Processos de Software

Centro de Informática - Universidade Federal de Pernambuco
Kiev Gama
kiev@cin.ufpe.br

Slides originais elaborados por Ian Sommerville e adaptado pelos profs. Márcio Cornélio, Vinicius Garcia e Kiev Gama

UNIVERSIDADE FEDERAL DE PERNAMBUCO

O processo de software

- Um conjunto estruturado de atividades, procedimentos, artefatos e ferramentas necessários para o desenvolvimento de um sistema de software
  - Atividades: Especificação, Projeto, Validação, Evolução
- Exemplos: Processo Unificado (RUP), Programação Extrema, UML Components
- Um modelo de processo de software apresenta a descrição de um processo de uma perspectiva particular, normalmente focando apenas em algumas atividades.

Modelos genéricos de processo de software

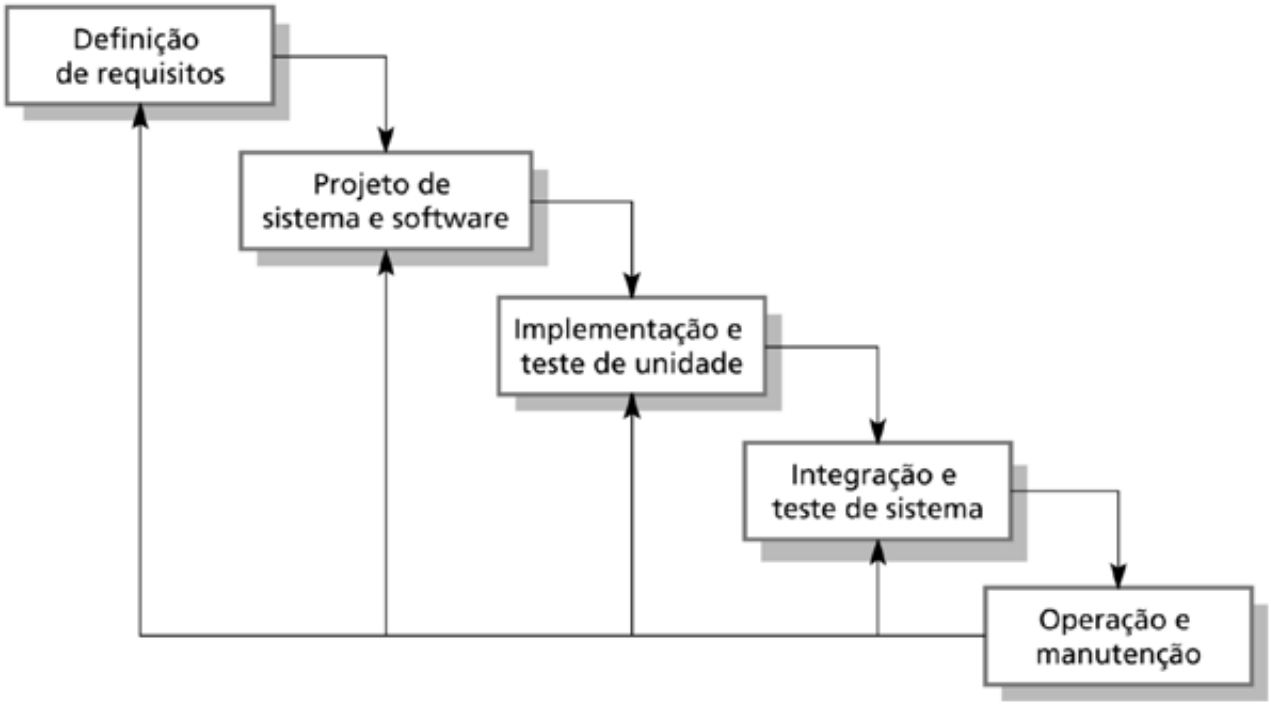
- O modelo cascata
  - Fases separadas e distintas de especificação e desenvolvimento.
- Engenharia de software baseada em componentes
  - O sistema é montado a partir de componentes existentes.
- Desenvolvimento iterativo
  - Sistema desenvolvido através de várias etapas
- Existem muitas variantes destes modelos
  - Ex: desenvolvimento formal onde um processo semelhante ao cascata é usado, mas a especificação formal é refinada durante os vários estágios para um projeto implementável.

Modelo cascata

Resposta ao modelo *code-and-fix* vigente na década de 70

Figura 4.1

Ciclo de vida de software.



Ian Sommerville, Engenharia de Software, 8ª. edição. Capítulo 4

Fases do modelo cascata

- Análise e definição de requisitos
- Projeto de sistema e software
- Implementação e teste de unidade
- Integração e teste de sistema
- Operação e manutenção

- Primeiro modelo a **organizar** as atividades de desenv.
- Uma fase tem de estar completa antes de passar para a próxima.
  - Saídas das fases são acordadas contratualmente!
- Todas as fases envolvem atividades de validação

Problemas do modelo cascata

- **Particionamento inflexível** do projeto em estágios
  - Dificulta a resposta aos requisitos de mudança do cliente.
- Documentos "completamente elaborados" são necessários para fazer as transições entre estágios
- Apropriado somente quando os requisitos são bem compreendidos e quando as **mudanças são raras**
  - Poucos sistemas de negócio têm requisitos estáveis.

