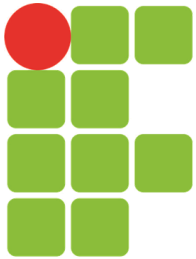


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO SUL
Campus Feliz

# Processos de Desenvolvimento de Software

Profa. Ana Paula Lemke

Última atualização em 21/03/2019.

# Pergunta

Por que não simplesmente codificar?


Windows
A fatal exception 0E has occurred at 0028:C0011E36 in VXD VMM(01) +
00010E36. The current application will be terminated.
* Press any key to terminate the current application.
* Press CTRL+ALT+DEL again to restart your computer. You will
lose any unsaved information in all applications.
Press any key to continue _

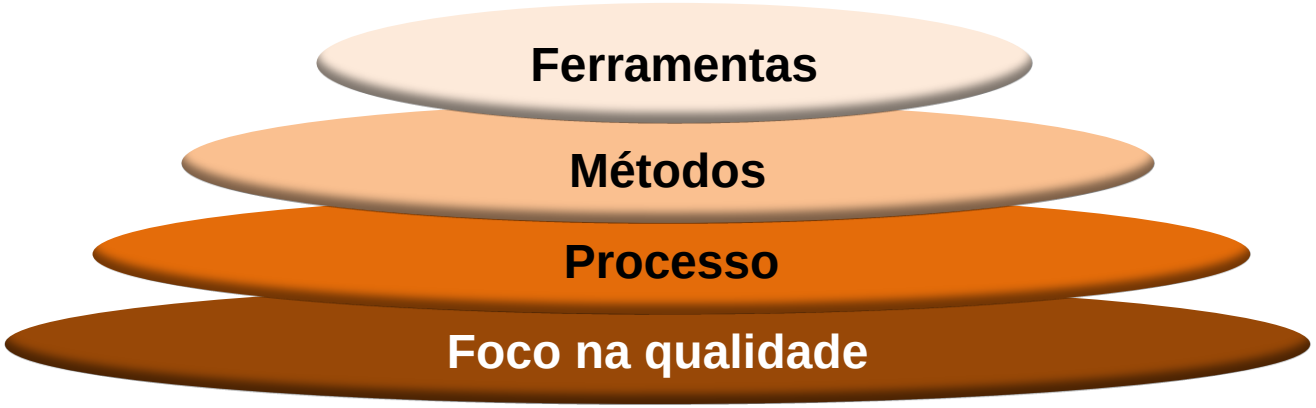
# Programação

- *Ad-hoc*
- Sem planejamento
- Sistemas pequenos
  - Pouco tempo
  - Baixo custo
- Poucas pessoas envolvidas
  - Geralmente um líder resolve tudo
- Não se preocupa com padronização
  - Muito menos com documentação
- Pouco impacto ao seu entorno

# Engenharia de Software

- Planejamento meticuloso
- Sistemas grandes
  - Alto custo
  - Muito tempo
- Desenvolvido por um time
  - Profissionais de diversos perfis
  - Decisão colegiada
- Necessidade de levar em conta padronização existente
  - Documentação
- Grande impacto ao seu entorno

# Engenharia de Software




Fonte: PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 7. ed. 2011. (pág. 39).

# Definições

- Processo de *software*
  - "Conjunto de atividades relacionadas que levam à produção de um produto de *software*". Sommerville, I. "Engenharia de Software", 9º ed, 2011 (pág. 18).
    - A meta do processo pode ser o desenvolvimento de um novo produto ou a evolução de um produto existente (extensão e modificação de um produto existente).
  - "Conjunto de atividades, ações e tarefas realizadas na criação de algum produto de trabalho". PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 7. ed. 2011. (pág. 37).

# Definições

- Modelos de ciclo de vida de *software* (ou modelos de processo de software)
  - Descrição simplificada de um processo de software, que é apresentada a partir de uma perspectiva específica.

# Atividades fundamentais para ES

**Especificação de software:** planejamento, definição dos requisitos, construção de protótipos, ...

**Projeto e Implementação de software**: implementação e teste unitário para atender a especificação

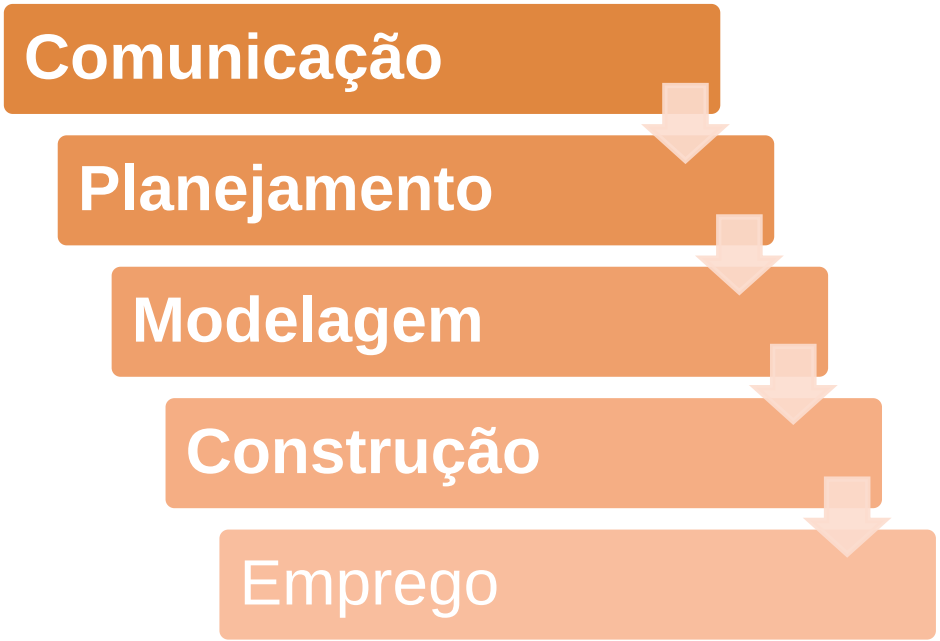
**Validação de software**: garantir que o software satisfaz o que o cliente deseja

**Evolução de software**: para atender as necessidades mutáveis dos clientes

Baseado em Sommervile, I. “Engenharia de Software”, 9º ed, 2011 (pág. 18).

# Atividades fundamentais para ES

- “Metodologia de processo”* genérica para ES, de Pressman:




* Alicerce para um processo de ES completo, que identifica um pequeno número de atividades estruturais aplicáveis a todos os processos de software.

Fonte: PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 7. ed. 2011. (pp. 40-41).

