Engenharia de Software

Processos

Baseado nos slides do Prof Marco Túlio (Engenharia Moderna)

Mas o que é Processo de Software?

- Um conjunto estruturado de atividades necessárias para o desenvolvimento de um sistema de software
  - Especificação
  - Projeto
  - Validação
  - Evolução
- Um processo de software pode ser definido como um conjunto coerente de atividades, políticas, estruturas organizacionais, tecnologias, procedimentos e artefatos necessários para conceber, desenvolver, dispor e manter um produto de software (FUGGETTA, 2000)

O que é Processo de Software?

- Um processo eficaz deve, claramente, considerar as relações entre as **atividades**, os **artefatos** produzidos no desenvolvimento, as **ferramentas** e os **procedimentos** necessários e a habilidade, o **treinamento** e a **motivação** do pessoal envolvido (FALBO, 1998)

- Uma **sequência** de **passos** requeridos para realizar uma tarefa (procedimento médico, operação militar, ou o desenvolvimento/ manutenção de um software), **com o objetivo de auxiliar os envolvidos na execução das tarefas e na realização dos trabalhos de forma coordenada**, com cada um auxiliando o outro (HUMPHERY,2005)

Descrições de Processos de Software

- Quando descrevemos e discutimos processos, geralmente falamos sobre as atividades desses processos (tais como especificação, desenvolvimento de interface de usuário, etc.) e organização dessas atividades

- Descrições de processos também podem incluir
  - Produtos, que são os resultados de uma atividade do processo
  - Papéis, que refletem as responsabilidades das pessoas envolvidas no processo
  - Pré e pós-condições, que são declarações que são verdadeiras

Modelos de Processo de Software

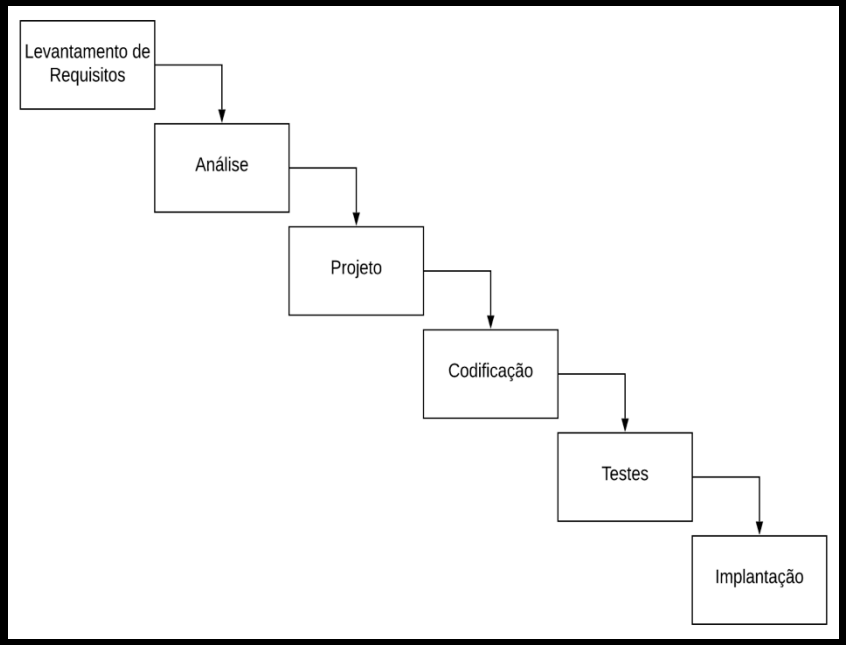
Características básicas comuns aos modelos

- Descrever principais fases do desenvolvimento
- Definir principais atividades a serem realizadas durante cada uma das fases
- Especificar produtos de cada uma das fases e insumos para o início das fases
- Fornecer um *framework* sobre o qual as atividades necessárias podem ser mapeadas

Como é na Engenharia Tradicional?

- Civil, mecânica, elétrica, aviação, automobilística, etc
- Projeto com duas características:
  - Planejamento detalhado
  - Sequencial
- Isto é: Waterfall
  - Há milhares de anos

Natural que ES começasse usando Waterfall



No entanto: Waterfall não funcionou com software!

Software é diferente

- Engenharia de Software ≠ Engenharia Tradicional
- Software ≠ (carro, ponte, casa, avião, celular, etc)
- Software ≠ (produtos físicos)
- Software é abstrato e "adaptável"

Dificuldade 1: Requisitos

- Clientes não sabem o que querem (em um software)
  - Funcionalidades são "infinitas" (difícil prever)
  - Mundo muda!
- Não dá mais para ficar 1 ano levantando requisitos, 1 ano projetando, 1 ano implementando, etc
- Quando o software ficar pronto,

  ele estará obsoleto!

Dificuldade 2: Documentações Detalhadas

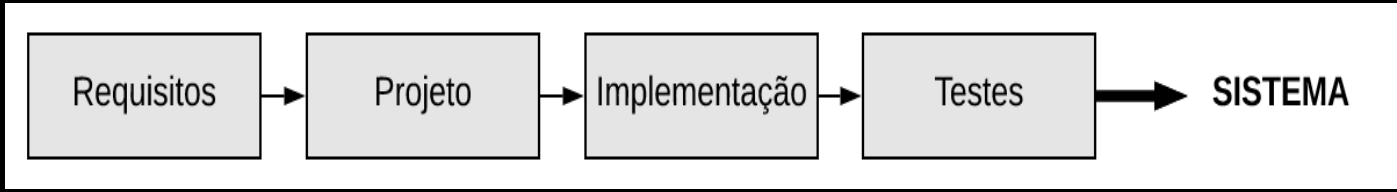
- Verbosas e pouco úteis
- Na prática, desconsideradas durante implementação
- *Plan-and-document* não funcionou