Desenvolvimento evolucionário

- Implementação inicial, exposição do resultado aos comentários do usuário e refinamento desse resultado em versões
- Especificação, desenvolvimento e validação são intercaladas
- Dois tipos
  - Desenvolvimento exploratório: explora requisitos e entrega um sistema final
  - Prototipação *throwaway*: objetivo é compreender os requisitos do cliente

Desenvolvimento evolucionário

- Vantagem: especificação desenvolvida de forma incremental
- Problemas:
  - Processo não é visível
  - Sistemas podem ser mal estruturados devido à mudança contínua

Desenvolvimento Exploratório

- Idéia geral:
  - Desenvolvimento da primeira versão do sistema o mais rápido possível
  - Modificações sucessivas até que o sistema seja considerado adequado
  - Após o desenvolvimento de cada uma das versões do sistema ele é mostrado aos usuários para comentários
- Adequado para o desenvolvimento de sistemas onde é difícil ou impossível se fazer uma especificação detalhada do sistema

Prototipação Descartável

- Como na programação exploratória, a primeira fase prevê o desenvolvimento de um programa para o usuário experimentar
  - No entanto, o objetivo aqui é estabelecer os requisitos do sistema
  - O software deve ser reimplementado na fase seguinte
- A construção de protótipos com os quais os usuários possam brincar é uma idéia bastante atrativa:
  - Para sistemas grandes e complicados
  - Quando não existe um sistema anterior ou um sistema manual que ajude a especificar os requisitos

Prototipação Descartável

- Os objetivos do protótipo devem estar bem claros antes do início da codificação. Possíveis objetivos:
  - Entender os requisitos dos usuários
  - Definir a interface com os usuários
  - Demonstrar a viabilidade do sistemas para os gerentes.
- Uma decisão importante a ser tomada é escolher o que será e o que não será parte do protótipo
  - Não é economicamente viável implementar todo o sistema!
  - Os objetivos do protótipo são o ponto de partida

Engenharia de software baseada em componentes

- Baseado em reuso sistemático onde sistemas são integrados a partir de componentes existentes ou de sistemas COTS (*Commercial-of-the-shelf*)
- Estágios do processo
  - Análise de componentes;
  - Modificação de requisitos;
  - Projeto de sistema com reuso;
  - Desenvolvimento e integração.

- Esta abordagem está se tornando cada vez mais usada à medida que padrões de componentes têm surgido.
- Reuso **acidental** vs. Reuso **planejado**

Processos Iterativos

- Requisitos de sistema **SEMPRE** evoluem no curso de um projeto
- Algum retrabalho é necessário
- A **abordagem iterativa** pode ser aplicada a qualquer um dos modelos genéricos de processo
- Duas abordagens (relacionadas)
  - Entrega incremental;
  - Desenvolvimento espiral.
- Essência: especificação desenvolvida em conjunto com o software

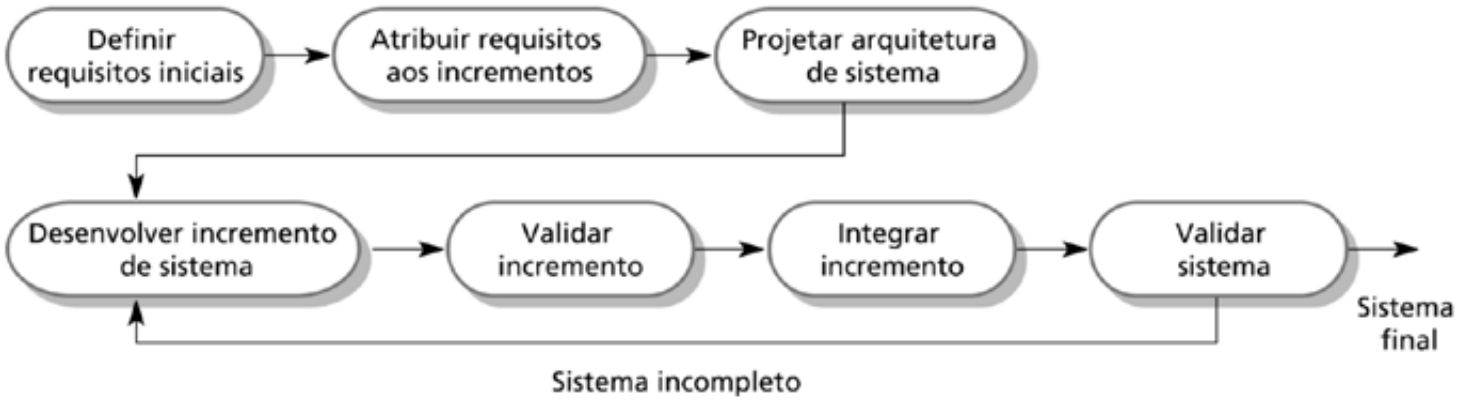
Entrega incremental

- O sistema é entregue ao cliente em **incrementos**
  - Cada incremento fornece parte da funcionalidade
- Os requisitos são **priorizados**
  - Requisitos de prioridade mais alta são incluídos nos incrementos iniciais.
- Uma vez que o desenvolvimento de um incremento é iniciado, os requisitos são congelados
  - Os requisitos para os incrementos posteriores podem continuar evoluindo (e incluir requisitos já implementados!)