# Atividades fundamentais para ES

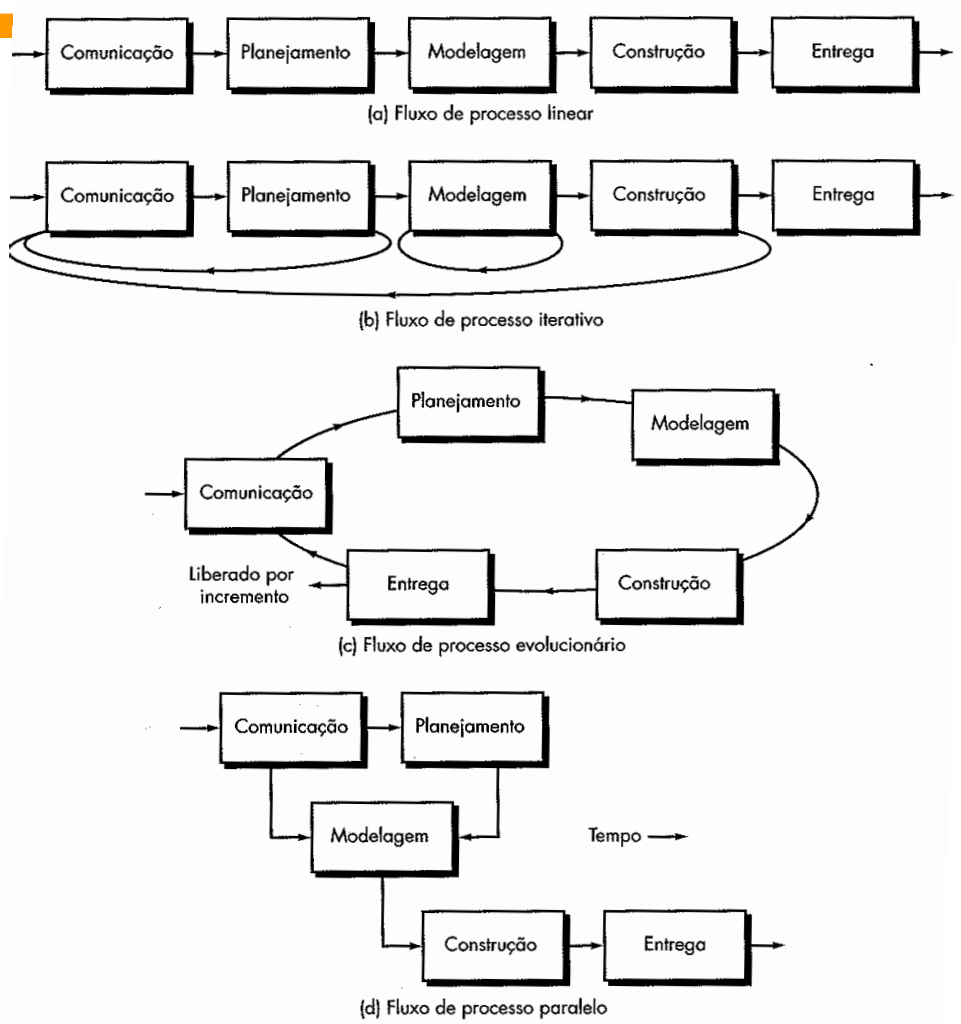
- "Metodologia de processo" genérica para ES, de Pressman:
  - **Comunicação**: envolve compreender os objetivos das partes interessadas e fazer o levantamento das necessidades do software.
  - **Planejamento**: consiste na criação de um plano de projeto de software, com as tarefas técnicas, riscos prováveis, recursos necessários, produtos resultantes e cronograma de trabalho.
  - **Modelagem**: envolve o desenvolvimento de modelos para esboçar o software.
  - **Construção**: combina geração de código e testes.
  - **Emprego**: envolve a entrega do software ao cliente, que avalia a entrega e fornece *feedback*.

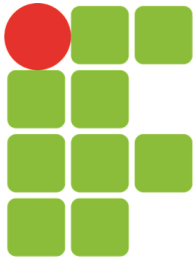
Fonte: PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 7. ed. 2011. (pp. 40-41).

# Atividades fundamentais para ES

- **Tipos de Fluxo de Processo** na "Metodologia de processo"* genérica para ES, de Pressman:
  - **Linear**: as atividades metodológicas são executadas em sequência.
  - **Iterativo**: repete uma ou mais atividades antes de seguir para a seguinte.
  - **Evolucionário**: as atividades são executadas de forma circular.
  - **Paralelo**: executa uma ou mais atividades em paralelo com outras atividades.

Fonte: PRESSMAN, R. S. Engenharia de software uma abordagem profissional. 7. ed. 2011. (pp. 54).





INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO SUL
Campus Feliz

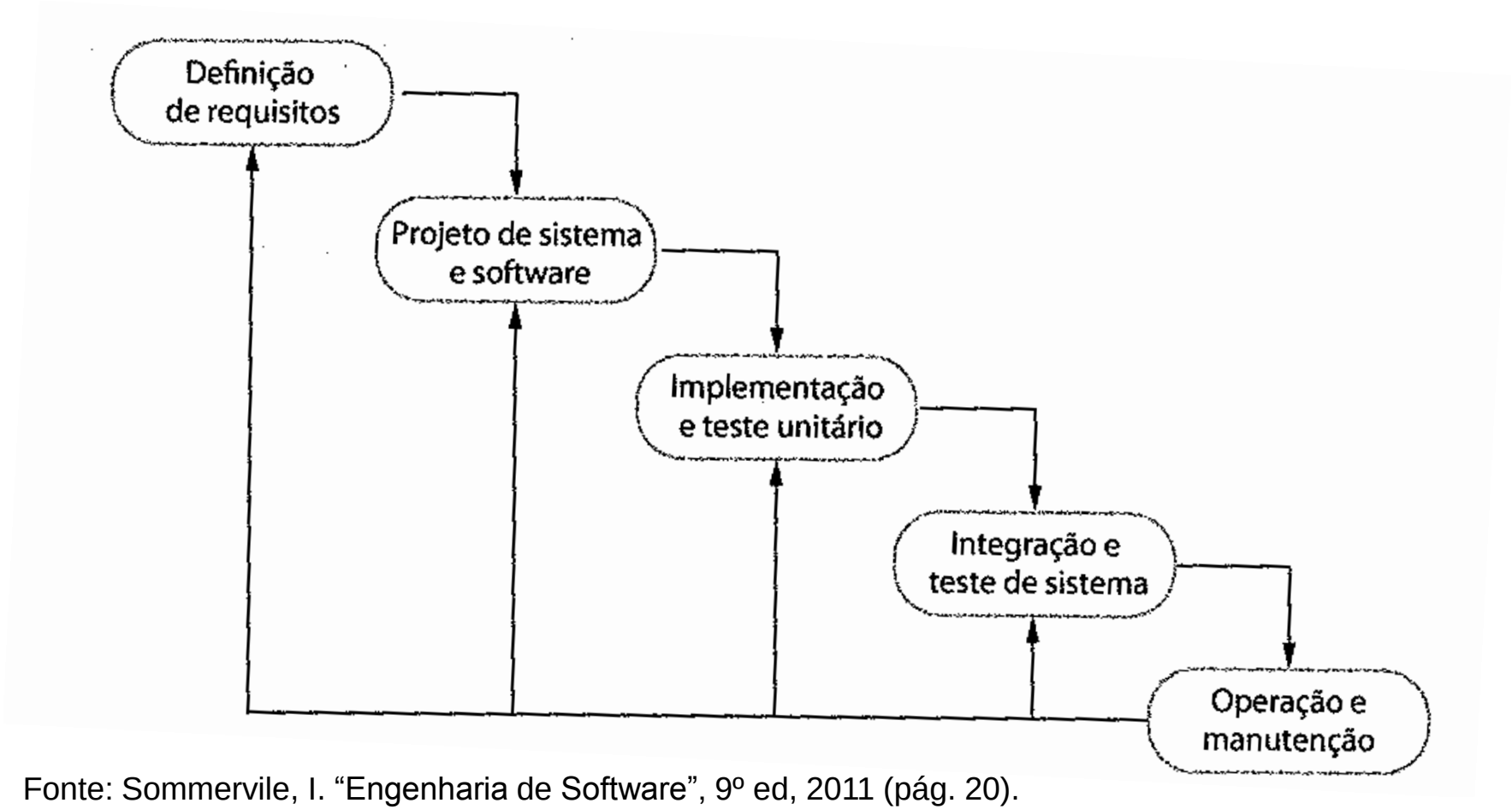
# Modelos de Processo de Software

## (ou Modelos de Processo Prescritivo)

# Modelo Cascata [1..5]

- Também chamado de ***ciclo de vida clássico***.
- Primeiro modelo de processo de software a ser publicado (em 1970), sendo derivado de processos mais gerais da engenharia de sistemas.

# Modelo Cascata [2..5]




Fonte: Sommervile, I. “Engenharia de Software”, 9º ed, 2011 (pág. 20).

# Modelo Cascata [3..5]

- Planificação cuidadosa e ordem linear de atividades: fases sucessivas onde os resultados de uma fase tornam-se entradas da próxima.
- Verificação no fim de cada fase para certificar-se que seus resultados intermediários são consistentes com a entrada e com requisitos do sistema.
- Requisitos do usuário são “congelados” antes do início do projeto.

# Modelo Cascata [4..5]

- É adequado quando:
  - Existe um conjunto de requisitos do usuário estáveis e de alta qualidade;
  - O sistema completo deve estar disponível de um única vez.

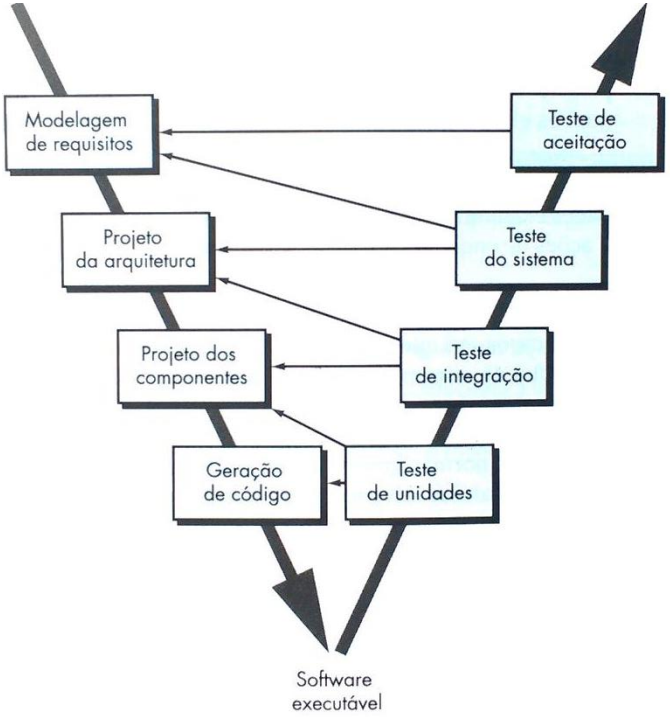
- Vantagens:
  - Torna o processo de desenvolvimento estruturado.
  - Só avança para a tarefa seguinte quando o cliente valida e aceita os produtos finais da tarefa atual.
  - Minimiza o impacto da compreensão adquirida no decorrer do projeto, uma vez que se um processo não pode voltar atrás de modo a alterar os modelos e as conclusões das tarefas anteriores, é normal que as novas ideias sobre o sistema não sejam aproveitadas.

# Modelo Cascata [5..5]

- Desvantagens:
  - Dificuldade de assimilar e reagir a mudanças (de requisitos, qualidade , etc.) depois de iniciado.
    - Por que é um problema? Qualquer erro ou mal-entendido, se não for detectado no início do processo, pode ser desastroso. É difícil para o cliente especificar os requisitos explicitamente no início do projeto.
  - É excessivamente sincronizado.
    - Por que é um problema? Os projetos raramente seguem o fluxo sequencial que o modelo propõe. A iteração é sempre necessária e está presente, criando problemas na aplicação do modelo.
  - Se ocorrer um atraso, todo o processo é afetado.
  - A maioria dos programas só fica disponível quando o cronograma já está bastante adiantado.

# Modelo V

- Variação do Modelo Cascata.
- Não é um novo modelo, mas sim uma forma de visualização da verificação.