com software

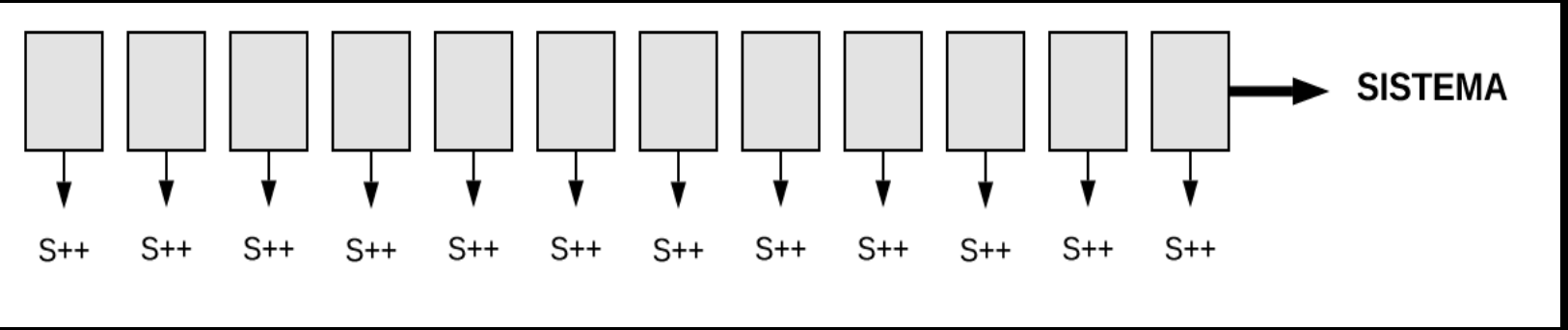
Manifesto Ágil (2001)

Ideia central: desenvolvimento iterativo

Waterfall



Ágil



Desenvolvimento iterativo

- Suponha um sistema imenso, complexo etc
  - Qual o menor "incremento de sistema" que eu consigo implementar em 15 dias e validar com o usuário?
- Validar é muito importante!
  - Cliente não sabe o que quer!

Reforçando: ágil = iterativo

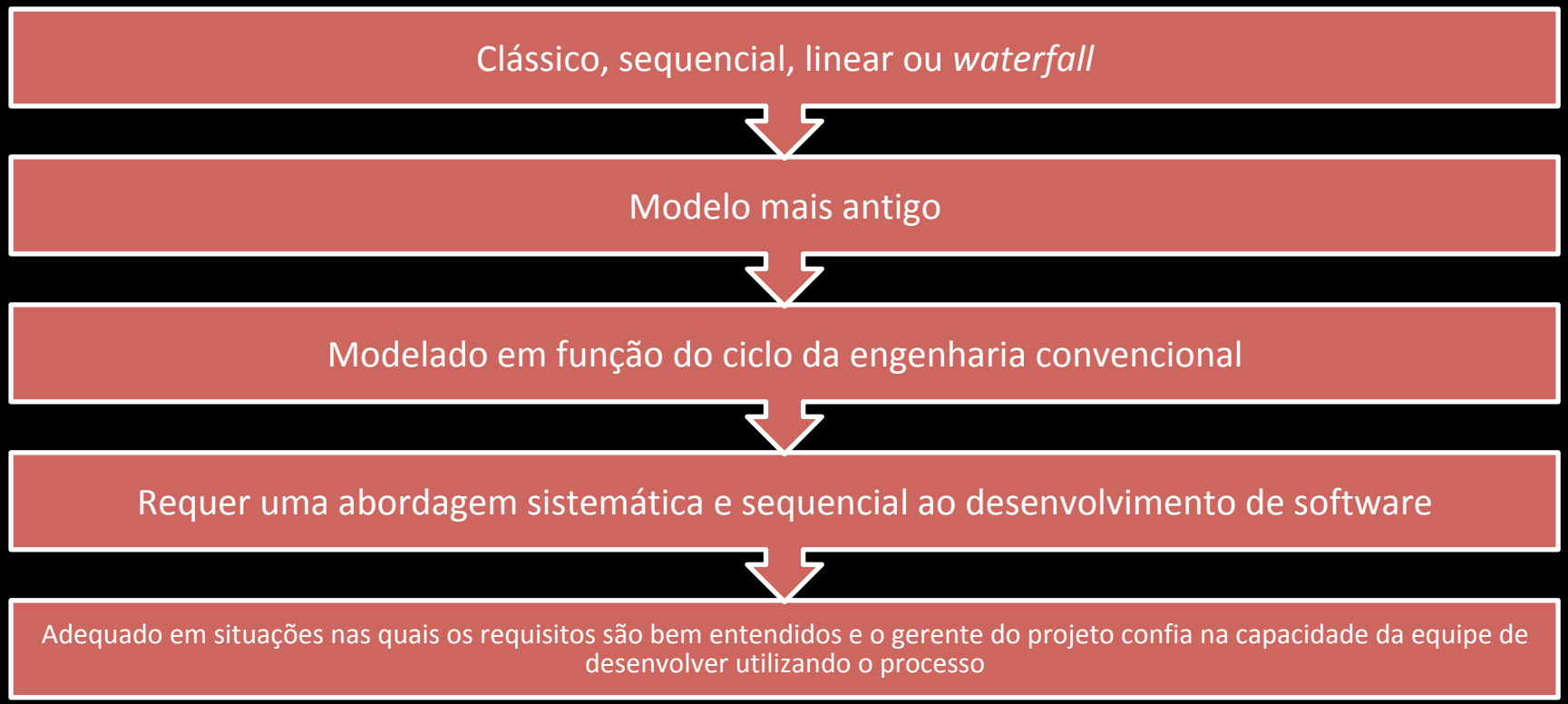
Outros pontos importantes (1)

- Menor ênfase em documentação
- Envolvimento constante do cliente (*product owner*)

Outros pontos importantes (2)

- Novas práticas de programação
  - Testes, refactoring, integração contínua, etc

