AULA 2 - PROCESSOS DE SOFTWARE

Objetivos da Aula

- **Conteúdo:**
  - Compreender o conceito e a importância de um Processo de Software.
  - Conhecer os componentes típicos que descrevem um processo de software.
  - Analisar os principais modelos genéricos de processo de software: Cascata, Desenvolvimento Incremental e Engenharia de Software Orientada a Reúso.
  - Identificar as atividades fundamentais presentes em todos os processos de software.
  - Discutir como os processos de software podem ser adaptados para lidar com mudanças.
  - Apresentar o Rational Unified Process (RUP) como um exemplo de processo de software moderno e adaptável.
- **Nota:** "Entender os processos de software é crucial para planejar, gerenciar e executar projetos de software de forma eficaz."

O que é um Processo de Software?

Definição Principal:

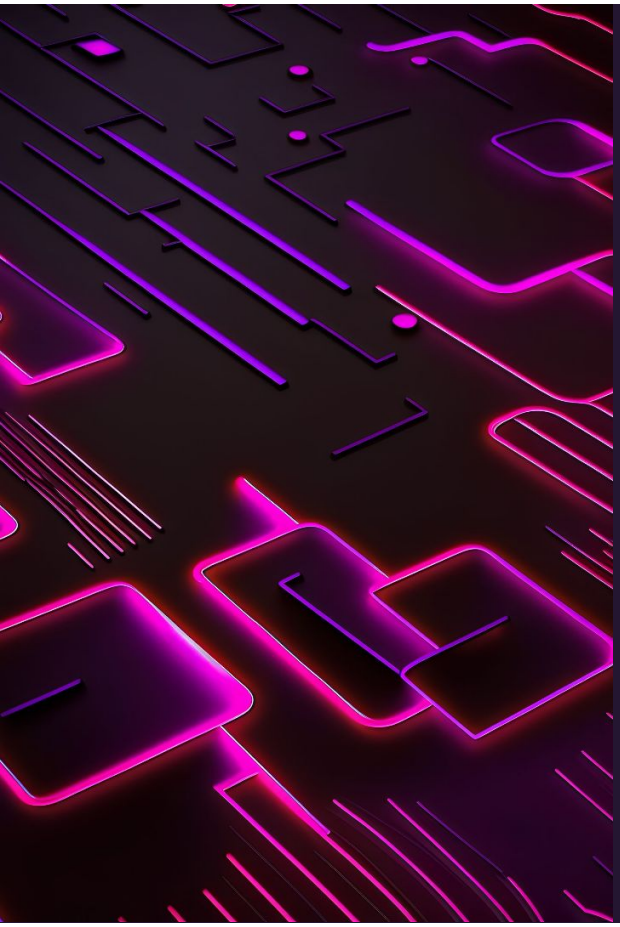
"Um processo de software é um conjunto estruturado de atividades cujo objetivo é o desenvolvimento ou a evolução de software." (Adaptado de Sommerville, 2011, 9ª ed., Cap. 2)

Explicação

- **Conjunto Estruturado:** As atividades não são aleatórias; elas possuem uma ordem, interdependências e um propósito definido.
- **Atividades:** Englobam tudo, desde a conversa inicial com o cliente até a operação e manutenção do sistema.
- **Desenvolvimento ou Evolução:** Aplica-se tanto à criação de novos sistemas quanto à modificação e melhoria de sistemas existentes.

Por que Processos?

1. **Organização e Controle:** Fornecem um framework para gerenciar a complexidade do desenvolvimento de software.
2. **Qualidade** Processos bem definidos tendem a levar a produtos de maior qualidade.
3. **Repetibilidade e Previsibilidade** Permitem que sucessos sejam repetidos e que se façam estimativas mais precisas.
4. **Comunicação** Facilitam a comunicação entre os membros da equipe e com os stakeholders.



Componentes de um Processo de Software

1. Ao descrever um processo, geralmente detalhamos (Sommerville, 2011, 9ª ed., Cap. 2):

Produtos | Papéis | Pré e Pós-condições | Atividades

Componentes de um Processo de Software

Produtos (ou Artefatos):

- **Definição:** São os resultados de uma atividade do processo.
- **Exemplos:** Documento de requisitos, modelo de arquitetura, código-fonte, plano de testes, manual do usuário.

Papéis:

- **Definição:** Refletem as responsabilidades das pessoas envolvidas no processo.
- **Exemplos:** Gerente de projeto, analista de requisitos, arquiteto de software, programador, testador.

Componentes de um Processo de Software

Pré e Pós-condições:

- **Definição:** Declarações que devem ser verdadeiras antes (pré-condição) e depois (pós-condição) de uma atividade do processo ser executada ou um produto ser criado.

Atividades:

- **Definição:** As tarefas específicas que transformam entradas em saídas.
- **Exemplos:** Elicitar requisitos, projetar a interface do usuário, codificar um módulo, executar casos de teste.

