

Introdução à ES



Histórico (era pré-ES)

- 1940s: Primeiro computador eletrônico de uso geral
 - ENIAC
 - Custo estimado de US\$ 500.000,00
 - Início da programação de computadores
- 1950s: Primeiros compiladores e interpretadores
- 1960s: Primeiro grande software relatado na literatura – OS/360
 - Mais de 1000 desenvolvedores
 - Custo estimado de US\$ 50.000.000,00 por ano
- 1968: Crise do software – nasce a Engenharia de Software

Histórico (era pós-ES)

- 1970s:
 - Lower-CASE tools (programação, depuração, colaboração)
 - Ciclo de vida cascata
 - Desenvolvimento estruturado
- 1980s:
 - Ciclo de vida espiral
 - Desenvolvimento orientado a objetos
 - Controle de versões
 - Testes
- 1990s: Upper-CASE tools
 - Processos
 - Modelagem

Histórico (era pós-ES)

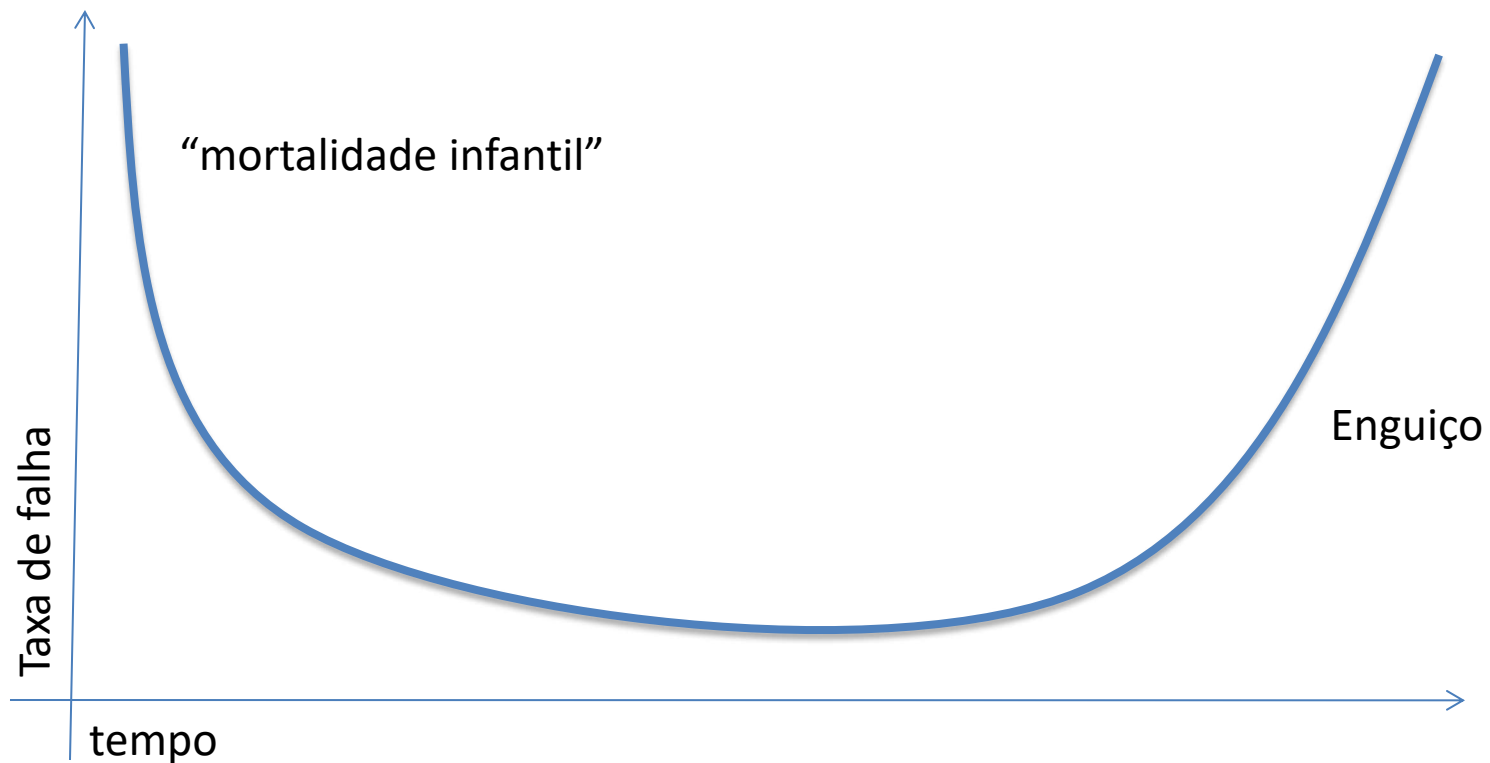
- 2000s:
 - Métodos ágeis
 - Desenvolvimento dirigido por modelos
 - Linhas de produto
 - Experimentação
- Atualmente
 - DevOps
 - Continuous*
 - Software Analytics
 - ...

Software x Hardware

- Software é desenvolvido
 - Alto custo de criação
 - Baixo custo de reprodução
 - Não enguiça, mas deteriora
 - Defeitos no produto usualmente são consequências de problemas no processo de desenvolvimento
- Hardware é manufaturado
 - Alto custo de reprodução
 - Pode enguiçar
 - Defeitos podem vir tanto da concepção quanto da produção
 - Pode ser substituído na totalidade ou em partes