Modelo Cascata



Modelo Cascata - Vantagens

- Fase única de requisitos leva à especificação antes do projeto e ao projeto antes da codificação
- Cada passo serve como uma base aprovada e documentada para o passo seguinte
- Toda fase do projeto pode ser cuidadosamente planejada
- Responsabilidades podem ser claramente definidas
- Existência de marcos ao fim de cada fase

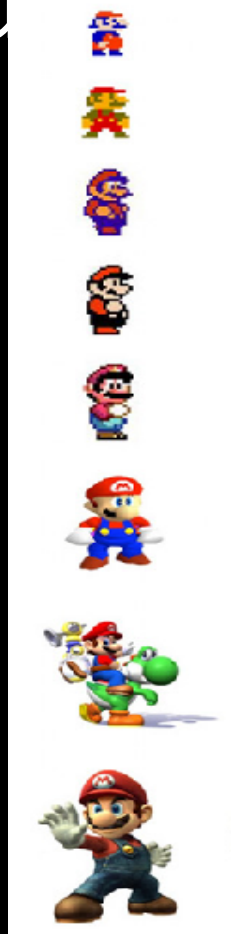
Modelo Cascata - Problemas

Dificuldade de acomodação das mudanças depois que o processo está em andamento. Uma fase tem de estar completa antes de passar para a próxima.

- O fluxo sequencial geralmente não é seguido em projetos reais
- Requisitos devem ser estabelecidos de maneira completa, correta e clara no início do projeto
- Dependência de requisitos corretos e estáveis
- O usuário (cliente) muitas vezes não consegue explicar todos os requisitos inicialmente
- Aplicação deve ser entendida pelo desenvolvedor desde o início do projeto
- Difícil avaliar o progresso verdadeiro do projeto durante as primeiras fases
- Uma versão executável do software só fica disponível numa etapa avançada do desenvolvimento
- Grande esforço de integração e testes no final do projeto

Desenvolvimento Evolucionário ou Evolutivo

- Versões parciais são desenvolvidas para atendimento aos requisitos conhecidos inicialmente
- Primeira versão utilizada para refinar os requisitos de uma segunda versão
- A partir do conhecimento sobre os requisitos, obtido com o uso, continua-se o desenvolvimento, evoluindo o produto
- Modelo baseado em
  - Versão inicial de implementação
  - Entrega de implementações intermediárias
  - Verificações constantes pelo usuário até entrega de uma versão final



Desenvolvimento Evolucionário - Vantagens

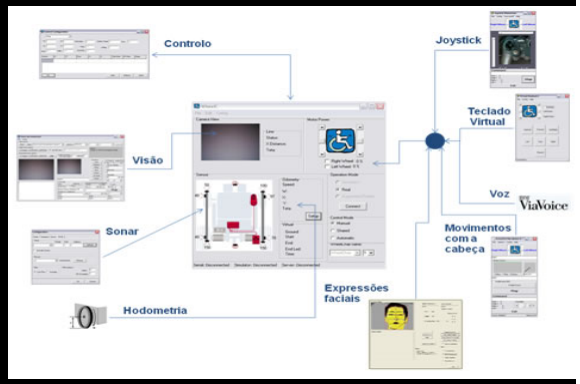
- Adequado quando os requisitos não podem ser completamente especificados de início
- O uso do sistema pode aumentar o conhecimento sobre o produto e melhorar os requisitos
- Produz sistemas que atendem às necessidades imediatas do cliente

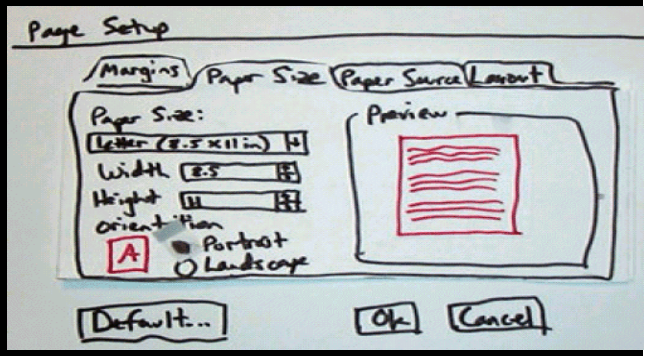
Desenvolvimento Evolucionário - Desvantagens

- Necessária uma forte gerência de custo, cronograma e configuração
- Usuários podem não entender a natureza da abordagem e se decepcionar quando os resultados são não satisfatórios
- O processo não é visível
  - Não se produzem documentos que reflitam as versões do sistema
- Frequentemente são mal estruturados
  - Software sem estrutura
  - Modificações cada vez mais custosas