Modelos de Processo de Software Genéricos - Introdução

Introdução:

- "Um modelo de processo de software é uma **representação abstrata simplificada** de um processo de software, apresentada de uma perspectiva particular." (Sommerville, 2011, 9ª ed., Cap. 2)
- Não são descrições definitivas, mas sim *frameworks* que podem ser adaptados.

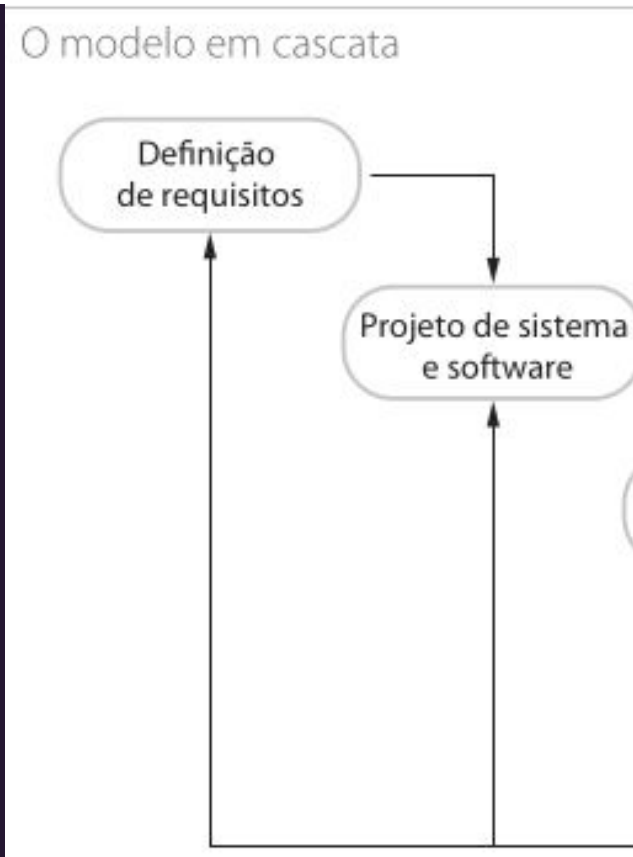
Modelos de Processo de Software Genéricos - Introdução

- Serão apresentados três modelos fundamentais:
  - Modelo em Cascata
  - Desenvolvimento Incremental
  - Engenharia de Software Orientada a Reúso
- **Nota:** "A escolha do modelo de processo depende do tipo de projeto, dos requisitos, da equipe e da cultura organizacional."

Modelo em Cascata (Waterfall)

Derivado de processos mais gerais da engenharia de sistemas (Royce, 1970 – embora Royce não tenha usado o termo "cascata" e tenha enfatizado a necessidade de iteração, seu diagrama influenciou este modelo).

Características Principais:
Processo sequencial com fases distintas e bem definidas. O resultado de cada fase flui para a próxima, como uma cascata.



Fases (Adaptado de Sommerville, Cap. 2):

Análise e Definição de Requisitos: Estabelecer serviços, restrições e metas do sistema.

Projeto de Sistema e Software: Alocar requisitos para hardware/software, definir arquitetura geral e abstrações do software.

Implementação e Teste Unitário: Desenvolver e verificar unidades de programa individuais.

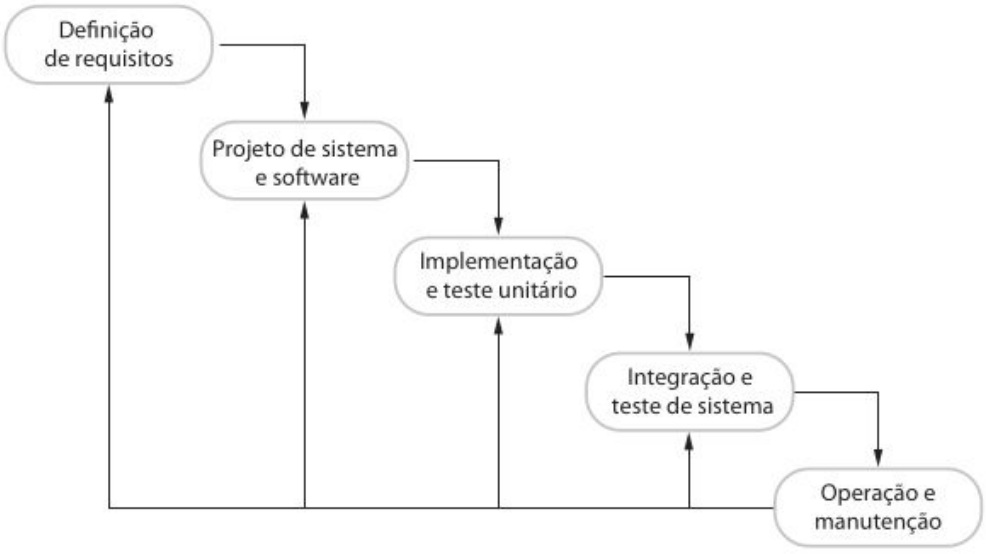
Integração e Teste de Sistema: Integrar unidades e testar o sistema completo contra os requisitos.