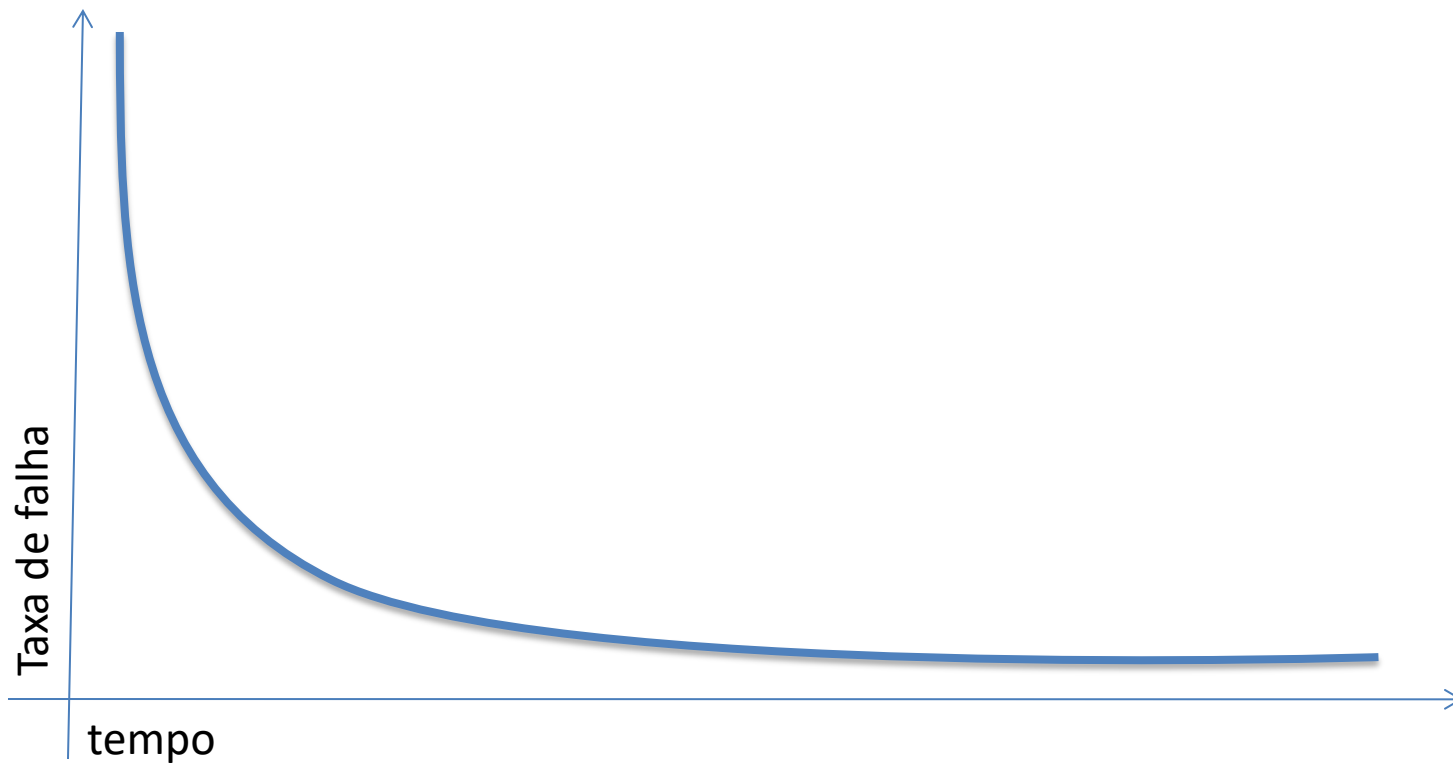
Software x Hardware

- Curva de falha de hardware



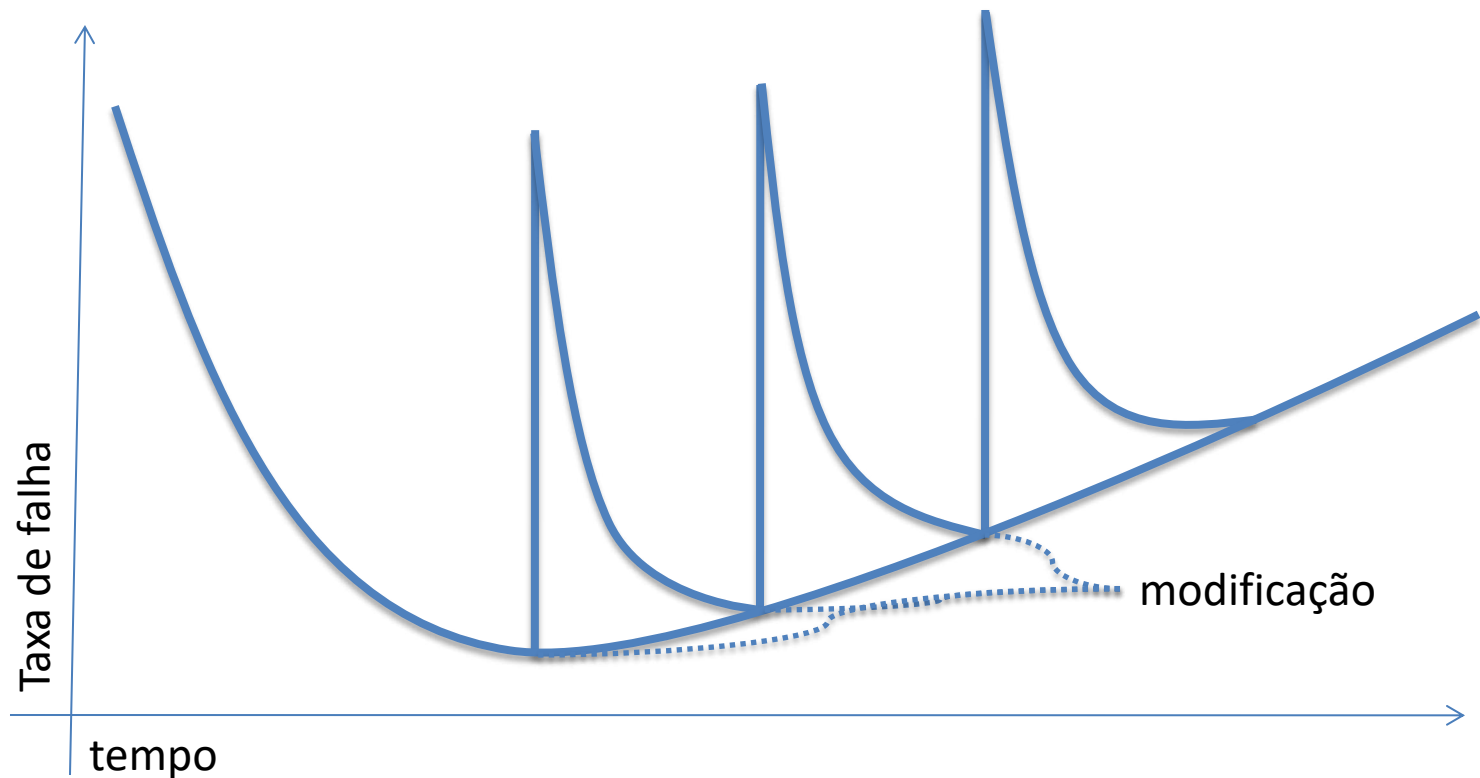
Software x Hardware

- Curva ideal de falha de software

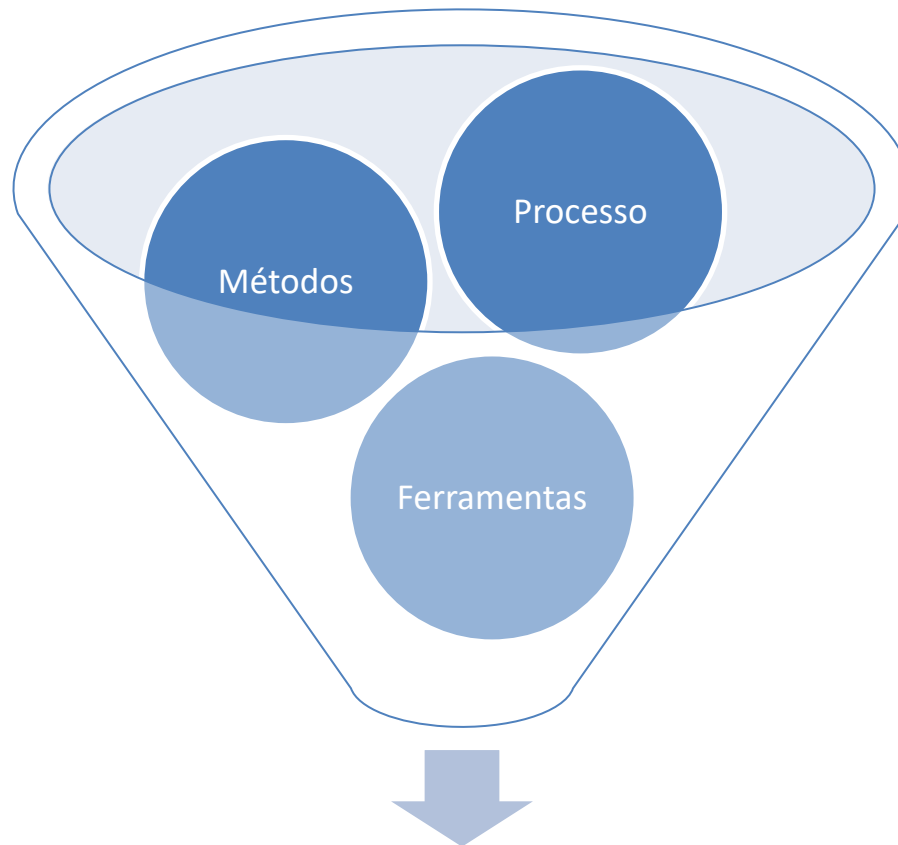


Software x Hardware

- Curva real de falha de software



Elementos da ES



Engenharia de Software

Elementos da ES

- Processo
 - Define os passos gerais para o desenvolvimento e manutenção do software
 - Serve como uma estrutura de encadeamento de métodos e ferramentas
- Métodos
 - São os “how to’s” de como fazer um passo específico do processo
- Ferramentas
 - Automatizam o processo e os métodos

Elementos da ES

- Cuidado com o “desenvolvimento guiado por ferramentas”
 - É importante usar a ferramenta certa para o problema
 - O problema não deve ser adaptado para a ferramenta disponível



“Para quem tem um martelo, tudo parece prego”

Elementos da ES

1. Coloque em uma panela funda o leite condensado, a margarina e o chocolate em pó.
2. Cozinhe [no fogão] em fogo médio e mexa sem parar com uma colher de pau.
3. Cozinhe até que o brigadeiro comece a desgrudar da panela.
4. Deixe esfriar bem, então unte as mãos com margarina, faça as bolinhas e envolva-as em chocolate granulado.

O que é
processo,
método ou
ferramenta?

<http://tudogostoso.uol.com.br/receita/114-brigadeiro.html>

Elementos da ES

1. **Coloque** em uma **panela** funda o leite condensado, a margarina e o chocolate em pó.