Desenvolvimento incremental

Figura 4.4

Entrega incremental.



Ian Sommerville, Engenharia de Software, 8ª. edição. Capítulo 4

Vantangens do desenvolvimento incremental

- Incrementos podem ser entregues regularmente ao cliente e, desse modo, a funcionalidade de sistema é disponibilizada mais **cedo**.
- Os incrementos iniciais agem como **protótipos** para 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**.

Problemas do desenvolvimento incremental

- Incrementos pequenos exigem mapeamento entre requisitos e incremento de tamanho adequado

- Dificuldade de identificar os recursos comuns exigidos por todos os incrementos

Extreme programming

- Uma abordagem baseada no desenvolvimento e na entrega de incrementos de funcionalidade muito pequenos.

- Baseia-se no aprimoramento constante do código, em testes automatizados, no envolvimento do usuário na equipe e no desenvolvimento em pares.

Desenvolvimento espiral

- O processo é representado como uma espiral ao invés de uma sequência de atividades com realimentação.
- Cada loop na espiral representa uma fase no processo.
- Sem fases definidas, tais como especificação ou projeto – os *loops* na espiral são escolhidos dependendo do que é requisitado.
- Os **riscos** são explicitamente avaliados e resolvidos ao longo do processo.

Modelo espiral do processo de software

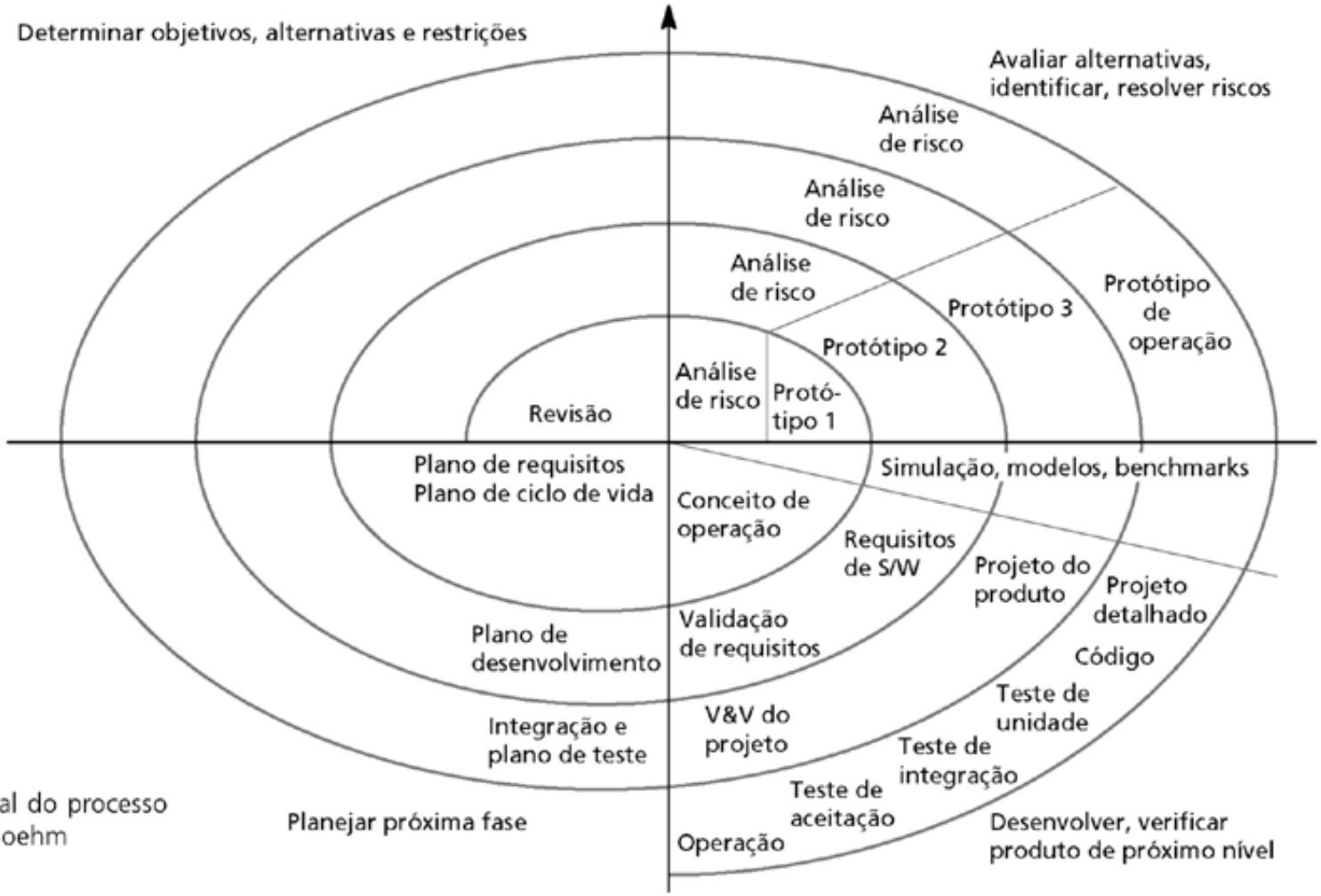


Figura 4.5

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

Ian Sommerville, Engenharia de Software, 8ª. edição. Capítulo 4

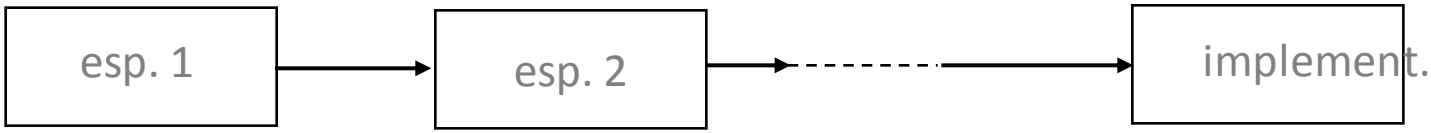
Setores do modelo espiral

- 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
  - O projeto é revisado e a próxima fase da espiral é planejada.

- Processo de **Desenvolvimento** vs. Processo de **Gerenciamento**