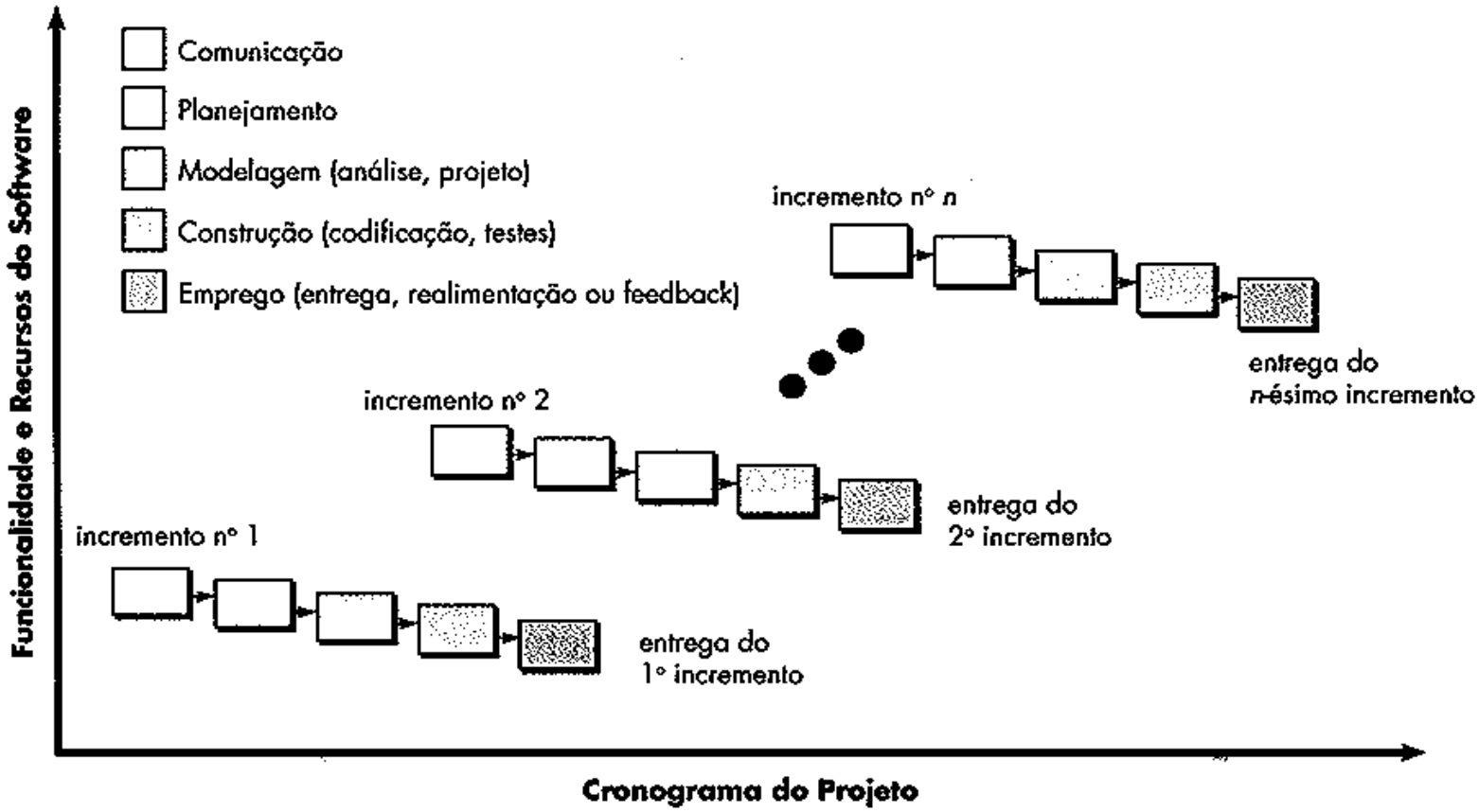

Fonte: adaptado de PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 7. ed. Rio de Janeiro: McGraw-Hill, 2011

# Modelos de processo Incremental [1..6]

- Baseia-se na ideia de desenvolver uma implementação inicial, expô-la aos comentários dos usuários e continuar por meio da criação de várias versões até que um sistema adequado seja desenvolvido.
  - Inspirado na maneira como resolvemos problemas (ir passo a passo em direção a uma solução).
- Cada incremento ou versão do sistema incorpora alguma funcionalidade necessária para o cliente.
  - Incrementos iniciais geralmente incluem a funcionalidade mais importante ou mais urgente.

# Modelos de processo Incremental [2..6]

# Modelos de processo Incremental [3..6]




Fonte: PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 7. ed. 2011. (pp. 61).

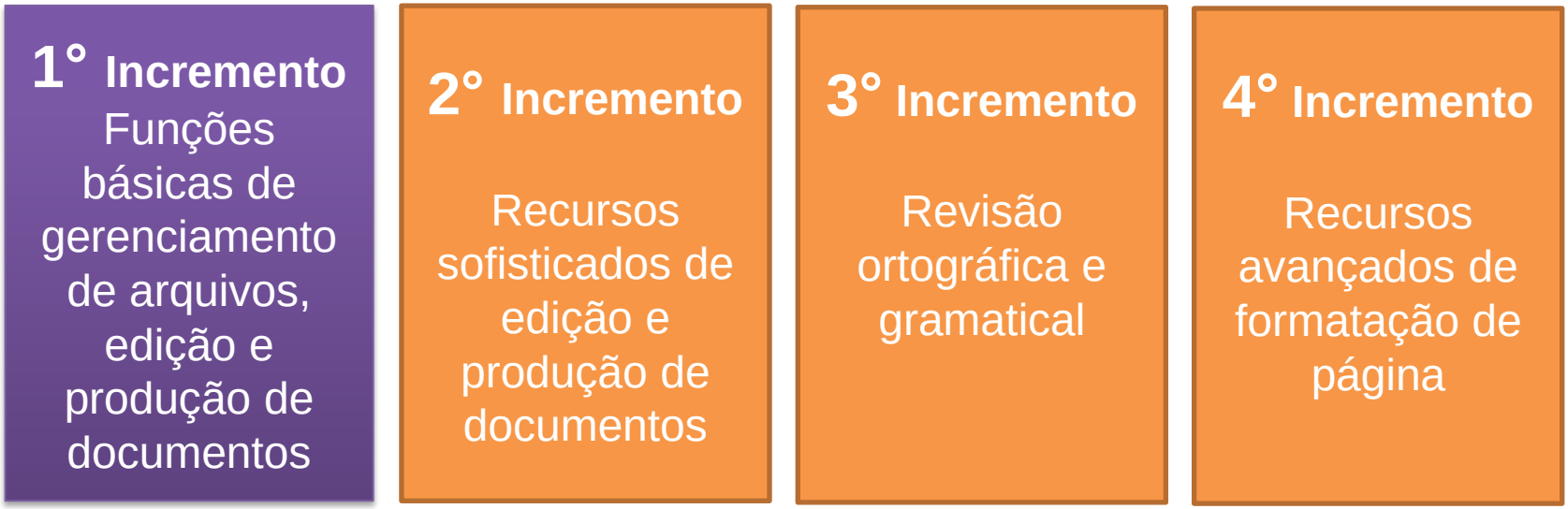
# Modelos de processo Incremental [4..6]

- Ao invés de implantar o sistema através de uma única entrega, o desenvolvimento e a implantação são quebrados em incrementos.
- Uma vez que o desenvolvimento de um incremento é iniciado, os requisitos deste incremento são congelados (embora aqueles de incrementos posteriores possam sofrer alterações).
- Envolve a integração contínua da arquitetura do sistema para a produção de novas versões do sistema, onde cada nova versão incorpora aperfeiçoamentos incrementais em relação a anterior pela incorporação do produto resultante da última **iteração.**

- **Iterações** e **incrementos** são características essenciais do Processo Unificado (RUP), do *Extreme Programming* e de *frameworks* para desenvolvimento ágil.

# Modelos de processo Incremental [5..6]

- Exemplo: software de **processamento de texto** desenvolvido de forma incremental:




**PRODUTO ESSENCIAL**



Fonte: adaptado de PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 7. ed. Rio de Janeiro: McGraw-Hill, 2011

# Modelos de processo Incremental [6..6]

- Vantagens do desenvolvimento incremental em relação ao modelo Cascata:
  - Menor custo para atender as mudanças nos requisitos dos clientes.
  - Facilidade em obter *feedback* dos clientes sobre o desenvolvimento já feito (é difícil avaliar a evolução por meio de documentos de projeto de software).
  - "É possível obter entrega e implementação rápida de um software útil ao cliente, mesmo se toda a funcionalidade não for incluída."