Modelos de Processo de Software Genéricos - Introdução

- **Operação e Manutenção:** Instalar, colocar em uso, corrigir erros e evoluir o sistema.

Última e Não menos Importante

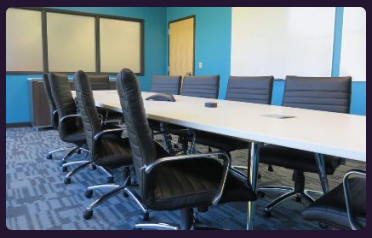
Figura 2.1 O modelo em cascata



Modelo em Cascata - Vantagens

Simplicidade: Fácil de entender e gerenciar.

Clareza nas Fases e Entregáveis: Marcos bem definidos facilitam o acompanhamento do progresso.



Disciplina: Força uma abordagem estruturada e documentada.

Modelo em Cascata - Desvantagens

- **Inflexibilidade a Mudanças:** Dificuldade em acomodar mudanças nos requisitos após uma fase ser "congelada".
- **Documentação Intensiva:** Pode gerar grande volume de documentação, nem sempre útil.
- **Feedback Tardio do Cliente:** O cliente só vê o sistema funcionando nas fases finais.
- **Separação Rígida das Fases:** Na prática, as fases frequentemente se sobrepõem.

Quando Usar (Sommerville, Cap. 2):

- Quando os requisitos são bem compreendidos e espera-se que sejam estáveis.
- Em grandes sistemas onde o planejamento detalhado e a documentação são essenciais.

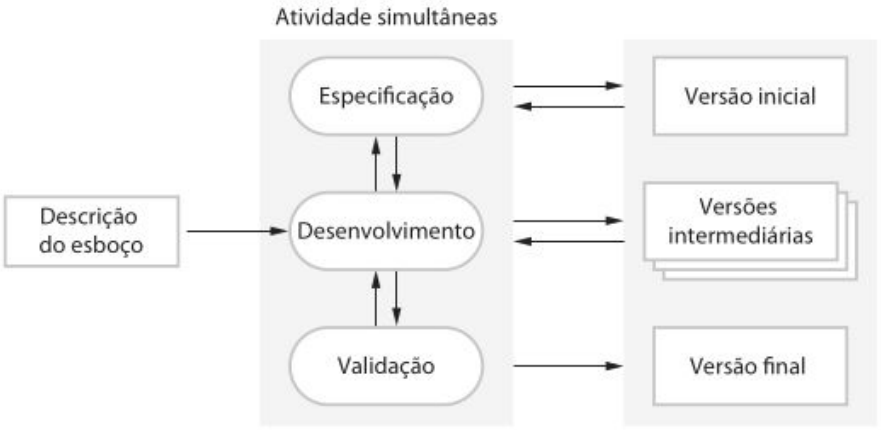
Citação Contextual: "O modelo em cascata é mais adequado para projetos onde os requisitos são bem entendidos e é improvável que mudem radicalmente durante o desenvolvimento do sistema." (Paráfrase de Sommerville, 2011, 9ª ed., Cap. 2)

Desenvolvimento Incremental

- "Uma abordagem que intercala as atividades de especificação, desenvolvimento e validação. O sistema é desenvolvido como uma série de versões (incrementos), de maneira que cada versão adiciona funcionalidade à anterior." (Sommerville, 2011, 9ª ed., Cap. 2)

- **Características:**
  - O sistema é entregue em partes funcionais (incrementos).
  - O cliente pode usar os primeiros incrementos e fornecer feedback.
  - Priorização de funcionalidades: as mais importantes ou de maior risco podem ser desenvolvidas primeiro.