2. **Cozinhe** [no **fogão**] em fogo médio e **mexa** sem parar com uma **colher de pau**.
3. **Cozinhe** até que o brigadeiro comece a desgrudar da **panela**.
4. Deixe esfriar bem, então **unte** as mãos com margarina, **faça as bolinhas** e **envolva**-as em chocolate granulado.



método



ferramenta

Processo

<http://tudogostoso.uol.com.br/receita/114-brigadeiro.html>

O Supermercado de ES

- ES fornece um conjunto de métodos para produzir software de qualidade
- Pense como em um supermercado...
 - Em função do problema, se escolhe o processo, os métodos e as ferramentas
- Cuidado
 - Menos do que o necessário pode levar a desordem
 - Mais do que o necessário pode emperrar o projeto

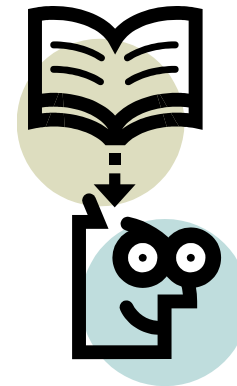
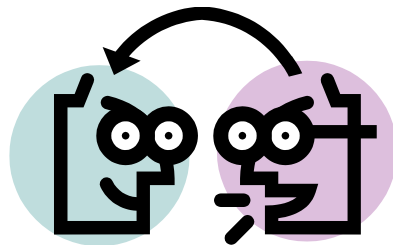


Exercício

- Problema
 - Definir o procedimento de implantação para os dois cenários a seguir
- Caso 1: urna eletrônica
 - O software da urna eletrônica acabou de ser implementado, e precisa ser instalado em 480 mil urnas
- Caso 2: padaria
 - O software de controle de venda de pão da padaria do seu Zé acabou de ser implementado, e precisa ser instalado

Processos implícitos x explícitos

- Lembrem-se: Processos sempre existem, seja de forma implícita ou explícita!
 - Processos implícitos são difíceis de serem seguidos, em especial por novatos
 - Processos explícitos estabelecem as regras de forma clara