Fonte: Sommervile, I. "Engenharia de Software", 9º ed, 2011 (pág. 20).

# Modelos de processo Evolucionário

- São iterativos e possibilitam desenvolver versões cada vez mais completas do software.

- Exemplos: **Prototipação** e **Espiral**

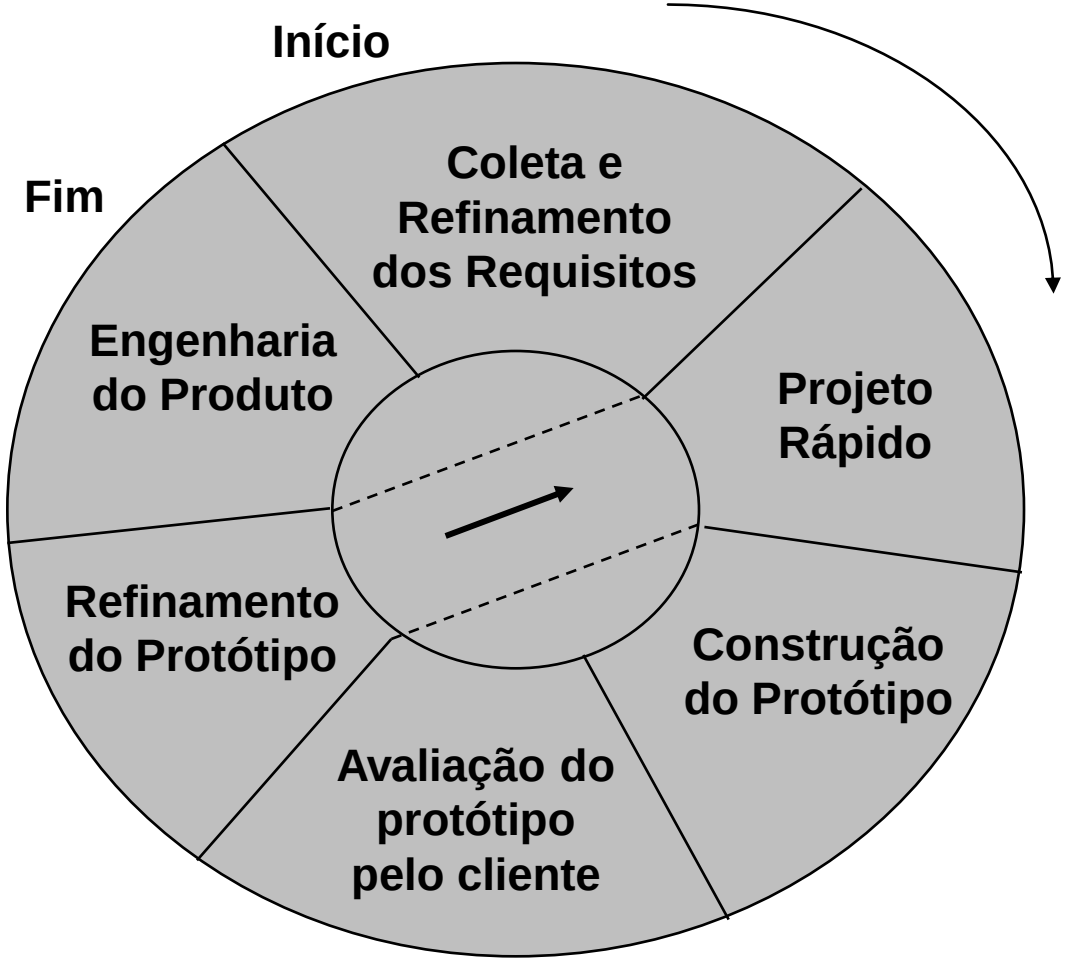
# Modelos de processo Evolucionário

## Prototipação [1..6]

- Pode ser utilizado como um modelo de processo isolado ou como uma técnica implementada no contexto de outro modelo de processo.
  - Independente de forma que é aplicado, auxilia a **compreender melhor o que está sendo construído**.
    - No processo de engenharia de requisitos, um protótipo pode ajudar na elicitação e validação de requisitos de sistema.
    - No processo de projeto de sistema, um protótipo pode ser usado para estudar soluções específicas do software e para apoiar o projeto de interface de usuário.

# Modelos de processo Evolucionário

## Prototipação [2..6]



# Modelos de processo Evolucionário

## Prototipação [3..6]

- Vantagens:
  - Prova de conceito;
  - Visibilidade antecipada do produto;
  - Encoraja participação dos usuários;
  - Exercício para identificar inconsistências de análise e projeto;
  - Redução do esforço no desenvolvimento;
  - Refina potenciais riscos de projeto.

# Modelos de processo Evolucionário

## Prototipação [4..6]

- Desvantagens:
  - Atividade de análise simplificada.
  - Confusão do usuário entre protótipo e o sistema real (desempenho, validação, etc).
  - Tempo e custo excessivo para desenvolvimento.
  - Falta de visibilidade do processo.
  - Dificuldade em manter a integridade entre protótipo e o *software* desenvolvido durante sua manutenção.