Transformação Formal

- Idéia geral:
  - Uma especificação formal (definição matemática, não ambígua) do software é desenvolvida e posteriormente "transformada" em um programa através de regras que preservam a corretude da especificação



Transformação Formal

- A grande motivação por trás da ideia de refinamento formal é a possibilidade de gerar programas que são corretos por construção
  - O próprio processo de desenvolvimento deve garantir que o programa faz exatamente o que foi especificado
- Este modelo tem sido aplicado ao desenvolvimento de sistemas críticos, especialmente naqueles onde a segurança é um fator crítico (ex: sistema de controle de ferrovias)

ISO 12207
Processos de Ciclo de vida do Software

- Norma internacional que identifica quais os "processos" no ciclo de vida do software
- Processos fundamentais
  - Aquisição
  - Fornecimento
  - Desenvolvimento
  - Operação
  - Manutenção
- O que é chamado de "processo" nesta norma, seria mais ou menos correspondente ao que é visto como "atividade" em Sommerville

Atividades de um processo de desenvolvimento (Sommerville)

- Especificação de software
- Projeto e implementação de software
- Validação de software
- Evolução de software

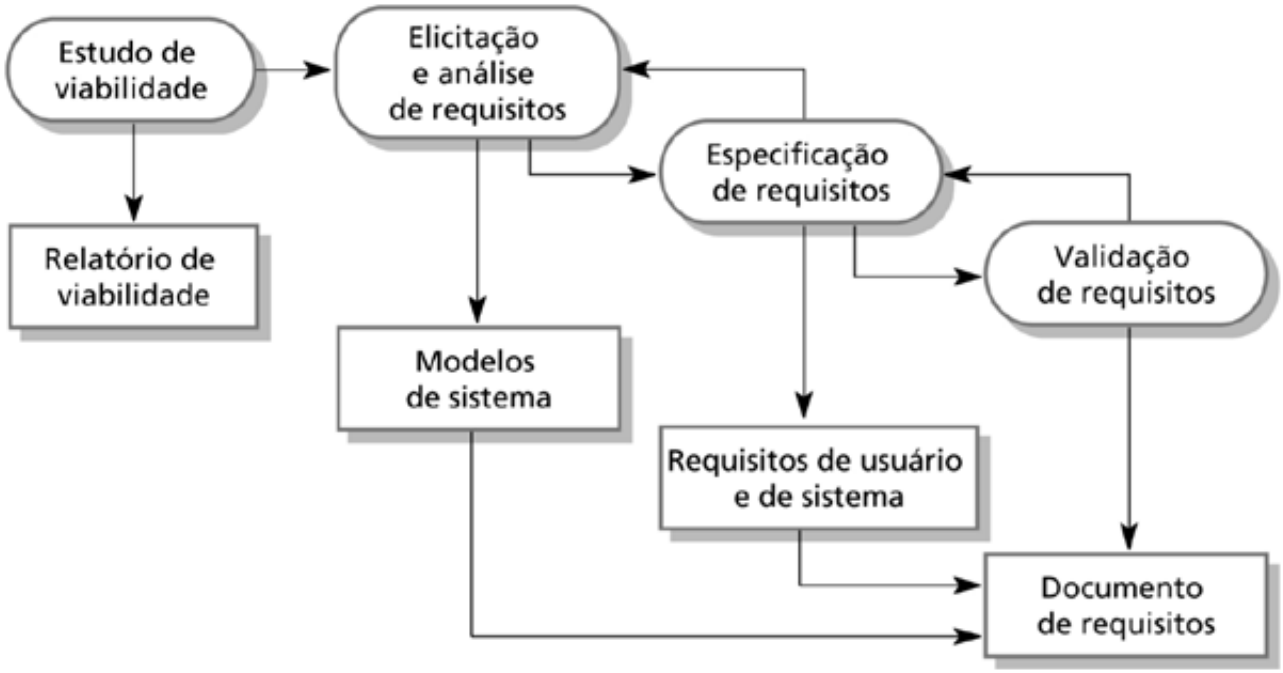
Especificação de software

- O processo para definir quais **serviços** são necessários e identificar as **restrições** de operação e de desenvolvimento do sistema.
- Processo de engenharia de requisitos
  - Estudo de viabilidade;
    - Realizado **antes** do projeto ser iniciado
  - Elicitação e análise de requisitos;
  - Especificação de requisitos;
    - Requisitos do usuário (mais abstrato) e requisitos do sistema (descrição detalhada de funcioalidades)
  - Validação de requisitos.