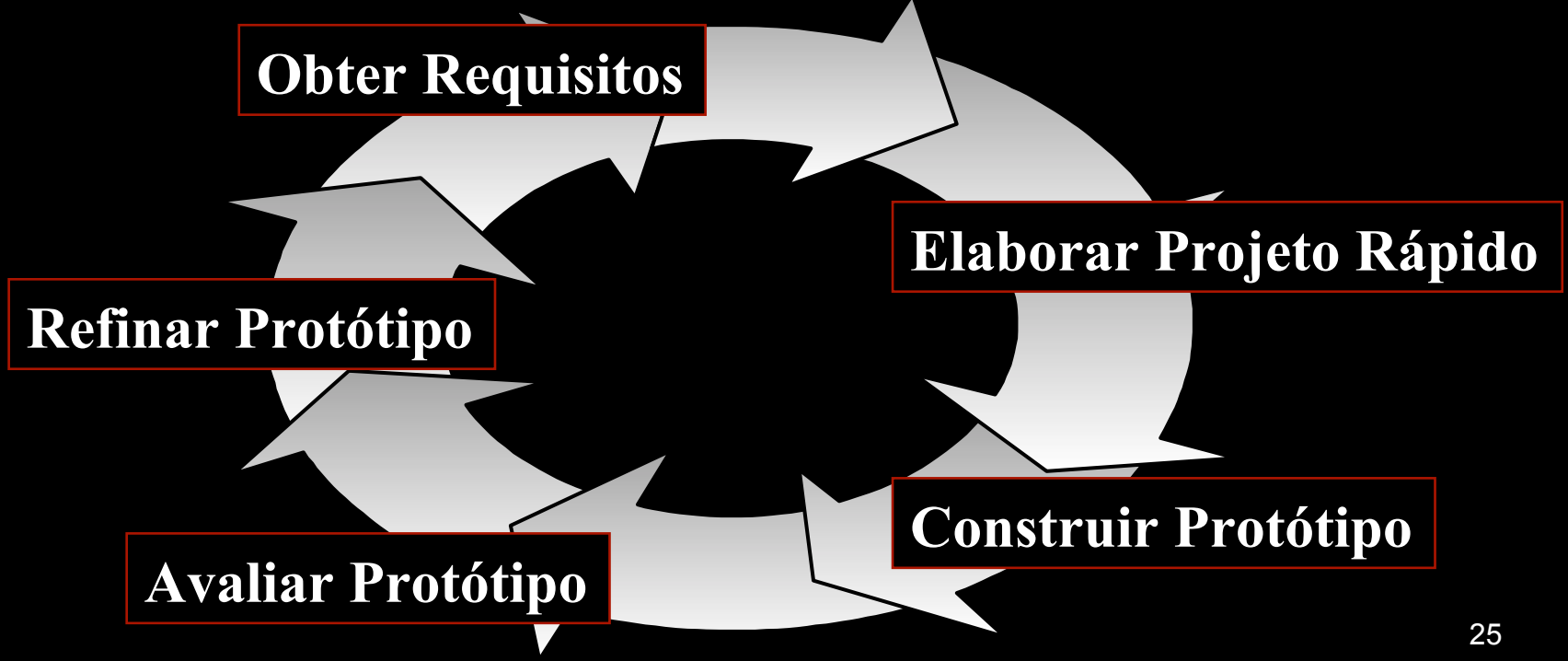
Prototipação

- Protótipos podem ser utilizados para
  - Explorar requisitos que serão implementados posteriormente em um incremento funcional
  - Determinar a viabilidade técnica, de custo e de cronograma para o projeto
- Objetivo
  - Entender os requisitos do sistema
- O protótipo concentra-se na verificação dos pontos pouco entendidos do sistema