Processo de qualidade

- Última palavra para medir a qualidade de um processo: **Satisfação do Cliente**
- Outros indicadores importantes
 - Qualidade dos produtos gerados
 - Custo real do projeto
 - Duração real do projeto

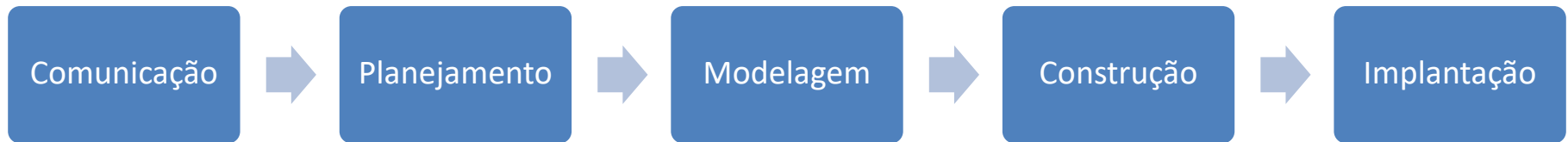
Modelos de ciclo de vida

- Existem alguns processos pré-fabricados
 - Esses processos são conhecidos como modelos de ciclo de vida
 - Esses processos apresentam características predefinidas
- Devem ser adaptados para o contexto real de uso
 - Características do projeto
 - Características da equipe
 - Características do cliente

Exercício

- Assuma as atividades básicas de todo processo como sendo
 - Comunicação
 - Planejamento
 - Modelagem
 - Construção
 - Implantação
- Projete um processo que determina a ordem com que cada uma dessas atividades é executada
- Quais as características positivas ou negativas desse processo?

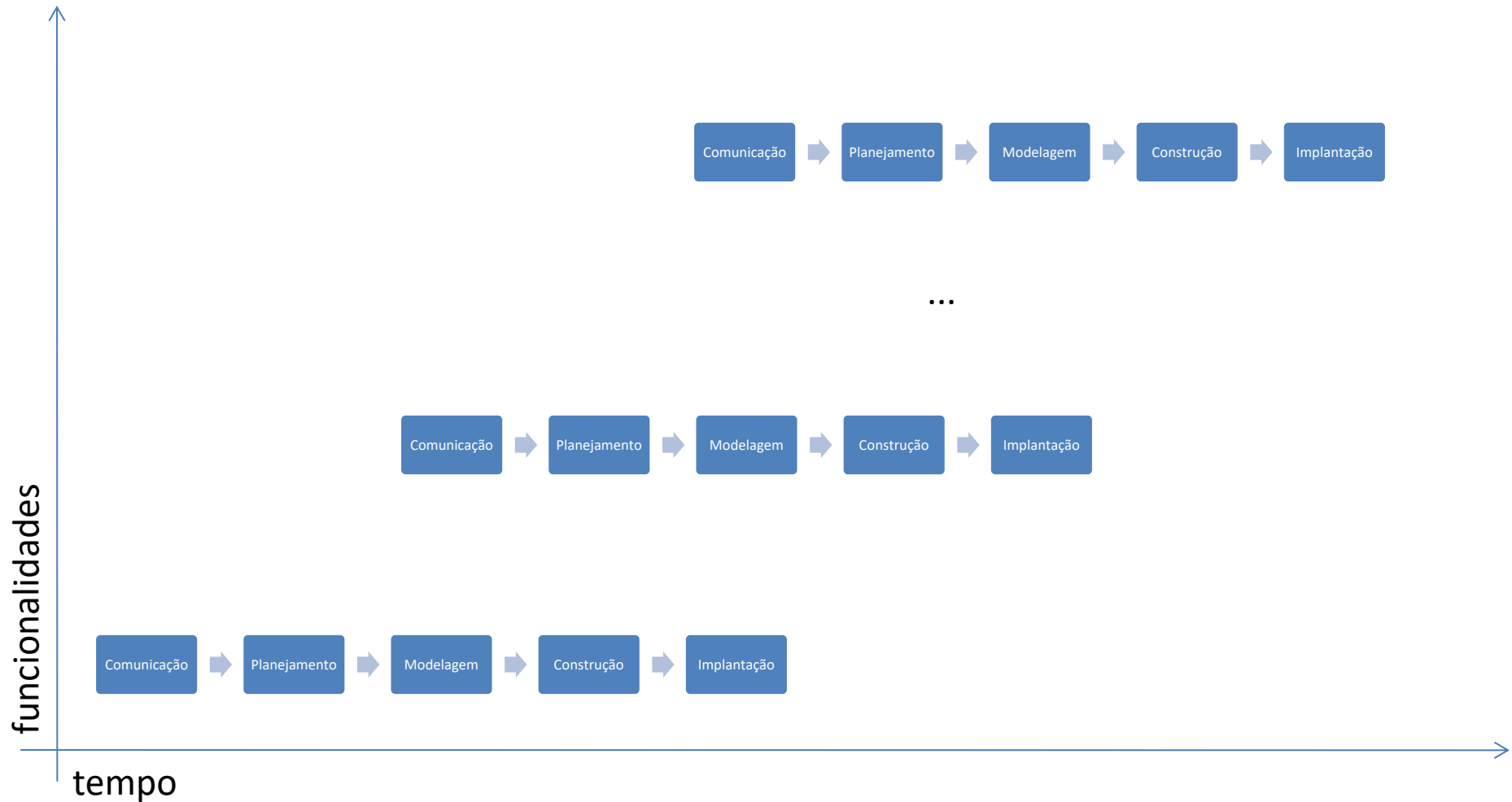
Ciclo de vida Cascata



Ciclo de vida Cascata

- Útil quando se tem requisitos estáveis e bem definidos
 - Ex.: Adicionar um novo dispositivo legal em um sistema de contabilidade
- Não lida bem com incertezas
- Fornece pouca visibilidade do estado do projeto
 - Muito tempo para a primeira entrega
 - Dificuldade na obtenção de feedback do cliente