O processo de engenharia de requisitos

Figura 4.6

Processo de engenharia de requisitos.



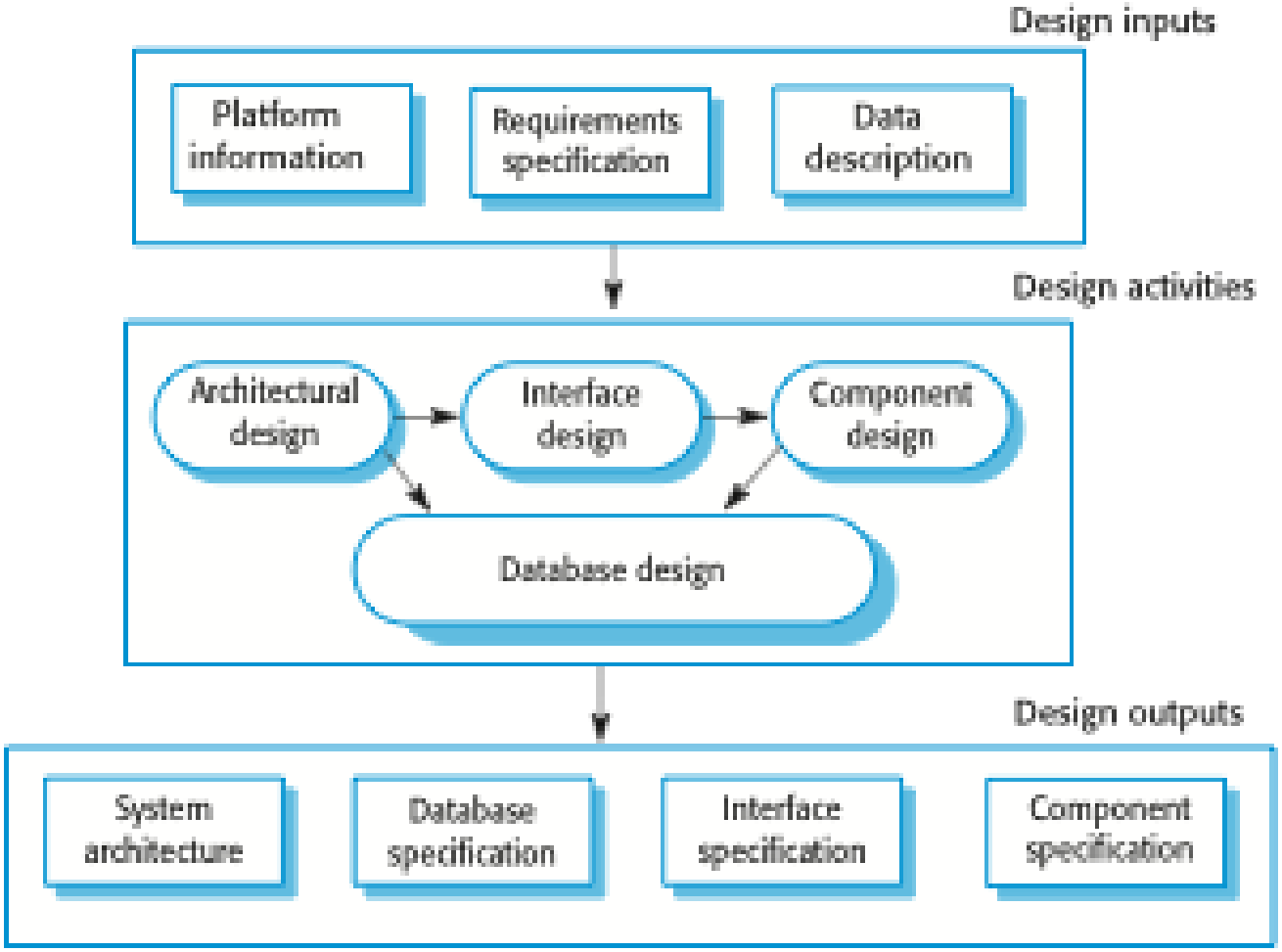
Também pode envolver a **prototipação** de partes do sistema!

Ian Sommerville, Engenharia de Software, 8ª. edição. Capítulo 4

Projeto e implementação de software

- É o processo de conversão da especificação em um sistema de software
- Projeto de software
  - Projetar uma estrutura de software que atenda à especificação.
- Implementação
  - Transformar essa estrutura em um programa executável.
- As atividades de projeto e implementação são fortemente relacionadas e podem ser intecaladas.