Figura 2.2 Desenvolvimento incremental



Desenvolvimento Incremental - Vantagens

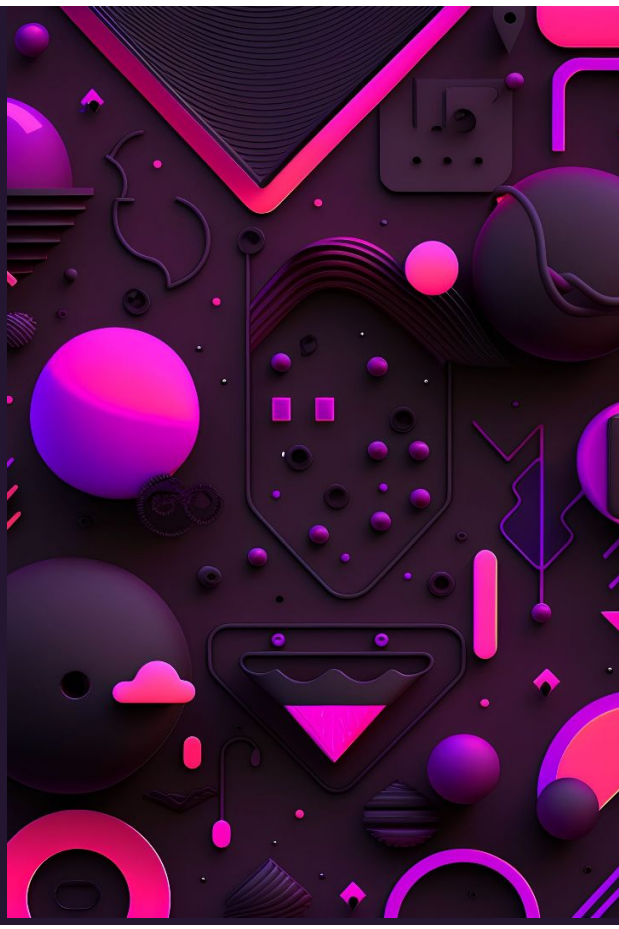
- **Redução do Custo de Mudanças:** Menos análise e documentação a serem refeitas.
- **Feedback Antecipado do Cliente:** Clientes podem avaliar o software em estágios iniciais e refinar requisitos.
- **Entrega Rápida de Software Útil:** Clientes obtêm valor mais cedo, mesmo que com funcionalidade parcial.
- Redução de riscos, pois os incrementos de maior risco podem ser implementados primeiro.

Desenvolvimento Incremental - Desvantagens

- **Visibilidade do Processo:** Gerentes podem ter dificuldade em medir o progresso sem a documentação formal de cada versão.
- **Degradação da Estrutura do Sistema:** Constantes mudanças, sem refatoração adequada, podem corromper a arquitetura, tornando futuras modificações difíceis e onerosas.
- **Contratos:** Pode ser difícil definir contratos completos quando os requisitos evoluem.

Quando Usar (Sommerville, Cap. 2):

  - Quando é difícil estabelecer todos os requisitos antecipadamente.
  - Para sistemas onde se espera uma rápida entrega de funcionalidades úteis.
  - Para sistemas de pequeno e médio porte, ou para subsistemas de sistemas maiores.
- **Citação Contextual:** "Atualmente, a abordagem mais comum para o desenvolvimento de sistemas aplicativos." (Paráfrase de Sommerville, 2011, 9ª ed., Cap. 2)



Engenharia de Software Orientada a Reúso (CBSE)

"Essa abordagem é baseada na existência de um número significativo de componentes reusáveis. O processo de desenvolvimento do sistema concentra-se na **integração desses componentes** em um sistema já existente em vez de desenvolver um sistema a partir do zero."

Fases do Processo (Adaptado de Sommerville, Cap. 2):

Análise de Componentes: Dada a especificação de requisitos, busca-se por componentes que implementem essa especificação.

Modificação de Requisitos: Os requisitos são analisados e modificados para refletir os componentes disponíveis.

Projeto do Sistema com Reúso: O framework do sistema é projetado (ou um existente é reusado), tendo em mente os componentes que serão integrados.

Desenvolvimento e Integração: Software novo é desenvolvido (se necessário) e os componentes (incluindo COTS - Commercial Off-The-Shelf) são integrados.

Tipos de Componentes Reusáveis:

Web services, coleções de objetos (.NET, J2EE), sistemas COTS configuráveis.

ES Orientada a Reúso - Vantagens

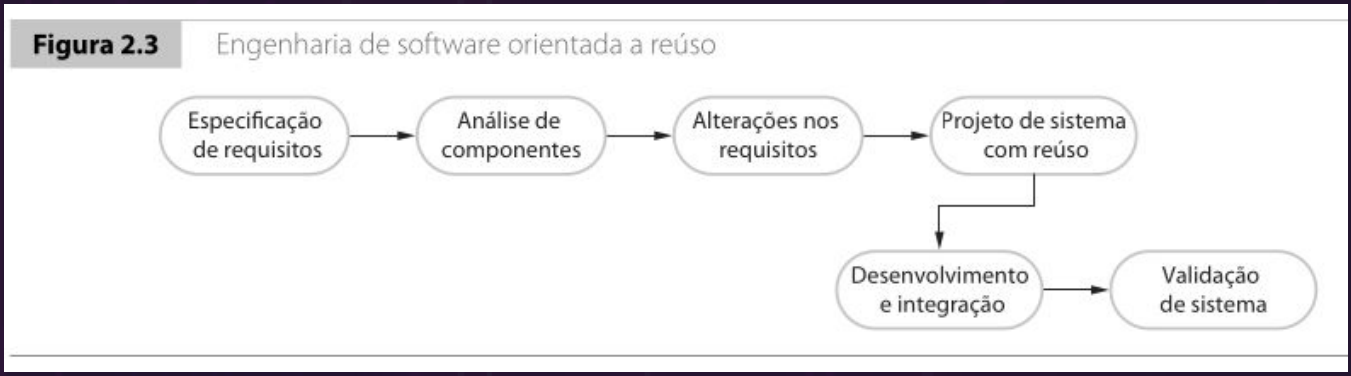
- **Redução de Custo e Tempo:** Menos software precisa ser desenvolvido do zero.
- **Redução de Riscos:** Componentes já testados e usados podem ser mais confiáveis.
- **Entrega Mais Rápida:** Montar o sistema a partir de peças prontas é geralmente mais rápido.
- **Especialização:** Permite que especialistas foquem no desenvolvimento de componentes de alta qualidade em suas áreas.