Ciclo de vida Incremental



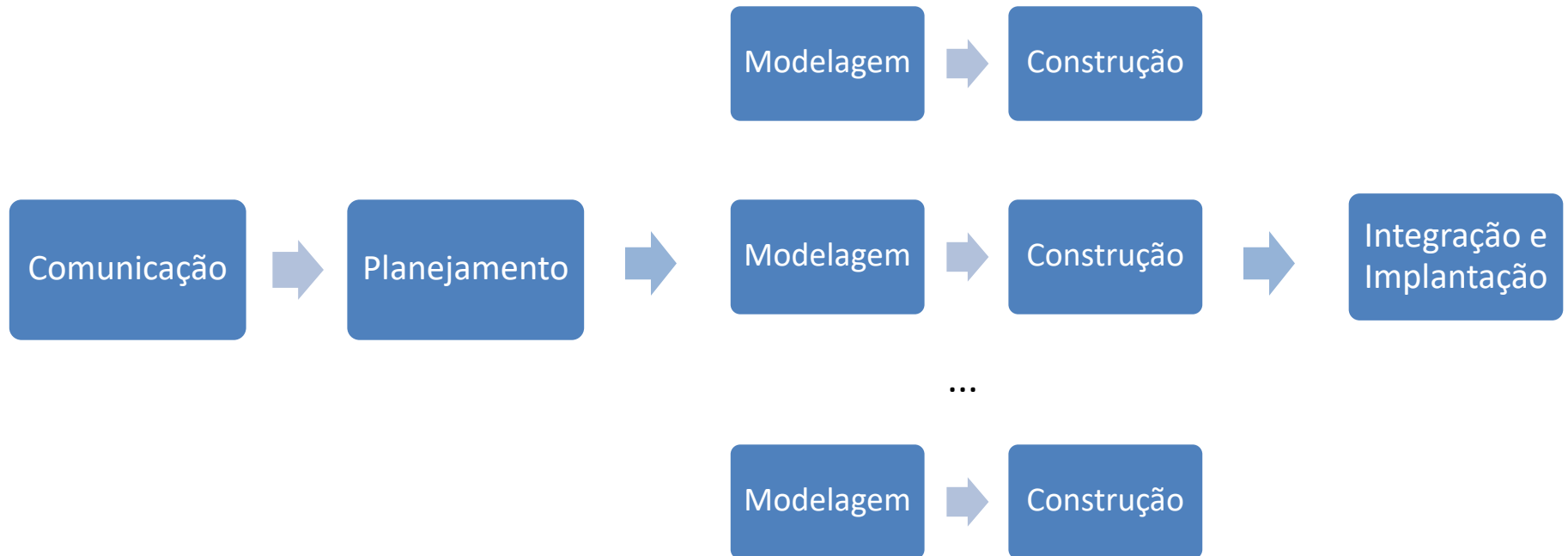
Ciclo de vida Incremental

- Faz entregas incrementais do software
 - Cada incremento é construído via um mini-cascata
 - Cada incremento é um software operacional
- Versões anteriores ajudam a refinar o plano
 - Feedback constante do cliente
- Diminuição da ansiedade do cliente
 - O cliente rapidamente recebe uma versão funcional do software



Jeff Patton (2008)

Ciclo de vida RAD

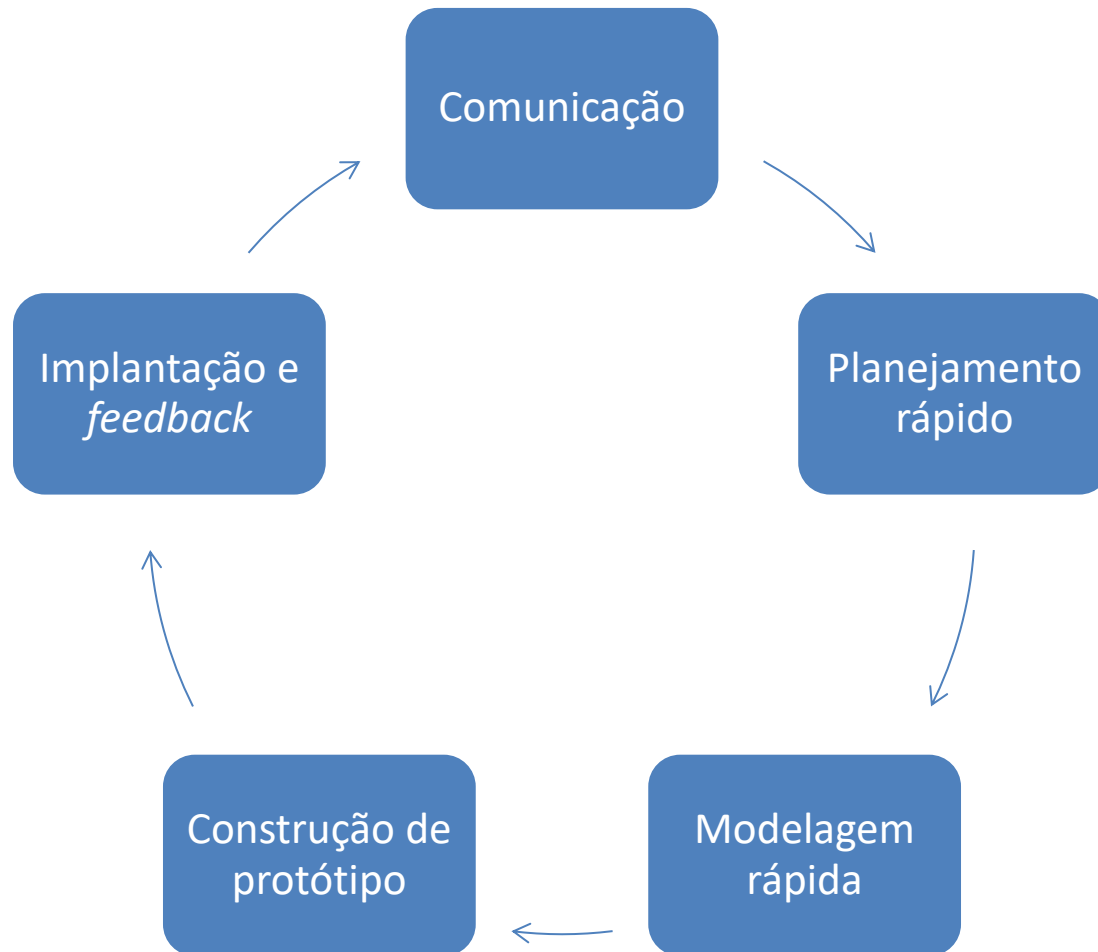


tempo

Ciclo de vida RAD

- Funcionamento equivalente ao cascata
- Principais diferenças
 - Visa entregar o sistema completo em 60 a 90 dias
 - Múltiplas equipes trabalham em paralelo na modelagem e construção
 - Assume a existência de componentes reutilizáveis e geração de código
- Difícil de ser utilizado em domínios novos ou instáveis