Modelo genérico do processo de projeto de sistema



Atividades específicas do processo de projeto

- Projeto de arquitetura
  - Subsistemas e relacionamento
- Especificação abstrata
  - Especificação abstrata de serviços e restrições de subsistemas
- Projeto de interfaces entre componentes
- Projeto de componente
- Projeto de estrutura de dados
- Projeto de algoritmo

Métodos estruturados

- Abordagens **sistemáticas** para projetar sistemas de software
  - *Project* (**gerenciamento**) vs. *Design* (**desenvolvimento**)
- O projeto é, em geral, documentado como um conjunto de modelos gráficos
- Modelos possíveis
  - Modelo de objeto;
  - Modelo de sequência;
  - Modelo de transição de estado;
  - Modelo estruturado;
  - Modelo de fluxo de dados.

Programação e depuração

- É a transformação de um projeto em um programa e a remoção de defeitos desse programa.
- Programação é uma atividade pessoal – não há processo genérico de programação.
  - Há algumas práticas, porém, que são universalmente consideradas **boas**
- Programadores realizam alguns **testes** para descobrir defeitos no programa e removem esses defeitos no processo de **depuração**

Validação de software

- Verificação e validação (V & V) têm a intenção de mostrar que um sistema está em conformidade com a sua especificação e que atende aos requisitos do cliente
- **Verificação**: "construímos o sistema corretamente?"
  - Exs: inspeção de código, análise estática
- **Validação**: "construímos o sistema correto?"
  - Exs: testes, animação de especificações
- Testes envolvem a execução do sistema com casos de teste que são derivados da especificação do sistema e de dados reais a ser processados por ele.

Estágios de teste

- Teste de componente ou unidade
  - Os componentes individuais são testados independentemente;
  - Esses componentes podem ser funções ou classes de objetos, ou grupos coerentes dessas entidades.
- Teste de sistema
  - Teste de sistema como um todo. O teste das propriedades emergentes é particularmente importante.
  - Busca erros que resultam de interações não previstas entre componentes
- Teste de aceitação
  - Teste com dados do cliente para verificar se o sistema atende às suas necessidades, ou seja, pode revelar problemas de requisitos

Fases de teste

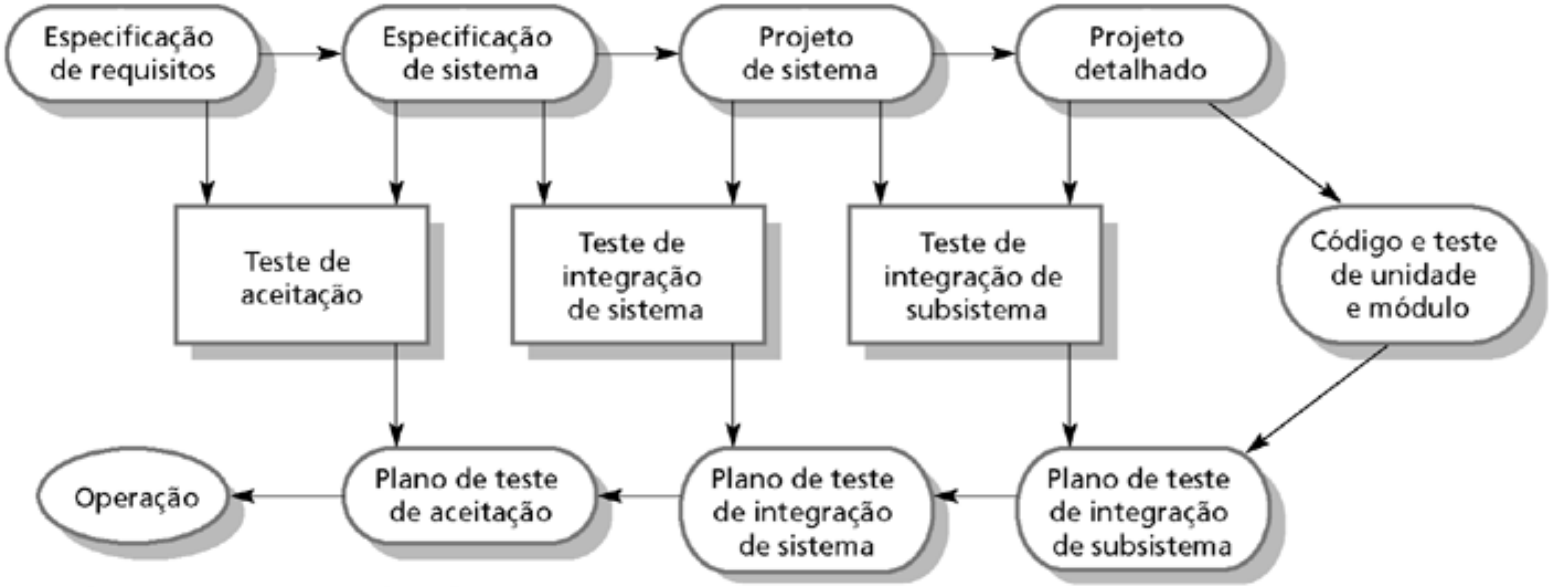


Figura 4.10 Fases de teste no processo de software.