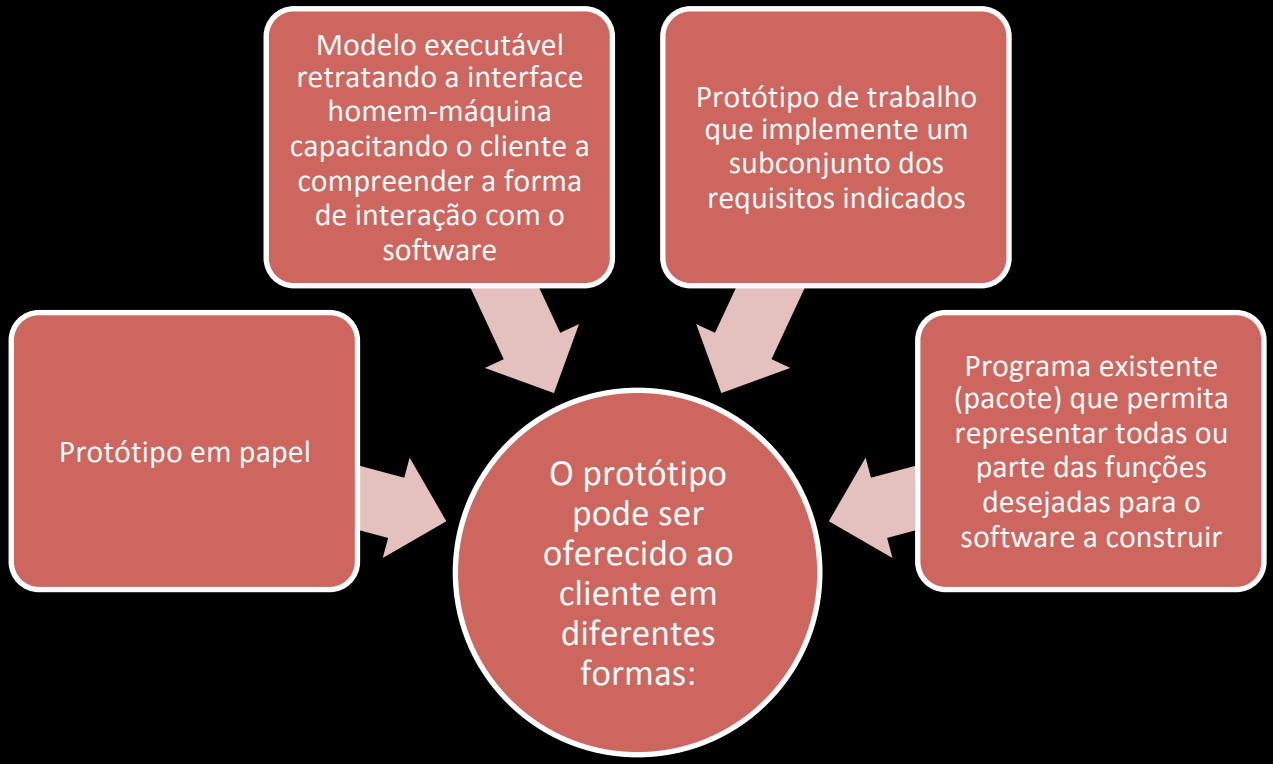




Prototipação



Prototipação



Prototipação - Vantagens

Um protótipo deve ser submetido a uma avaliação

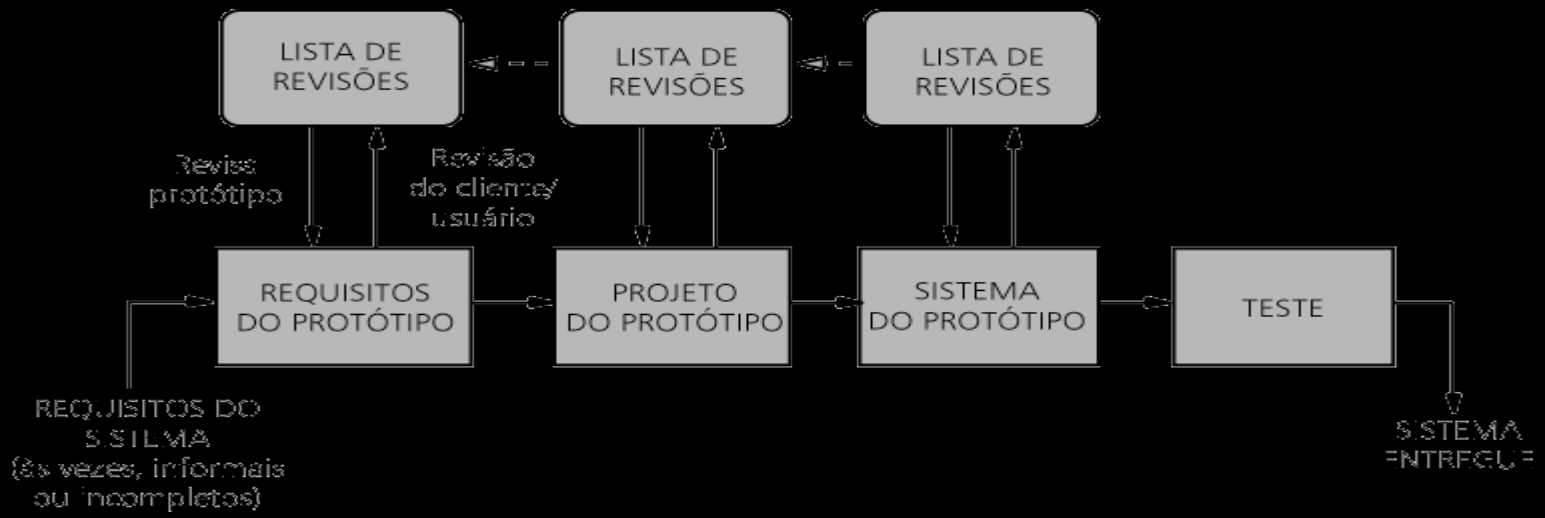
- Geralmente realizada pelo cliente

- O fato de o cliente poder interagir com um protótipo ajuda a balancear suas necessidades funcionais e de desempenho
- Os desenvolvedores podem implementar os requisitos baseado no *feedback* do usuário
- Modelo de desenvolvimento interessante para alguns sistemas de grande porte que representem um certo grau de dificuldade para exprimir rigorosamente os requisitos
- A experiência adquirida no desenvolvimento do protótipo vai ser de extrema utilidade nas etapas posteriores do desenvolvimento do sistema real, permitindo reduzir custo e resultando num sistema melhor concebido

Prototipação - Desvantagens

- Clientes imaginam que a maior parte do trabalho já foi feita
- Protótipo pode crescer de maneira não planejada, se tornando um incremento funcional
- Protótipo pode ter um desempenho melhor do que um incremento funcional, pois não implementa toda a funcionalidade, causando frustração aos clientes quando o sistema completo é entregue
- Quando informamos que o produto precisa ser reconstruído, o cliente exige que alguns acertos sejam aplicados para tornar o protótipo um produto, e muito freqüentemente, a gerência de desenvolvimento de software cede
- O desenvolvedor muitas vezes faz concessões de implementação a fim de colocar um protótipo em funcionamento rapidamente
  - A opção menos ideal se tornou então parte integrante do sistema

Prototipação



Iteração de processo

- Requisitos de sistema SEMPRE evoluem no curso de um projeto e, sendo assim, a iteração de processo, onde estágios iniciais são retrabalhados, é sempre parte do processo dos sistemas de grande porte
- A iteração pode ser aplicada a qualquer um dos modelos genéricos do processo

Iteração de processo

- O sistema de software é desenvolvido em vários passos similares
- Idéia de melhorar (ou refinar) pouco a pouco o sistema (iterações)
- Em cada iteração a equipe de desenvolvimento
  - Identifica e especifica os requisitos relevantes
  - Cria um projeto utilizando a arquitetura escolhida como guia
  - Implementa o projeto em componentes
  - Verifica se esses componentes satisfazem os requisitos
- Se uma iteração atinge os seus objetivos, o desenvolvimento prossegue com a próxima iteração
- Caso contrário a equipe deve rever as suas decisões e tentar uma nova abordagem
- Conclusão
  - O âmbito do sistema não é alterado
  - Seu detalhe vai aumentando em iterações sucessivas