ES Orientada a Reúso - Desvantagens

- **Compromissos com Requisitos:** É inevitável ter que adaptar os requisitos para se ajustarem aos componentes existentes, o que pode levar a um sistema que não atende plenamente às necessidades dos usuários. (Sommerville, Cap. 2)
- **Perda de Controle sobre a Evolução:** Novas versões dos componentes reusados não estão sob o controle da organização que os utiliza.
- **Custos de Busca e Adaptação:** Encontrar, compreender e adaptar componentes pode ser custoso.
- **Manutenção:** Se o código-fonte do componente não está disponível, a manutenção pode ser mais cara.

ES Orientada a Reúso

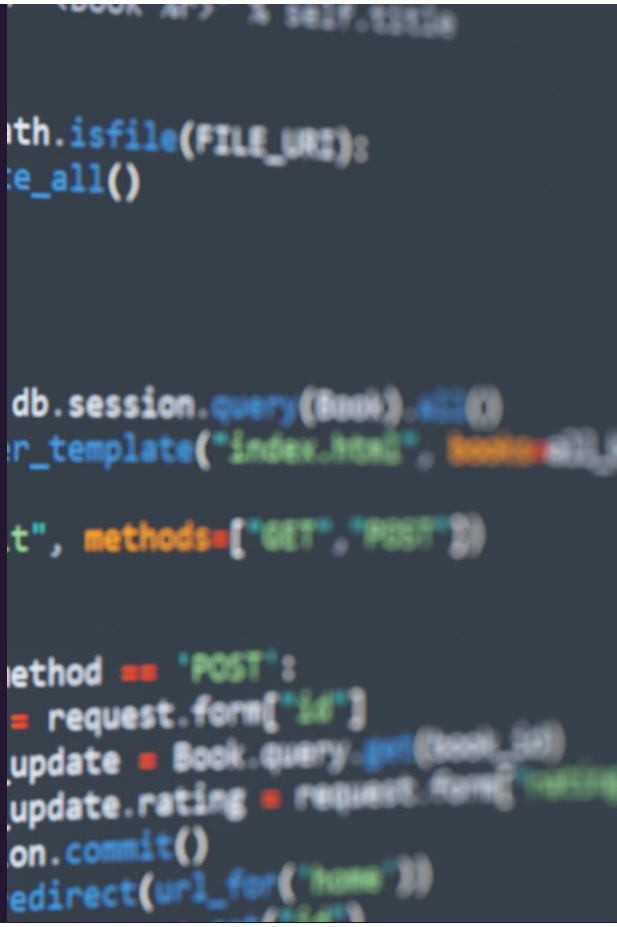
- **Citação (Contextual, M. D. McIlroy, 1968, "Mass Produced Software Components"):** A ideia de construir software a partir de componentes reutilizáveis é antiga, mas sua efetivação em larga escala ainda apresenta desafios.



Atividades Fundamentais do Processo de Software

"Embora os modelos de processo variem, todos eles, de alguma forma, incluem estas quatro atividades básicas." (Sommerville, Cap. 2)

Temos 4 Atividades:



ES Orientada a Reúso

Especificação de Software (Engenharia de Requisitos):

- Definição: O processo de compreender e definir quais serviços são requeridos do sistema e identificar as restrições sobre sua operação e desenvolvimento.
- Resultado: Documento de Requisitos.

Projeto e Implementação de Software:

- Definição: O processo de converter a especificação do sistema em um sistema executável. Envolve projeto de arquitetura, projeto detalhado e codificação.

ES Orientada a Reúso

Validação de Software (Verificação e Validação - V&V):

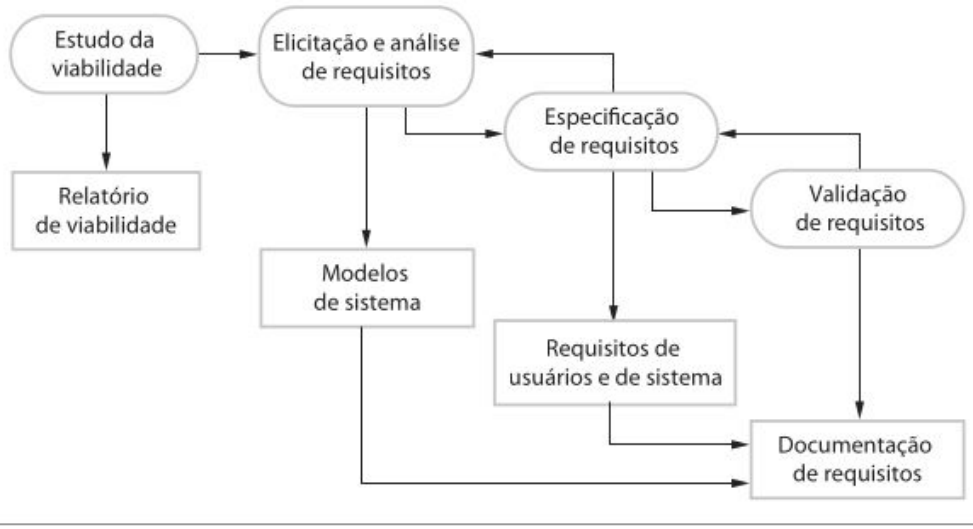
- Definição: O processo de checar se o sistema está de acordo com sua especificação e se atende às reais necessidades do cliente. Envolve testes, revisões e inspeções.