Ian Sommerville, Engenharia de Software, 8ª. edição. Capítulo 4

Evolução de software

- O software é inerentemente **flexível** e pode mudar
- Requisitos mudam devido a diversos fatores e o software deve acompanhar essas mudanças
- Processos antigos separavam explicitamente desenvolvimento de evolução
  - Processos e métodos iterativos (XP, RUP, Espiral) normalmente **não fazem uma sepação explícita**
- Evolução pode se dever a diversas razões:
  - Correções (patches)
  - Mudanças de requisitos
  - Melhoria de funcionalidades pré-existentes

Evolução de software

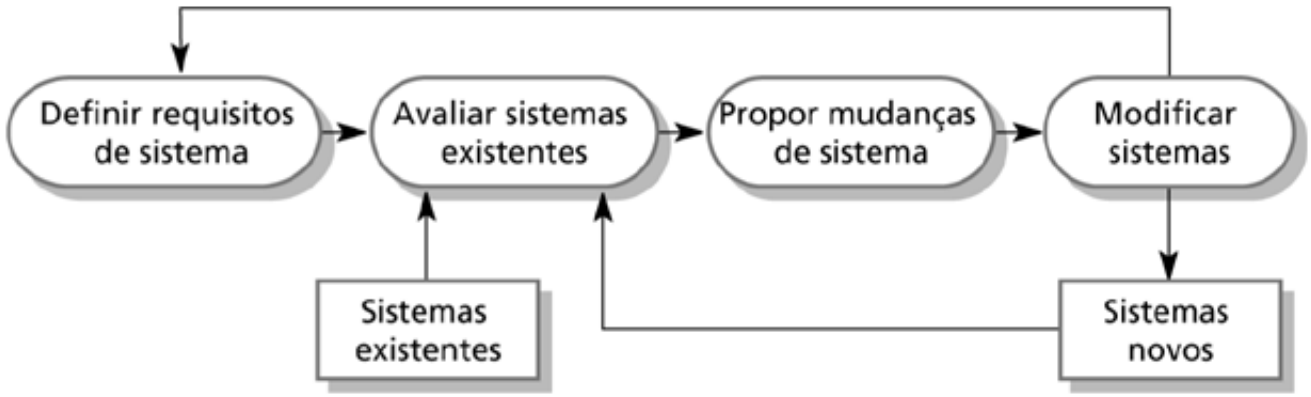


Figura 4.11

Evolução de sistema.

Ian Sommerville, Engenharia de Software, 8ª. edição. Capítulo 4

(Rational) Unified Process

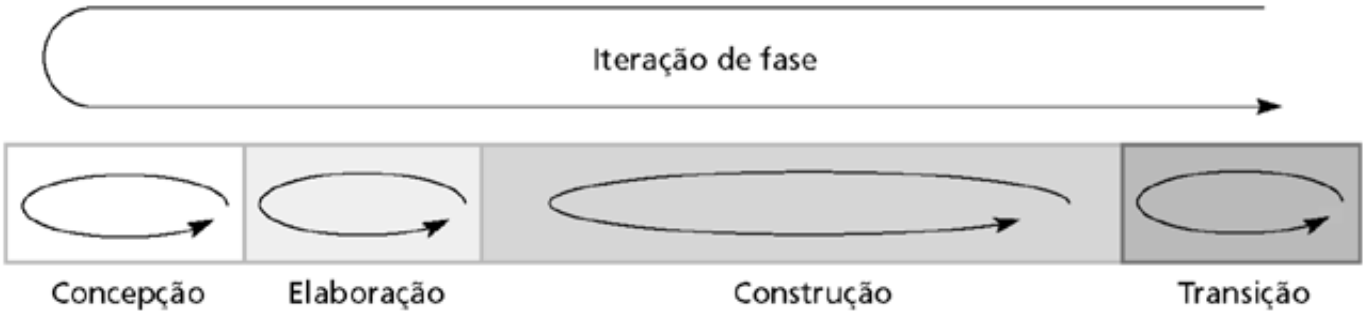
- É um (“modelo de”?) processo moderno baseado na UML
  - Tenta cobrir todos os aspectos do desenvolvimento de software
- Fortemente focado na **documentação** do sistema
- Normalmente descrito a partir de três perspectivas:
  - Uma perspectiva dinâmica que mostra as **fases** ao longo do tempo;
  - Uma perspectiva estática que mostra **atividades** de processo;
  - Uma perspectiva prática que sugere bons **princípios** e **práticas** de desenvolvimento

Modelo de fases do RUP

Centrado no gerenciamento de projetos

Figura 4.12

Fases no Rational Unified Process.