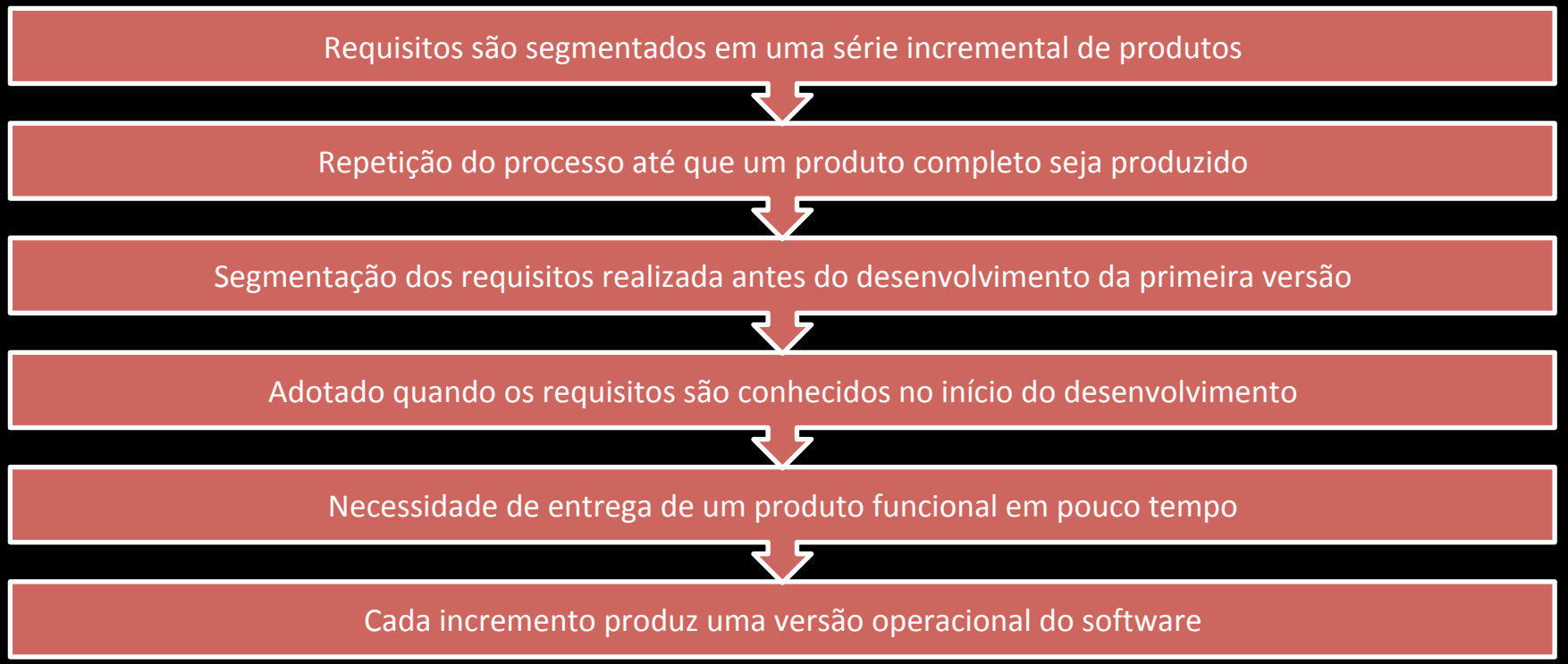
Entrega incremental

- Em cada passo, o sistema é estendido com mais funcionalidades
- Idéia de aumentar pouco-a-pouco o âmbito do sistema
- Um incremento não é necessariamente a adição do código executável correspondente aos casos de uso que pertencem à iteração em andamento
- Especialmente nas primeiras fases do ciclo de desenvolvimento, os desenvolvedores podem substituir um projeto superficial por um mais detalhado ou sofisticado
- Em fases avançadas os incrementos são tipicamente aditivos

Entrega incremental

- Ao invés de entregar o sistema como uma única entrega, o desenvolvimento e a entrega são separados em incrementos, sendo que cada incremento fornece parte da funcionalidade solicitada
- Os requisitos de usuário são priorizados e os requisitos de prioridade mais alta são incluídos nos incrementos iniciais
- Uma vez que o desenvolvimento de um incremento é iniciado, os requisitos são congelados, embora os requisitos para os incrementos posteriores possam continuar evoluindo

Modelo incremental



Entrega incremental

- Exemplo
  - O desenvolvimento de um produto comercial de software é uma grande tarefa que pode ser estendida por vários meses/anos,...
  - É mais fácil dividir o trabalho em partes menores (iterações) tendo cada uma como resultado um incremento (processo incremental)
  - Assim a equipe envolvida pode refinar e melhorar gradativamente a qualidade e os detalhes do sistema

Desenvolvimento incremental

- **Vantagens**
- O valor pode ser entregue para o cliente com cada incremento e, desse modo, a funcionalidade de sistema é disponibilizada mais cedo
  - Menor custo e menos tempo são necessários para se entregar a primeira versão
- O incremento inicial age como um protótipo para auxiliar a elicitar os requisitos para incrementos posteriores do sistema
- Riscos menores de falha geral do projeto.
- Os serviços de sistema de mais alta prioridade tendem a receber mais testes

Desenvolvimento incremental

- **Desvantagens**
- Se os requisitos não são tão estáveis ou completos quanto se esperava, alguns incrementos podem precisar ser retirados de uso e retrabalhados