Evolução de Software (Manutenção):

- Definição: O processo de modificar o software após sua entrega para corrigir defeitos, adaptar-se a mudanças no ambiente ou a novas necessidades.

Nota: "Essas atividades são complexas e frequentemente intercaladas na prática."

Figura 2.4 Os requisitos da engenharia de processos



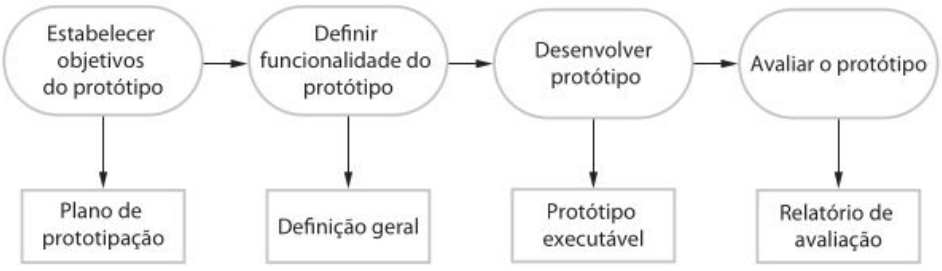
A Inevitabilidade da Mudança:

- Requisitos mudam (negócios evoluem, usuários aprendem mais).
- O ambiente (tecnológico, de mercado) muda.
- Processos de software devem ser capazes de acomodar essas mudanças.

Prototipação

Um protótipo é uma versão inicial de um sistema de software, usado para demonstrar conceitos, experimentar opções de projeto e descobrir mais sobre o problema e suas possíveis soluções.

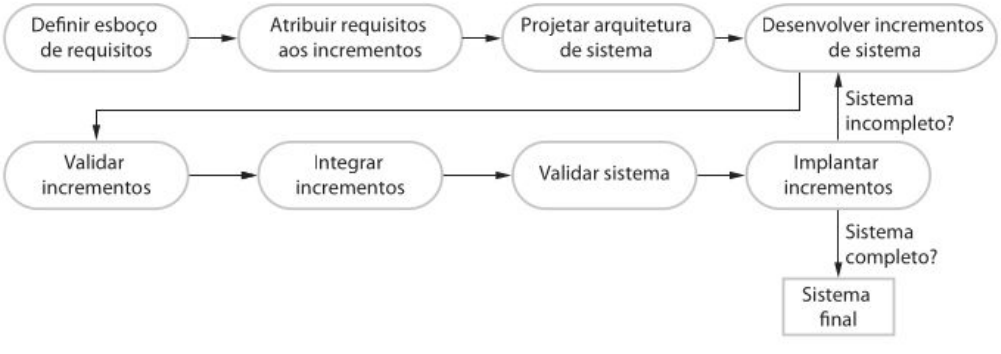
Figura 2.9 O processo de desenvolvimento de protótipo



Entrega incremental

Entrega incremental (Figura 2.10) é uma abordagem para desenvolvimento de software na qual alguns dos incrementos desenvolvidos são entregues ao cliente e implantados para uso em um ambiente operacional.