# Modelos de processo Evolucionário

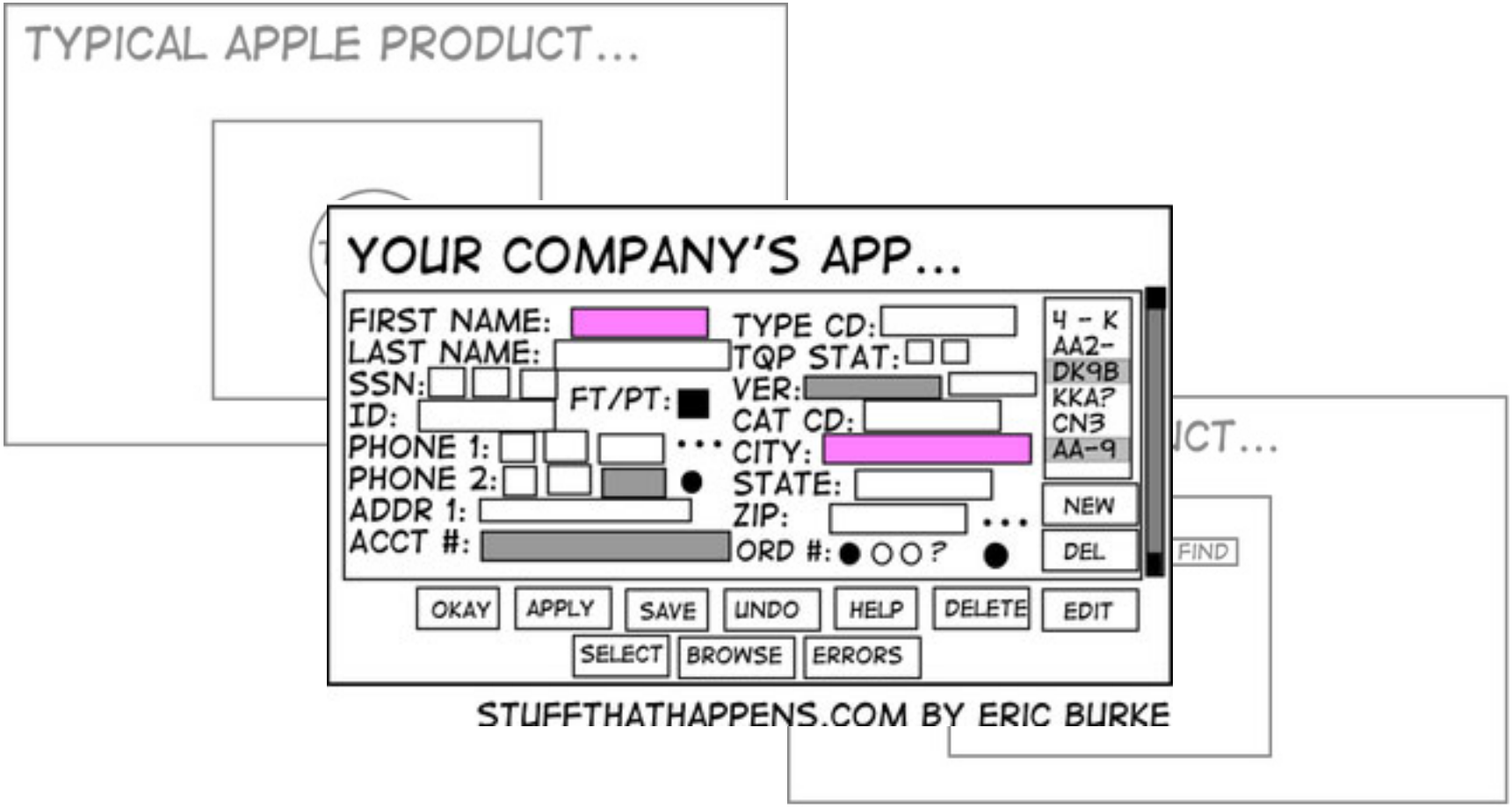
## Prototipação [5..6]

- Classificações comuns:
  - **Vertical**: funcionalidade em profundidade;
  - **Horizontal**: amplitude de funções superficiais.

  - **Evolucionário**: evoluir até chegar a um produto;
  - **Descartável**: entender requisitos.

# Modelos de processo Evolucionário

## Prototipação [6..6]



# Atividade - Modelo de Prototipação

- Faça um *mock-up* para um sistema de reserva de salas.
  - O Usuário deve poder indicar qual sala deseja utilizar em determinada data e também a descrição e o horário de início e término da atividade.

- Sugestões de ferramentas:
  - http://builds.balsamiq.com/b/mockups-web-demo/
  - http://pencil.evolus.vn/en-US/Home.aspx
  - http://mockupbuilder.com/

# Modelos de processo Evolucionário

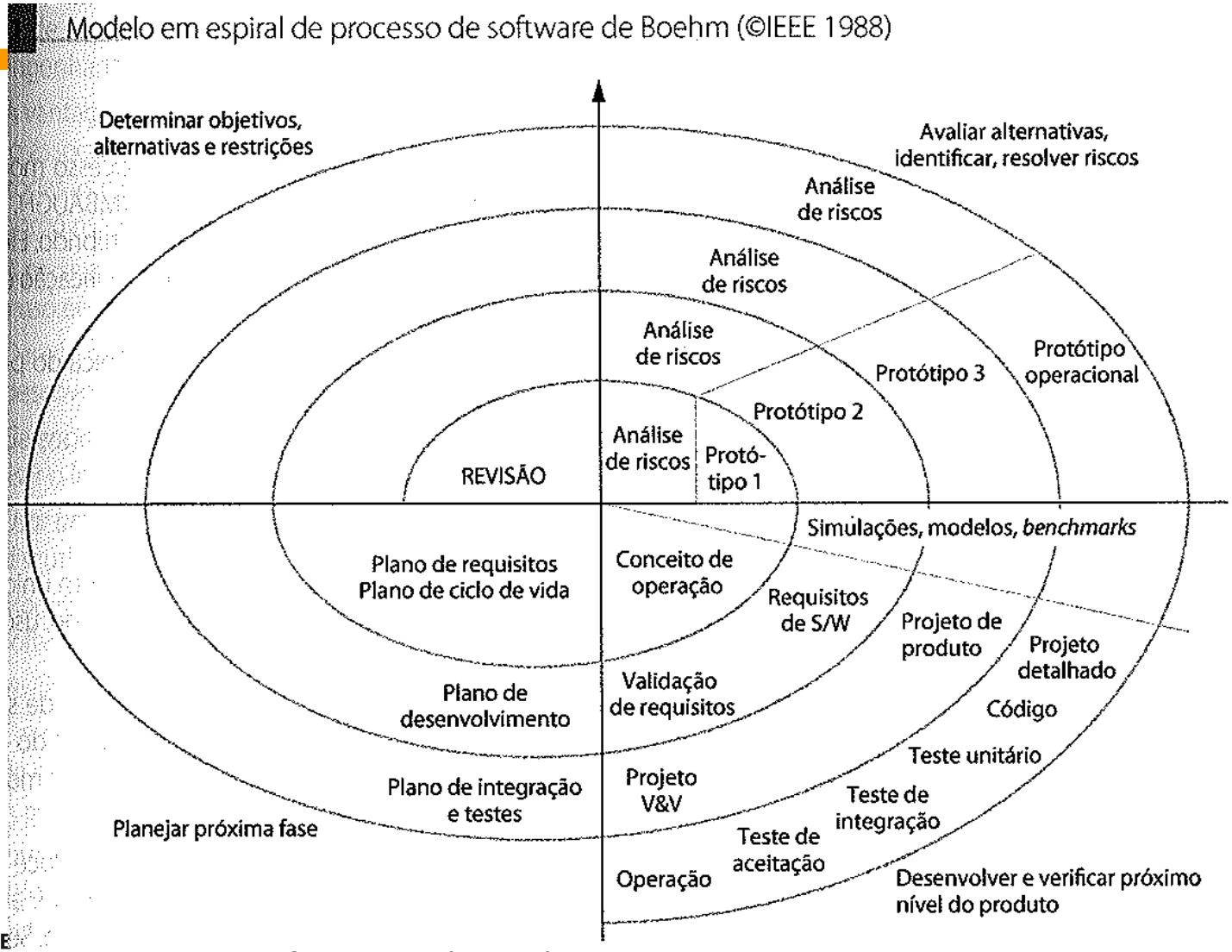
## Espiral [1..5]

- Proposto por Barry Boehm em 1988.
- “Acopla a natureza iterativa da prototipação com os aspectos sistemáticos e controlados do modelo cascata”
- O software é desenvolvido em uma série de versões evolucionárias.
  - Nas primeiras iterações, a versão pode ser um modelo ou um protótipo. Nas iterações posteriores, são produzidas versões cada vez mais completas do software.