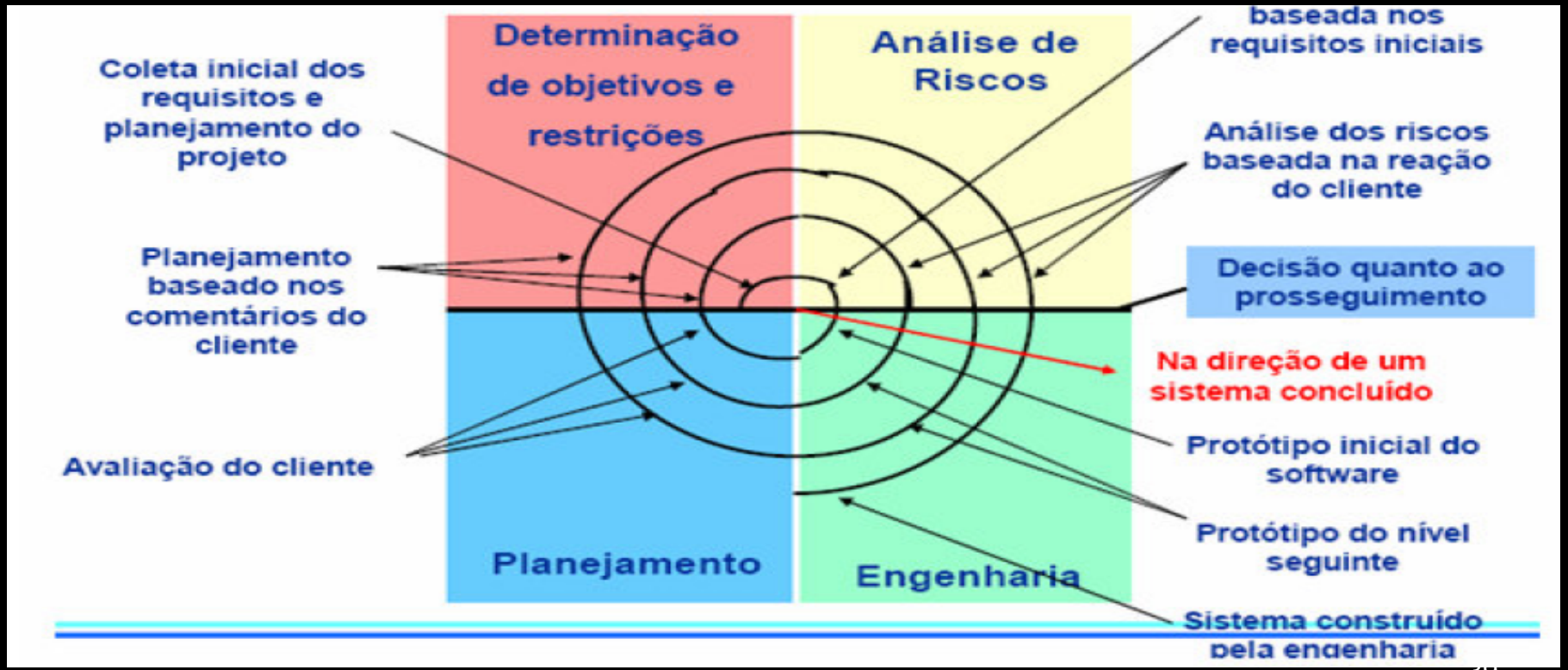
- Retrabalho

- O gerenciamento de custo, cronograma é mais complexo

Modelo Espiral

- O processo é representado como uma espiral ao invés de uma sequência de atividades
- Cada *loop* na espiral representa uma fase no processo
- Os riscos são explicitamente avaliados e resolvidos ao longo do processo.

Modelo Espiral



Modelo Espiral

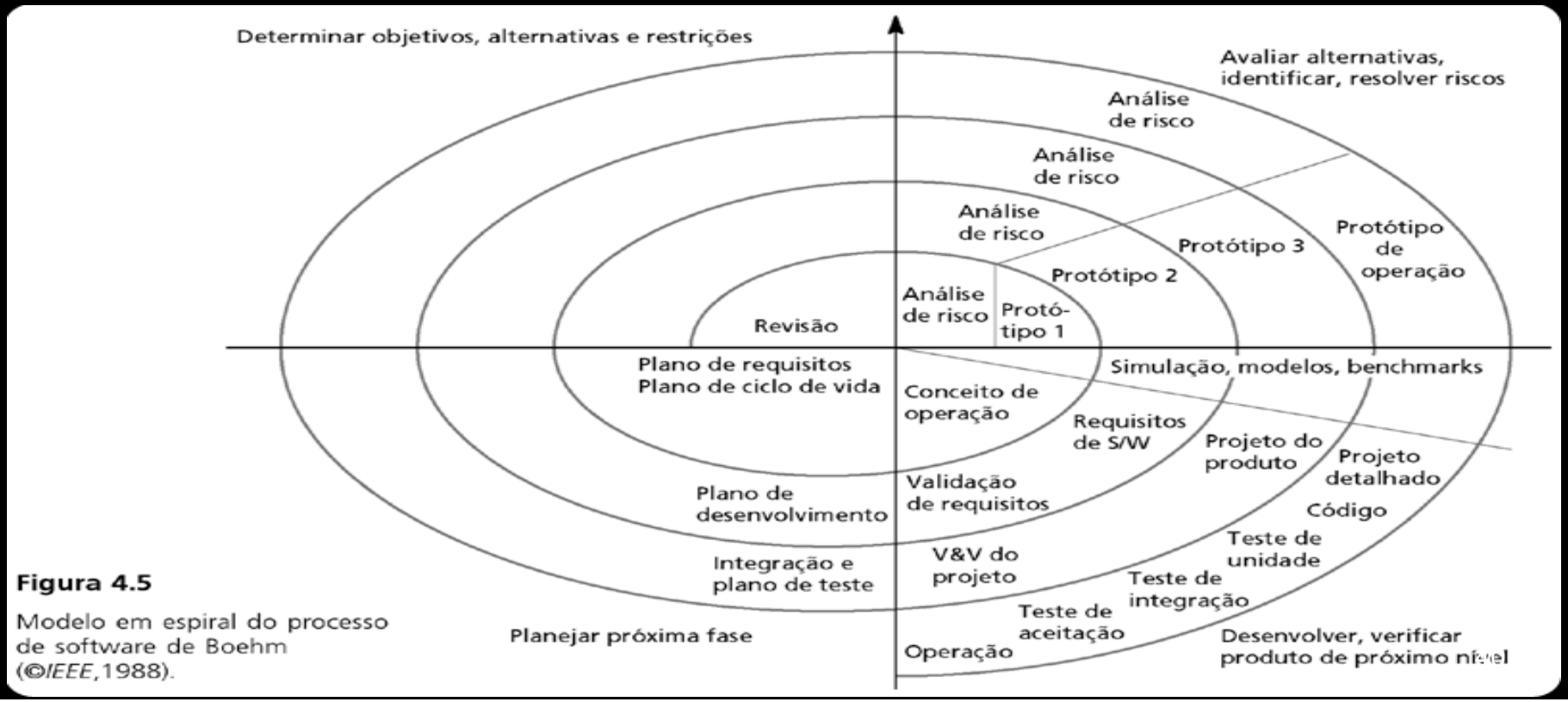


Figura 4.5

Modelo em espiral do processo de software de Boehm (©*IEEE*, 1988).