Prototipação



Prototipação

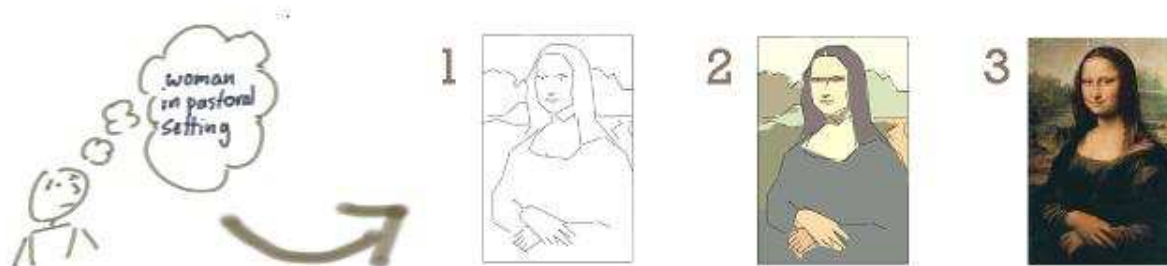
- Usualmente utilizado como auxílio a outro modelo de ciclo de vida
- Útil para
 - Validar um requisito obscuro com o cliente
 - Verificar o desempenho de um algoritmo específico
- Deveria ser jogado fora no final
 - Protótipos não são produtos
 - Usualmente os clientes desejam colocar protótipos em produção

Ciclo de vida Espiral



Ciclo de vida Espiral

- Foco principal no gerenciamento de riscos
- A cada ciclo
 - O conhecimento aumenta
 - O planejamento é refinado
 - O produto gerado no ciclo anterior é evoluído (não é jogado fora)
- Cada ciclo evolui o sistema, mas não necessariamente entrega um software operacional
 - Modelo em papel
 - Protótipo
 - Versões do produto
 - Etc.



Jeff Patton (2008)

Por que fazer bem feito?

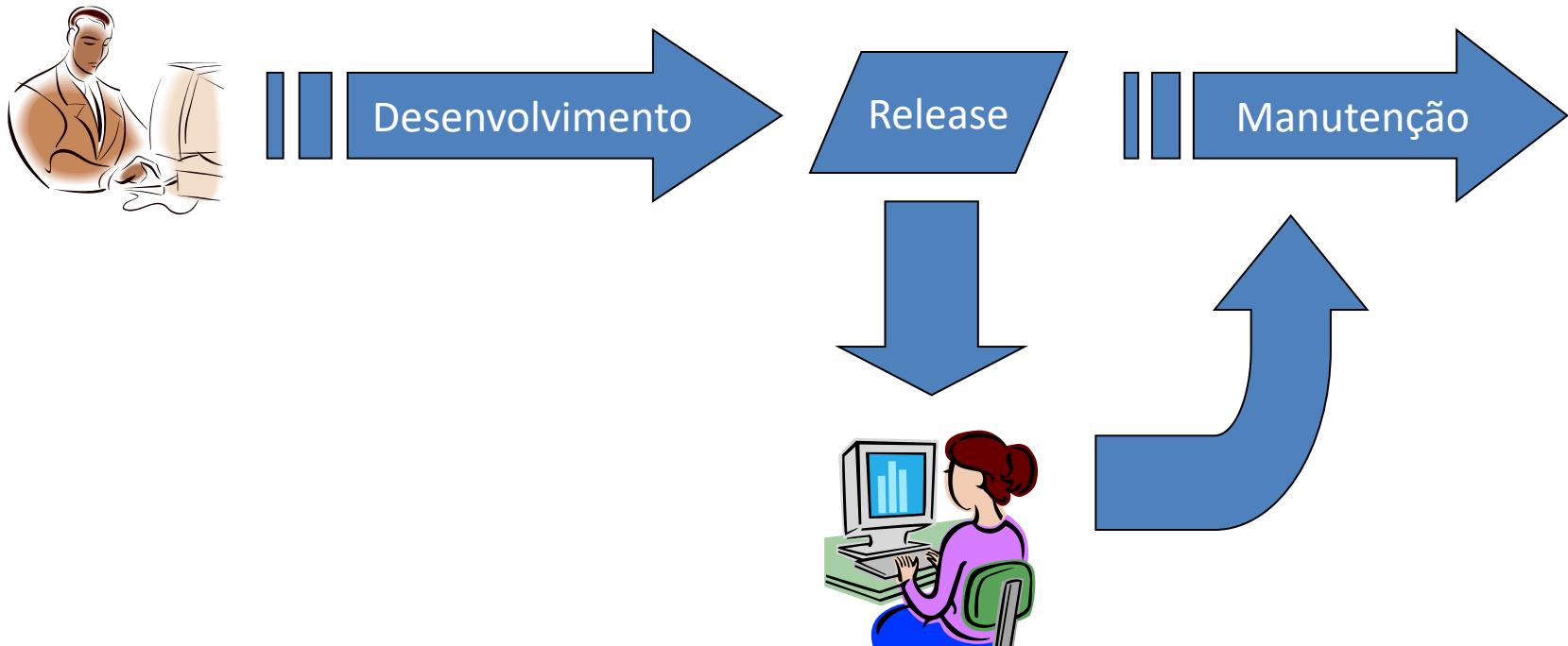
- Por que é mais barato!
 - Historicamente, 60% a 80% do esforço total ocorre na manutenção
- Por que é mais rápido!
 - Não ter tempo para fazer bem feito agora significa ter tempo para refazer depois
- Por que é mais fácil!
 - Desenvolvimento ocorre uma única vez
 - Manutenção é para sempre

O que é a manutenção?