Fonte: PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 7. ed. 2011. (pág. 65).

# Modelos de processo Evolucionário

## Espiral [2..5]



# Modelos de processo Evolucionário

## Espiral [3..5]

- Cada volta da espiral é dividida em quatro setores:
  - Definição de objetivos: são definidos os objetivos específicos para essa fase do projeto; são identificadas restrições ao processo e ao produto; é elaborado um plano de gerenciamento detalhado; os riscos do projeto são identificados.
  - Avaliação e redução de riscos: para cada risco identificado, é feita uma análise detalhada. São realizadas medidas para redução de riscos.
  - Desenvolvimento e validação: é selecionado um modelo de desenvolvimento para o sistema (prototipação, transformações formais, cascata ou outro).
  - Planejamento: o projeto é revisado e é tomada uma decisão a respeito da continuidade com mais uma volta na espiral. Caso se decida pela continuidade, são elaborados planos para a próxima fase do projeto.

Fonte: Sommervile, I. “Engenharia de Software”, 9º ed, 2011 (pág. 33).

# Modelos de processo Evolucionário

## Espiral [4..5]

- Acrescenta **aspectos gerenciais** ao processo de desenvolvimento de software.
- Cada volta na espiral representa uma fase no processo.
- Não há fases fixas como especificação ou projeto: voltas na espiral são escolhidas dependendo do que é requerido.
- **Riscos** são avaliados explicitamente e resolvidos ao longo do processo.

# Modelos de processo Evolucionário

## Espiral [5..5]

- Vantagens:
  - Modelo evolutivo que possibilita uma maior integração entre as fases e facilita a depuração e a manutenção do sistema.
  - Permite que o projetista e cliente possam entender e reagir aos riscos em cada etapa evolutiva.
- Desvantagens:
  - Avaliação dos riscos exige muita experiência.

# Modelos de processo Especializado

- De acordo com Pressman, baseiam-se nas características de um ou mais modelos tradicionais apresentados anteriormente.
  - Exemplos: desenvolvimento baseado em componentes, modelo de métodos formais e desenvolvimento de software orientado a aspectos.
  - Observação: Sommerville cita a **Engenharia de Software orientada a reuso**, cuja ideia é bastante similar ao **Desenvolvimento baseado em Componentes**.

# Modelos de processo Especializado

## Desenvolvimento baseado em Componentes [1..2]

- Incorpora muitas das características do modelo espiral.
- Desenvolve aplicações a partir de componentes de software pré-empacotados.
  - Componentes de software comercial de prateleira ou COTS (*Commercial Off-The-Shelf*) são desenvolvidos por vendedores e comercializados como produtos.
- Etapas:
  1. Produtos baseados em componentes disponíveis são pesquisados e avaliados para o campo de aplicação em questão.
  2. Itens de integração de componentes são considerados.
  3. Uma arquitetura de software é projetada para acomodar os componentes.
  4. Os componentes são integrados na arquitetura.
  5. Testes são realizados para assegurar funcionalidade adequada.



Fonte: PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 7. ed. 2011. (pág. 69).

# Modelos de processo Especializado

## Desenvolvimento baseado em Componentes [2..2]

- Modelo de processo geral de desenvolvimento baseado no reuso:


Fonte: Sommervile, I. “Engenharia de Software”, 9º ed, 2011 (pág. 23).

# Modelos de processo Especializado

## Métodos Formais

- Engloba um conjunto de atividades que conduzem à **especificação matemática formal do software**.
- Métodos formais eliminam muitos dos problemas difíceis de ser superados com o uso de outros paradigmas de ES (como ambiguidade, incompletude e inconsistência).
- Dificuldades encontradas para uso:
  - Consome muito tempo e dinheiro;
  - Poucos desenvolvedores possuem formação e experiência;
  - Dificuldade de usar os modelos para comunicação com os clientes.

Fonte: PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 7. ed. 2011. (pág. 69-70).

# Modelos de processo Especializado

## Desenvolvimento de Software Orientado a Aspectos

- Sistemas mais sofisticados podem apresentar certas *restrições* que se estendem por toda a sua arquitetura.
  - Restrições cruzadas "cruzam" múltiplas funções, recursos e informações do sistema.
- O desenvolvimento de software orientado a aspectos é um paradigma novo que oferece um abordagem metodológica e de processos para definir, especificar, projetar e construir **aspectos** – "mecanismos além das sub-rotinas e herança para localizar a expressão de uma restrição cruzada."

Fonte: PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 7. ed. 2011. (pág. 70-71).

# Processo Unificado

- Inicialmente proposto por Ivar Jacobson, Grady Booch e James Rumbaugh em 1999.
  - Defendiam a necessidade de um processo de software “dirigido a casos de uso, centrado na arquitetura, iterativo e incremental”.
    PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 7. ed. 2011. (pág. 71).
- Também chamado de PU ou UP (*Unified Process*).
- O RUP (*Rational Unified Process*), proposto em 2003, é um processo proprietário derivado do PU.
  - Atualmente vem sendo chamado de **IRUP**, de *IBM Rational Unified Process.*

# Processo Unificado

- É dividido em fases:
  - **Concepção***: estabelece o caso de negócio para o sistema e delimita o escopo do projeto, ou seja, envolve tanto a atividade de comunicação com o cliente como a de planejamento.
    - Necessidades de negócio, arquitetura rudimentar, planejamento (recursos necessários, riscos e cronograma).
  - **Elaboração**: envolve a análise do domínio do problema e o refinamento da especificação da arquitetura do sistema.
    - Pode gerar uma "base de arquitetura executável", que demonstra a viabilidade, mas não oferece todos os recursos.
    - O plano é revisado.

* do inglês *inception*, também traduzido como "iniciação".