Figura 2.10 Entrega incremental

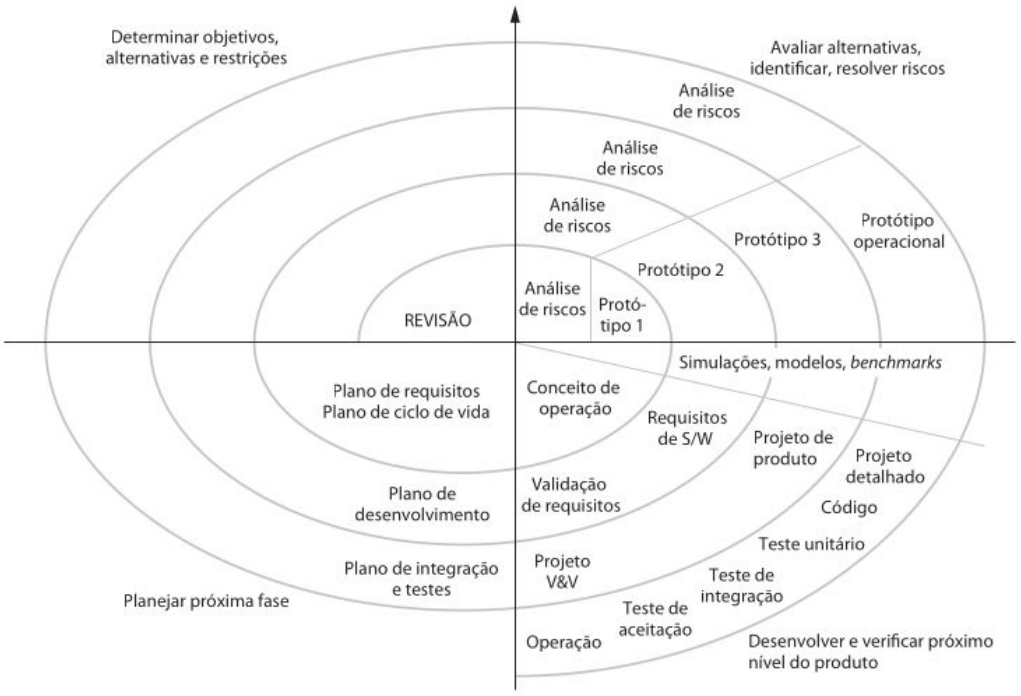


Modelo espiral de Boehm

Um framework de processo de software dirigido a riscos (o modelo em espiral) foi proposto por Boehm (1988). Isso está na Figura 2.11. Aqui, o processo de software é representado como uma espiral, e não como uma sequência de atividades com alguns retornos de uma para outra.

Cada volta na espiral representa uma fase do processo de software.

Figura 2.11 Modelo em espiral de processo de software de Boehm (©IEEE 1988)



Duas Abordagens para Reduzir o Custo do Retrabalho (Sommerville, Cap. 2):

Prevenção de Mudanças (Change Anticipation):

- O processo inclui atividades para antecipar possíveis mudanças.
- **Prototipação:** Uma versão inicial ou parcial do sistema é desenvolvida rapidamente para demonstrar conceitos
    - *Benefício:* Ajuda a evitar más decisões sobre requisitos e projeto

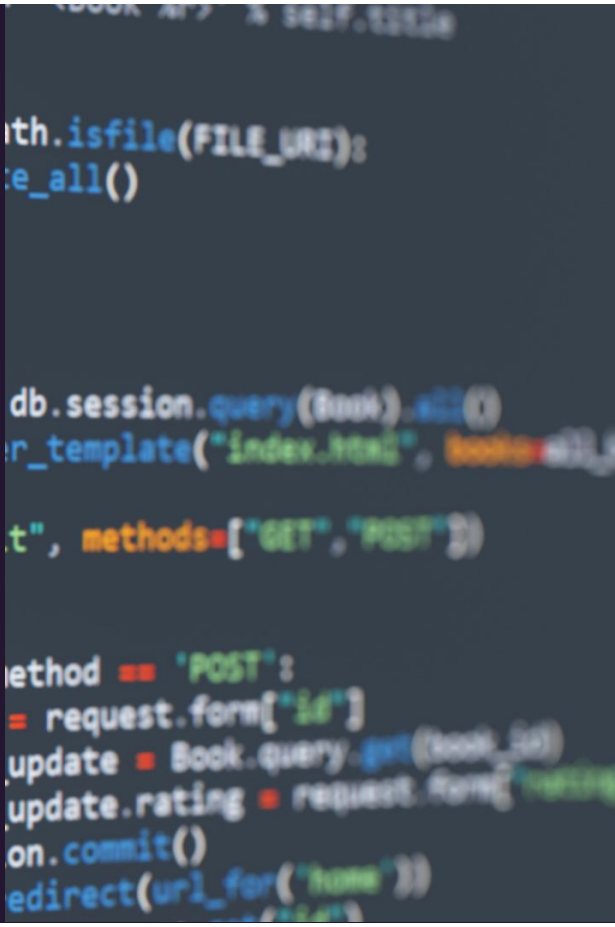
Tolerância a Mudanças (Change Tolerance):

- O processo é projetado para que as mudanças possam ser acomodadas a um custo relativamente baixo.
- **Entrega Incremental:** Os incrementos são entregues aos clientes para comentários e experimentação.
    - *Benefício:* Evita o comprometimento prematuro com requisitos para todo o sistema.