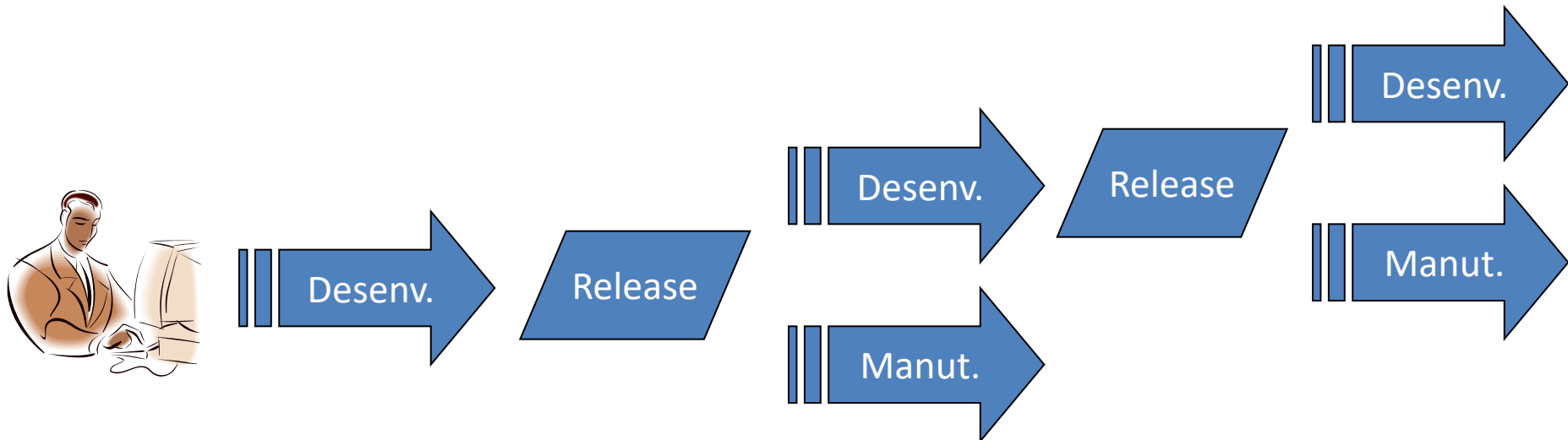
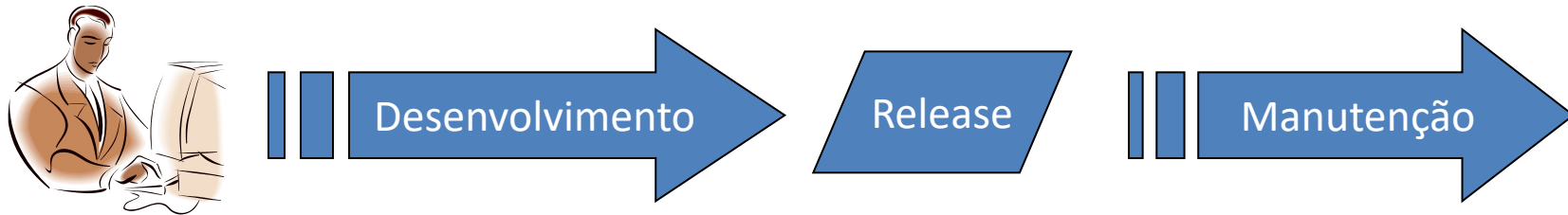
O processo de **modificar um sistema de software** ou componente, **depois da entrega**, para **corrigir falhas, melhorar desempenho ou outros atributos**, ou **adaptar a mudanças no ambiente**.

IEEE Std 620.12 1990

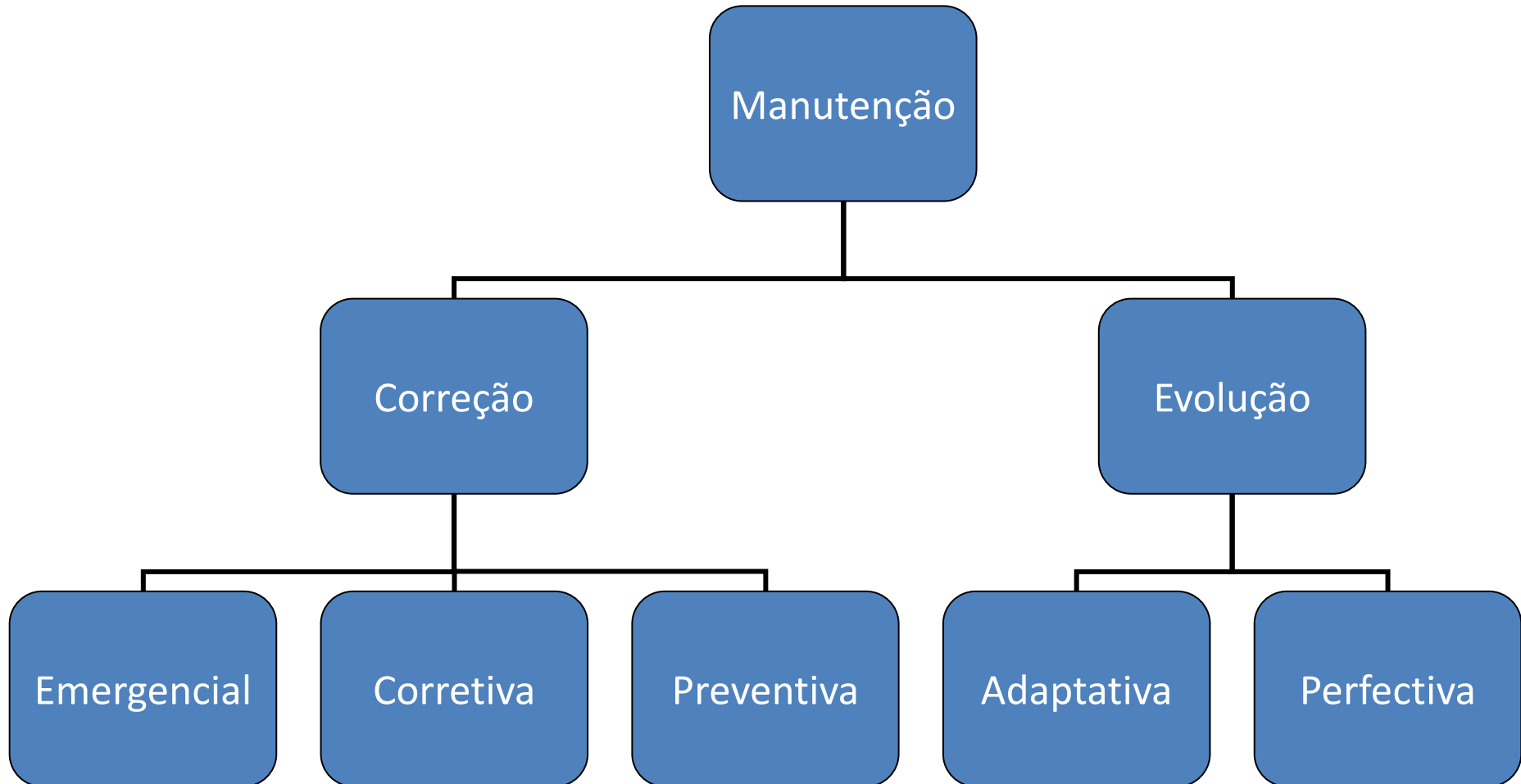
Quando inicia a manutenção?



