de 5% de todo o esforço de manutenção.

Processos de Manutenção

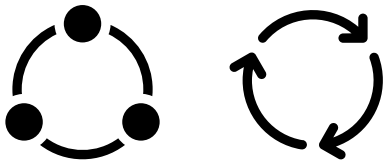


Manutenção de software



Evolução de software

A evolução do software é o processo de desenvolvimento, manutenção e atualização do software por vários motivos.



Evolução de software

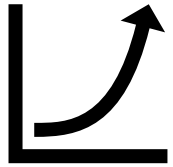
As mudanças de software são inevitáveis porque há muitos fatores que mudam durante o ciclo de vida de um software.

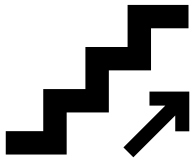
Logo, a reengenharia é uma estratégia que pode ser adotada para sanar diversos problemas.

Entretanto é um processo que toma tempo e tem custo financeiro alto.

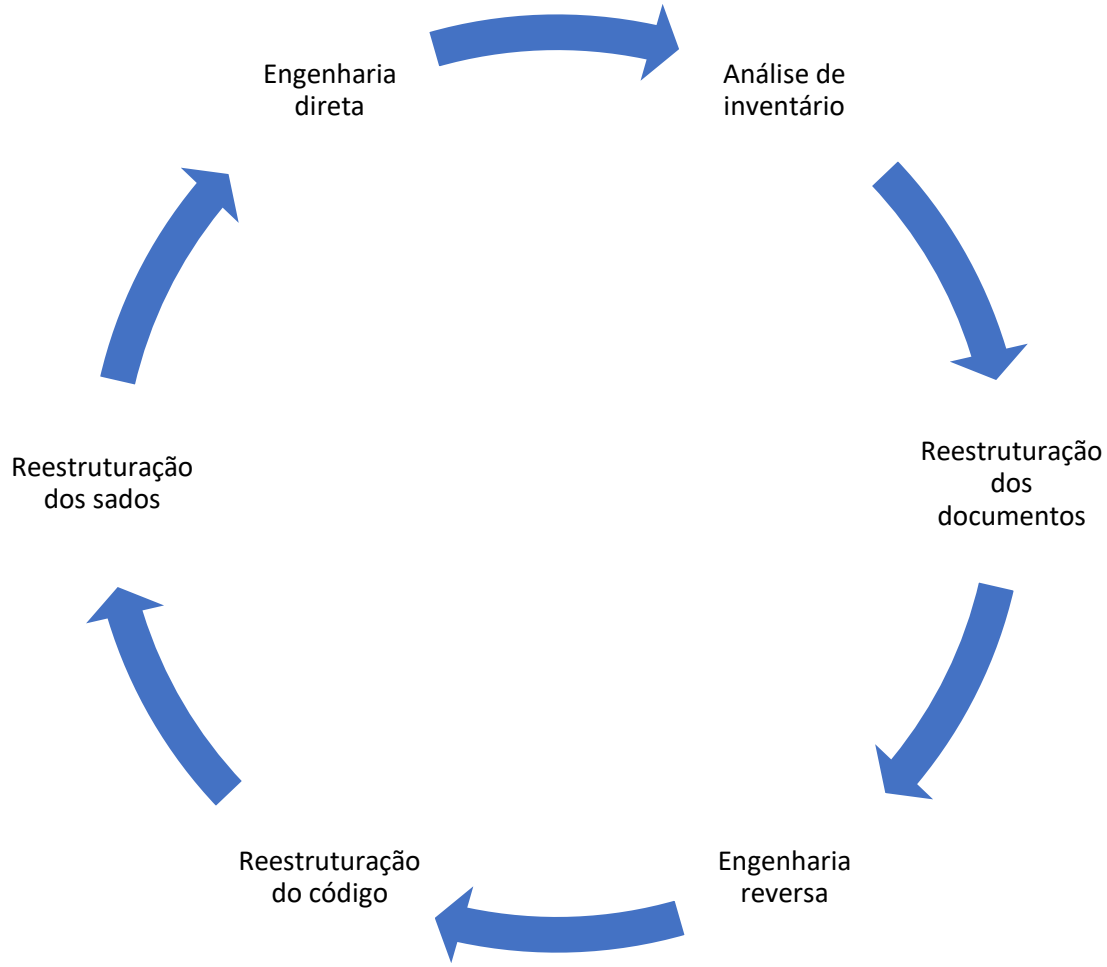
Nesse sentido, pode-se aplicar o princípio de Pareto, que submete 20% do software ao processo de reengenharia, que são responsáveis por 80% dos problemas.

Processo de reengenharia de software





Evolução de software



Evolução de software

- Análise de inventário:
  - Toda organização deve ter um inventário de todas as aplicações, contendo tamanho, idade, criticalidade nos negócios de cada aplicação ativa.
- Reestruturação dos documentos:
  - Cada organização deve escolher a opção mais adequada para documentação para cada caso. É importante documentar todas as interações e/ou intervenções realizadas no software.
- Engenharia reversa:
  - É processo de recuperação do projeto através de uma representação mais alto nível.