Modelo Espiral - Setores

- Definição de objetivos
  - Objetivos específicos para a fase são identificados.
- Avaliação e redução de riscos
  - Riscos são avaliados e atividades são realizadas para reduzir os riscos-chave.
- Desenvolvimento e validação
  - Um modelo de desenvolvimento para o sistema, que pode ser qualquer um dos modelos genéricos, é escolhido.
- Planejamento/Avaliação
  - O projeto é revisado e a próxima fase da espiral é planejada

Modelo Espiral - Vantagens

- O modelo em espiral permite que ao longo de cada iteração se obtenham versões do sistema cada vez mais completas, recorrendo à prototipagem para reduzir os riscos
- Este tipo de modelo permite a abordagem do refinamento seguido pelo modelo em cascata, mas que incorpora um enquadramento iterativo que reflete, de uma forma bastante realística, o processo de desenvolvimento

Modelo Espiral - Desvantagens

- Pode ser difícil convencer grandes clientes de que a abordagem evolutiva é controlável
- A abordagem deste tipo de modelo exige considerável experiência na avaliação dos riscos
- Gerenciamento de custo e prazo do projeto complexo

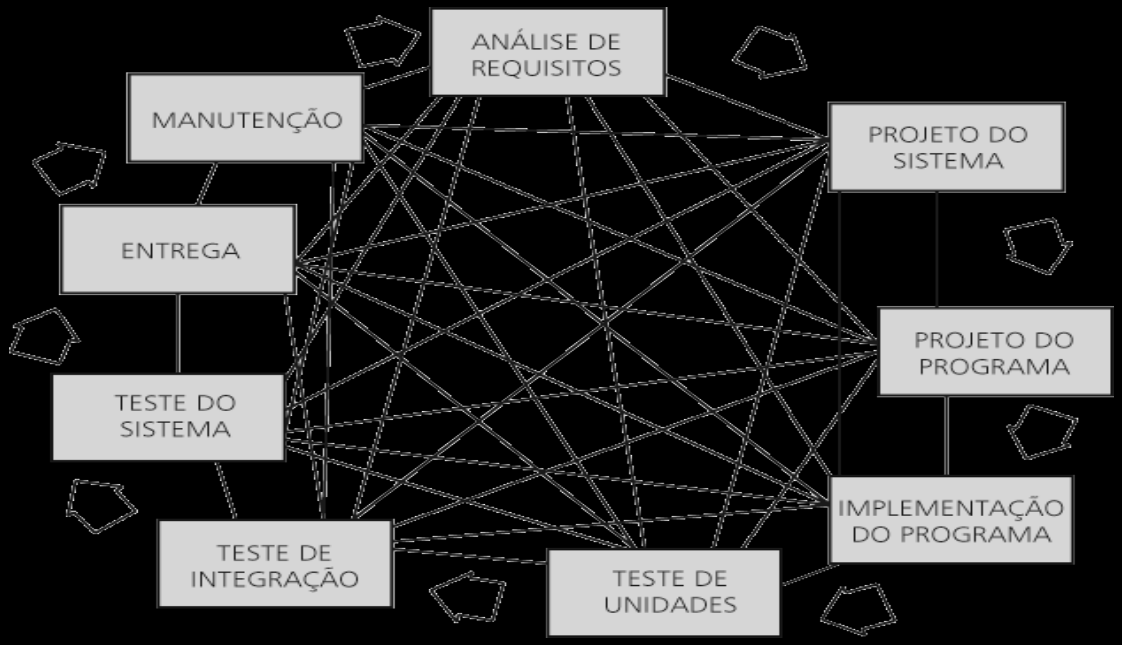
Benefícios da definição e utilização de modelos de processos

- A descrição e utilização de modelos de processos de software resultam em diversos benefícios às organizações de desenvolvimento de software
  - Permite que o conhecimento e experiências sobre um processo sejam organizados e consequentemente incorporados ao processo ou reaproveitadas em outro processo
  - Auxilia na tomada de decisão em situações imprevistas
  - Torna possível a identificação de tarefas críticas
  - Atribui maior visibilidade aos processos
  - Forma uma base para a elaboração de guias, scripts, manuais
  - Auxilia no gerenciamento do projeto (organização, planejamento, alocação de recursos, estimativas etc)

Benefícios da definição e utilização de modelos de processos

- A descrição e utilização de modelos de processos de software resultam em diversos benefícios às organizações de desenvolvimento de software
  - Estabelece bases que podem ser avaliadas para análise e mudanças para melhoria do processo de software;
  - Possibilita que novas tecnologias sejam avaliadas e incorporadas ao processo de software
  - Habilita a empresa na definição de programas de mensuração

Como funciona este modelo?



Pergunta...

- Você considera que uma organização deve adotar um único modelo de processo para o desenvolvimento de todo o software?

Caso 1

- Objetivo: desenvolver um sistema para uma aplicação de comércio eletrônico. Apesar do cliente ter uma certa urgência em colocar o sistema em operação, os requisitos para o mesmo não se encontram bem definidos. O cliente se comprometeu em acompanhar o desenvolvimento. Porém, este possui dificuldades em expressar os requisitos do sistema.

Caso 1

- Objetivo: desenvolver um sistema para uma aplicação de comércio eletrônico. Apesar do cliente ter uma certa urgência em colocar o sistema em operação, os requisitos para o mesmo não se encontram bem definidos. O cliente se comprometeu em acompanhar o desenvolvimento. Porém, este possui dificuldades em expressar os requisitos do sistema.

- Possibilidades: Cascata, Evolutivo, Espiral, Incremental, Prototipação.

Caso 2

- Objetivo:desenvolver um sistema de cadastro de usuários de uma biblioteca virtual. Os requisitos para o sistema foram fornecidos pelo usuário de antemão e estão bem definidos. A organização dispõe de uma quantidade adequada de desenvolvedores experientes no domínio da aplicação. Porém, há uma alta disputa interna entre a equipe de desenvolvimento.

Caso 2

- Objetivo:desenvolver um sistema de cadastro de usuários de uma biblioteca virtual. Os requisitos para o sistema foram fornecidos pelo usuário de antemão e estão bem definidos. A organização dispõe de uma quantidade adequada de desenvolvedores experientes no domínio da aplicação. Porém, há uma alta disputa interna entre a equipe de desenvolvimento.

- Possibilidades: Cascata, Evolutivo, Espiral, Incremental, Prototipação.