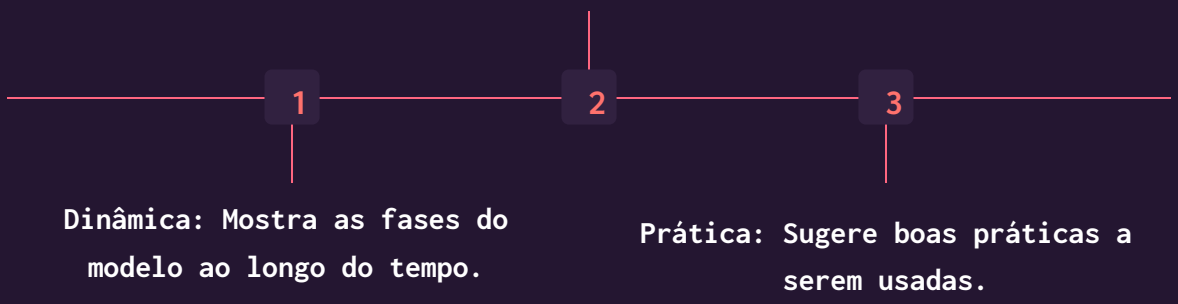
Rational Unified Process (RUP) - Um Exemplo de Processo Moderno

Processo da Rational Adquirido Pela IBM

Derivado de trabalhos sobre UML,



Lidando com Mudanças nos Processos



Rational Unified Process (RUP)

Concepção

O objetivo da fase de concepção é estabelecer um business case para o sistema.

Elaboração

As metas da fase de elaboração são desenvolver uma compreensão do problema dominante, estabelecer um framework da arquitetura para o sistema, desenvolver o plano do projeto e identificar os maiores riscos do projeto.

Construção

A fase de construção envolve projeto, programação e testes do sistema.

Transição

A fase final do RUP implica transferência do sistema da comunidade de desenvolvimento para a comunidade de usuários e em seu funcionamento em um ambiente real.

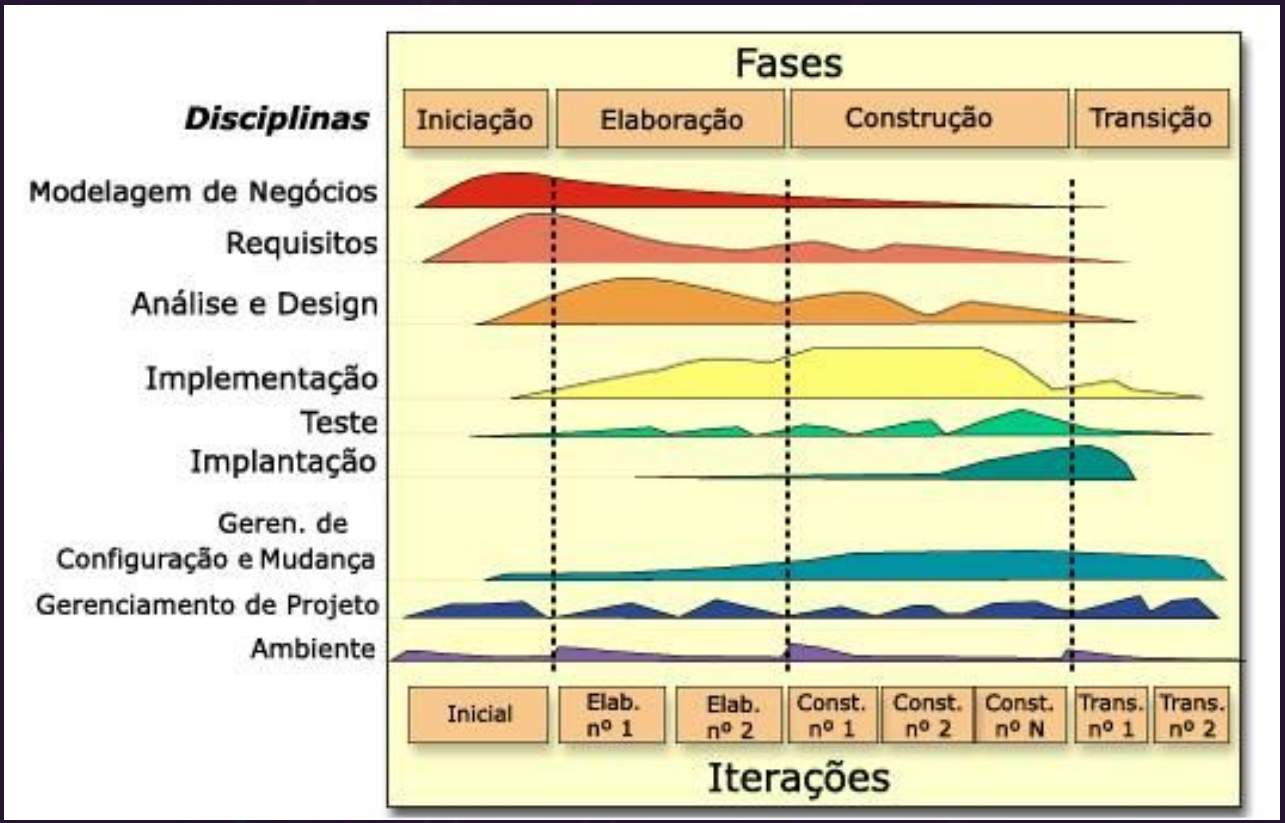


Tabela 2.1 *Workflows* estáticos no Rational Unified Process

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.

Tabela 2.1 *Workflows* estáticos no Rational Unified Process

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.

Exercícios
Teóricos & Prático