Quando inicia a manutenção?



Quais são os tipos de manutenção?



Quais são os tipos de manutenção?

- Manutenção emergencial
 - Não programada
 - Mantém temporariamente o sistema funcionando
 - Necessita uma manutenção corretiva posterior
- Manutenção corretiva
 - Reativa
 - Corrige problemas reportados
 - Faz o software voltar a atender aos requisitos

Quais são os tipos de manutenção?

- Manutenção preventiva
 - Pró-ativa
 - Corrige problemas latentes
- Manutenção adaptativa
 - Mantém o software usável após mudanças no ambiente
- Manutenção perfectiva
 - Provê melhorias para o usuário
 - Melhora atributos de qualidade do software

Mitos gerenciais

- Basta um bom livro de ES para fazer bom software
 - Um bom livro certamente ajuda, mas ele precisa refletir as técnicas mais modernas de ES e ser lido!
- Se estivermos com o cronograma atrasado, basta adicionar mais gente ao projeto
 - Adicionar gente a um projeto atrasado faz o projeto atrasar mais!
- Se o projeto for terceirizado, todos os meus problemas estão resolvidos
 - É mais difícil gerenciar projetos terceirizados do que projetos internos!

Mitos do cliente

- Basta dar uma idéia geral do que é necessário no início
 - Requisitos ambíguos normalmente são uma receita para desastre!
 - Comunicação contínua com o cliente é fundamental!
- Modificações podem ser facilmente acomodadas, porque software é flexível
 - O impacto de modificações no software varia em função da modificação e do momento em que ela é requisitada!
 - Comunicação contínua com o cliente é fundamental!

Mitos do desenvolvedor

- Assim que o código for escrito o trabalho termina
 - 60% a 80% do esforço será gasto depois que o código foi escrito!
 - Vale a pena esforçar para chegar a um bom código (boa documentação, bom projeto, etc.)!
- Só é possível verificar a qualidade de um software quando o executável existir
 - Revisões usualmente são mais eficazes que testes, e podem ser utilizadas antes do software estar executável!

Mitos do desenvolvedor

- O único produto a ser entregue em um projeto é o código
 - Além do código, documentações tanto para a manutenção quanto para o uso são fundamentais!
- Engenharia de software gera documentação desnecessária
 - Engenharia de software foca em criar qualidade, e não criar documentos!
 - Algum grau de documentação é necessário para evitar retrabalho!
 - Questione sempre que encontrar um documento desnecessário para o projeto!

7 princípios de Hooker

- Tem que existir uma razão para se fazer software
 - Se não for possível identificar essa razão, é melhor não fazer
 - Fazer software, em última instância, consiste em “agregar valor para o usuário”
 - É importante enxergar os reais requisitos do software!
- Keep it simple, sir! (KISS)
 - “um projeto deve ser o mais simples possível, mas não mais simples que isso”
 - As soluções mais elegantes normalmente são simples
 - Fazer algo simples usualmente demanda mais tempo do que fazer de forma complexa

7 princípios de Hooker

- Mantenha o estilo
 - O projeto de um software deve seguir um único estilo
 - A combinação de diferentes estilos corretos pode levar a um software incorreto
 - Padrões e estilos devem ser estabelecidos no início e seguidos por todos
- O que é produzido por você é consumido por outros
 - Sempre especifique, projete e codifique algo pensando que outros vão ler
 - Sempre exija qualidade nos produtos que você consome e forneça qualidade nos produtos que você produz

7 princípios de Hooker

- Esteja pronto para o futuro
 - Sistemas de boa qualidade têm vida longa
 - Projete desde o início pensando na manutenção
- Planeje para reutilização
 - Pense no problema geral, e não só no problema específico
 - Busque por soluções já existentes
- Pense!
 - “plano é desnecessário, mas planejar é indispensável” – D. Eisenhower
 - Avalie alternativas
 - Mitigue os riscos

Exercício

- Cenário
 - Você deseja abrir uma empresa e lançar no mercado um produto inovador
- Qual ciclo de vida utilizar como base?
- Quais outras atividades de ES você incorporaria nesse processo?
- Quais são os maiores riscos que você está se expondo?

Nós precisamos de ES!



Como o cliente explicou



Como o líder de projeto entendeu



Como o analista planejou



Como o programador codificou



O que os beta testers receberam



Como o consultor de negocios descreveu



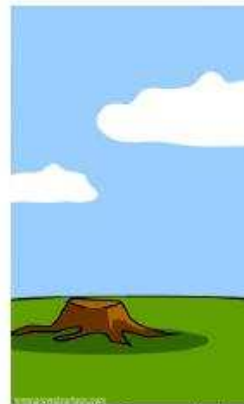
Valor que o cliente pagou



Como o projeto foi documentado



O que a assistencia tecnica instalou



Como foi suportado



Quando foi entregue



O que o cliente realmente necessitava

Introdução à ES