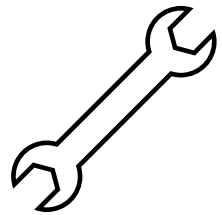
Evolução de software

- Refatoração de código:
  - É o tipo mais comum de reengenharia, que consiste em analisar um código-fonte com uma ferramenta de reestruturação, que analisa as violações e então refatora ou reescreve o código.
- Refatoração de dados:
  - A refatoração de dados inicia com a engenharia reversa, onde a arquitetura de dados atual é dissecada, e os modelos de dados necessários são definidos. Identifica-se os objetos de dados e atributos, e a qualidade das estruturas de dados existentes é revisada.

Evolução de software

- Engenharia direta:
  - A engenharia direta recupera as informações do projeto do software existente, e as utiliza para alterar ou reconstituir o sistema, em um esforço para melhorar sua qualidade geral.

Ferramentas



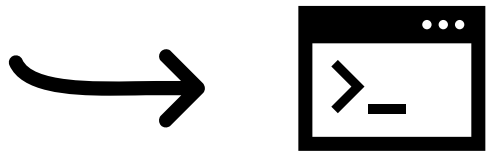
Ferramentas

- Ferramentas de versionamento;
- Ferramentas de rastreamento de problemas;
- Ferramentas de integração contínua;
- Ferramentas de análise estática de código;
- Ferramentas de testes automatizados;

Ferramenta de versionamento

Uma ferramenta de versionamento gerencia todo o ciclo de vida do trabalho, ou seja, guarda o histórico de cada documento (imagem, código, PDF, entre outros) do projeto.

A composição e o significado dos números das versões é uma escolha particular, mas essa escolha está relacionada com o processo de desenvolvimento. Os números refletem fatos ocorridos durante o trabalho e demonstram sua utilidade quando é necessário verificar compatibilidades e controlar alterações.



Ferramenta de versionamento

O principais recursos, geralmente são:

- rastreamento de alterações desde a versão inicial;
- possibilidade de percorrer o histórico de mudanças e voltar a uma versão anterior;
- controle de acesso que permite o trabalho de várias pessoas ao mesmo tempo;
- ramificação de versões, que possibilita uma linha paralela de desenvolvimento.

Ferramentas de versionamento

Ferramentas de rastreamento de problemas

Plataforma de relato e acompanhamento de bugs.

Funcionamento em nuvem e integração com a ferramenta JIRA.

Código aberto.

Integra diretamente ao código. Suporta projetos ágeis.

Código aberto, múltiplos bancos de dados, web e mobile.

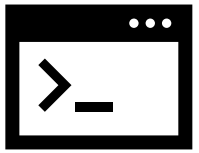
Código aberto, múltiplos bancos de dados

Ferramentas de integração contínua

Integração contínua é uma prática de automatizar a integração de alterações de código de vários contribuidores em um único projeto de software.

É uma prática que permite que os desenvolvedores mesclem com frequência as alterações de código em um repositório central onde *builds* e testes são executados.

Ferramentas automatizadas são usadas para afirmar a correção do novo código antes da integração.



Ferramentas de integração contínua

Bitbucket Pipelines

Jenkins

AWS CodePipeline

circleci

Azure Pipelines

GitLab

Ferramentas de análise estática de código

VisualCodeGrepper é uma ferramenta de análise de código-fonte super rápida para diversas linguagens.

RISP é uma ferramenta de análise de código que identifica vulnerabilidade específica para PHP, Java e Node.Js.

Bandido é uma ferramenta gratuita para encontrar problemas de segurança em Python.

Flawfinder

Finds vulnerabilities in C/C++ source code

Flawfinder é uma ferramenta gratuita de varredura de segurança em código C ou C++.

Ferramentas de testes automatizados

Boas práticas

Para garantir uma manutenção eficiente e de qualidade em um software, é importante seguir algumas boas práticas.

Boas práticas

Documentação: manter a documentação do software atualizada e completa ajuda a entender melhor como ele funciona e facilita a identificação de problemas e soluções.

Refatoração: consiste em melhorar a estrutura e o design do código-fonte sem alterar seu comportamento externo. Isso ajuda a reduzir a complexidade do software e torna sua manutenção mais fácil e eficiente.

Boas práticas

Testes automatizados: a realização de testes automatizados ajuda a identificar problemas e defeitos no software de forma rápida e eficiente, permitindo que eles sejam corrigidos antes que se tornem mais graves.

Controle de versão: o uso de um sistema de controle de versão permite a rastrear mudanças no software e facilita o trabalho em equipe, permitindo que várias pessoas trabalhem no mesmo projeto de forma organizada e segura.

Resumo

O suporte de software é uma atividade contínua que ocorre ao longo de todo o ciclo de vida de uma aplicação.

Resumo

Durante o suporte, são iniciadas ações de manutenção, defeitos são corrigidos, aplicações são adaptadas a um ambiente operacional ou de negócio, melhorias são implementadas por solicitação dos envolvidos.

As atividades de manutenção e suporte de software precisam ser proativas.

A reengenharia examina os sistemas de informações e as aplicações, com a finalidade de reestruturá-los para que tenham melhor qualidade.