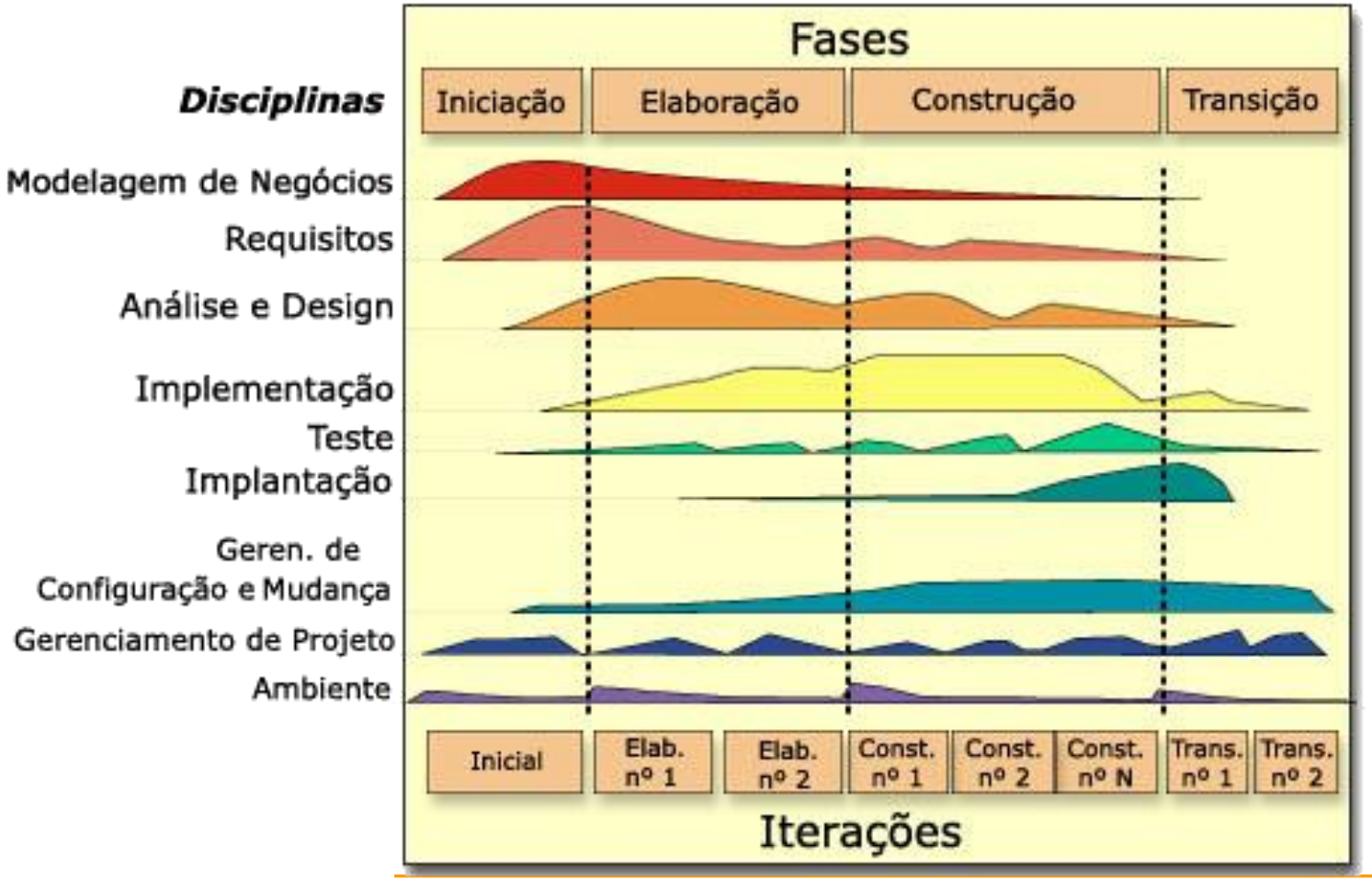
Fonte: PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 7. ed. 2011. (pág. 72-73).

# Processo Unificado

- Fases (continuação):
  - **Construção**: envolve a elaboração do software a partir de arquitetura em um produto completo para utilização pelos usuários.
    - Os modelos de requisitos e de projeto (iniciados na fase de elaboração) são completados.
    - Podem ser desenvolvidos ou adquiridos componentes de software.
    - São realizados testes de unidade nos componentes já desenvolvidos.
    - Realizam-se atividades de integração.
  - **Transição**: viabiliza que o software possa ser utilizado pelos usuários.
    - São realizados testes beta e o *feedback* dos usuários relata defeitos e mudanças necessárias.
    - São elaborados materiais de apoio (como manuais).
  - **Produção**: emprego do processo genérico. É onde monitora-se o uso contínuo do software.

Fonte: PRESSMAN, R. S. Engenharia de software: uma abordagem profissional. 7. ed. 2011. (pág. 72-73).

# Processo Unificado - RUP



# Processo Unificado - RUP

- *Framework* que fornece: atividades, papéis e artefatos necessários para o desenvolvimento de software.
- Suas principais características são:
  - Desenvolvimento iterativo e incremental;
  - Orientado a objetos;
  - Foco na criação de uma arquitetura robusta;
  - Análise de riscos;
  - Utilização de casos de uso para o desenvolvimento.

# Processo Unificado - RUP

- Normalmente descrito em três perspectivas:
  - Perspectiva dinâmica, que mostra as fases do modelo ao longo do processo.
  - Perspectiva estática, que mostra as atividades realizadas (no caso, os *workflows* centrais e de apoio).
  - Perspectiva prática, que sugere boas práticas a serem utilizadas.

# Processo Unificado – RUP

## *Workflows* ou Disciplinas (perspectiva estática)

| WORKFLOW | DESCRIÇÃO |
|---|---|
| Modelagem de negócios | Os processos de negócio são modelados por meio de casos de uso de negócios. |
| Requisitos | Atores que interagem com o sistema são identificados e casos de uso são desenvolvidos para modelar os requisitos do sistema. |
| Análise e projeto | Um modelo de projeto é criado e documentado com modelos de arquitetura, modelos de componentes, modelos de objetos e modelos de sequência. |
| Implementação | Os componentes do sistema são implementados e estruturados em subsistemas de implementação. A geração automática de código a partir de modelos de projeto ajuda a acelerar esse processo. |
| Teste | O teste é um processo iterativo que é feito em conjunto com a implementação. O teste do sistema segue a conclusão da implementação. |
| Implantação | Um *release* do produto é criado, distribuído aos usuários e instalado em seu local de trabalho. |
| Gerenciamento de configuração e mudanças | Esse *workflow* de apoio gerencia as mudanças do sistema (veja o Capítulo 25). |
| Gerenciamento de projeto | Esse *workflow* de apoio gerencia o desenvolvimento do sistema (veja os capítulos 22 e 23). |
| Meio ambiente | Esse *workflow* está relacionado com a disponibilização de ferramentas apropriadas para a equipe de desenvolvimento de software. |

# Processo Unificado – RUP

## Boas práticas (perspectiva prática)

1. Desenvolver software iterativamente (planejar os incrementos com base nas prioridades do cliente, desenvolvendo primeiro os recursos de alta prioridade)
2. Gerenciar os requisitos (documentar explicitamente, analisando o impacto das mudanças antes de aceitá-las)
3. Usar arquitetura baseada em componentes
4. Modelar o software visualmente (gráficos UML)
5. Verificar a qualidade do software
6. Controlar as mudanças do software (usar sistema de gerenciamento de mudanças)

# Considerações

- Aspectos comuns a todos modelos:
  - Todos começam com (ou visam) uma compreensão melhor dos requisitos do sistema a ser desenvolvido.
  - Todos visam aumento da qualidade dos resultados parciais e finais.
  - Todos visam melhoria da gerência do processo de desenvolvimento (alguns, o controle do processo; outros, o acompanhamento do processo).