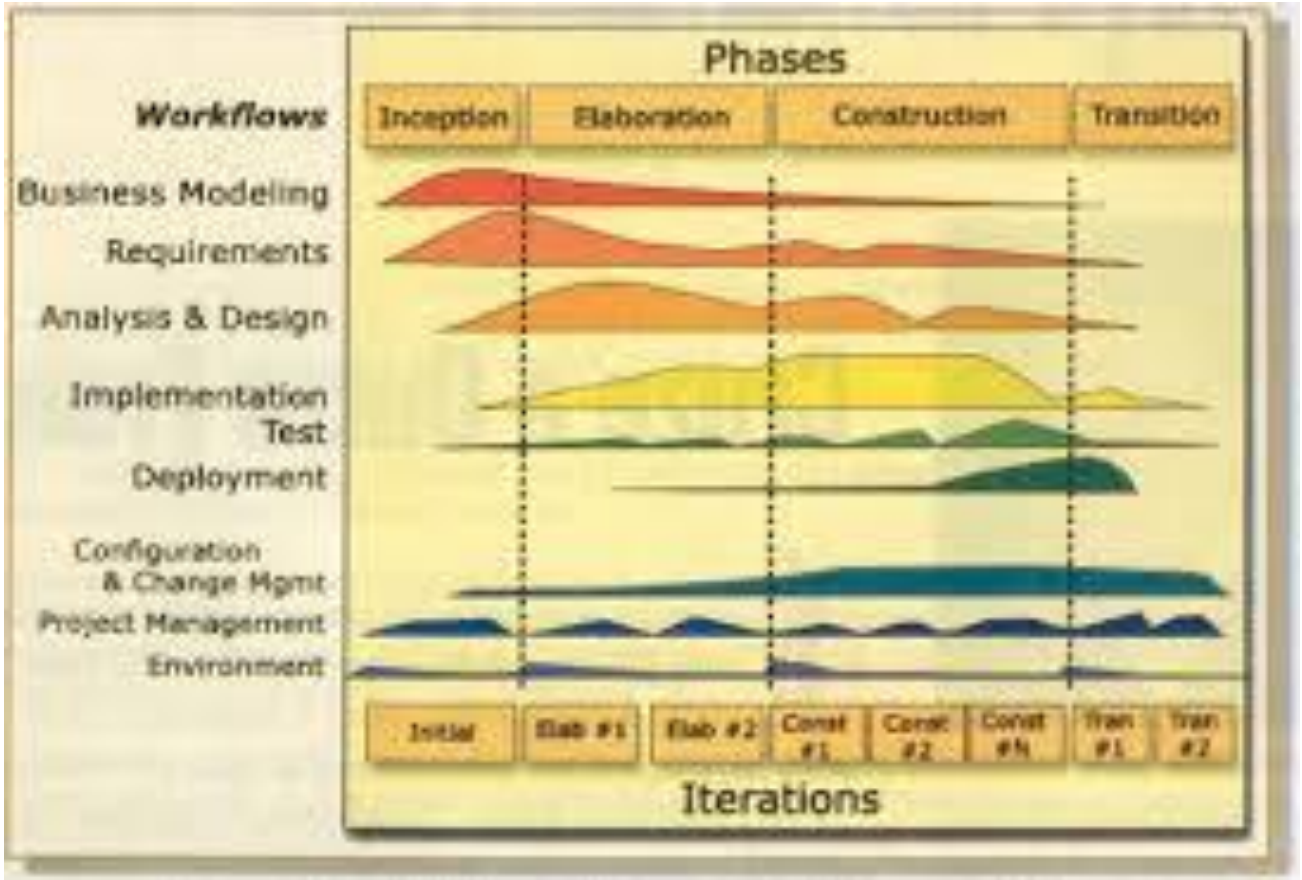


Ian Sommerville, Engenharia de Software, 8ª. edição. Capítulo 4

Fases do RUP

- **Concepção**
  - Estabelecer o business case para o sistema.
- **Elaboração**
  - Desenvolver um entendimento do domínio do problema, arquitetura do sistema e identificar riscos
- **Construção**
  - Projeto, programação, teste de sistema e documentação
- **Transição**
  - Implantar o sistema no seu ambiente operacional.

Fases do RUP


Phases
Workflows
Inception
Elaboration
Construction
Transition
Business Modeling
Requirements
Analysis & Design
Implementation
Test
Deployment
Configuration
& Change Mgmt
Project Management
Environment
Initial
Elab #1
Elab #2
Const #1
Const #2
Const #N
Tran #1
Tran #2
Iterations

Workflows estáticos

Tabela 4.1 Workflows estáticos no Rational Unified Process.

Workflow	Descrição
Modelagem de negócios	Os processos de negócios são modelados usando casos de uso de negócios.
Requisitos	Os agentes que interagem com o sistema são identificados e os casos de uso são desenvolvidos para modelar os requisitos de sistema.
Análise e projeto	Um modelo de projeto é criado e documentado usando modelos de arquitetura, modelos de componente, modelos de objeto e modelos de seqüência.
Implementação	Os componentes de sistema são implementados e estruturados em subsistemas de implementação. A geração automática de código com base nos modelos de projeto ajuda a acelerar esse processo.
Teste	O teste é um processo iterativo realizado em conjunto com a implementação. O teste de sistema segue o término da implementação.
Implantação	Uma versão do produto é criada, distribuída aos usuários e instalada no local de trabalho.
Gerenciamento de configuração e mudanças	Este workflow de apoio gerencia as mudanças do sistema (veja o Capítulo 29).
Gerenciamento de projetos	Este workflow de apoio gerencia o desenvolvimento do sistema (veja o Capítulo 5).
Ambiente	Este workflow está relacionado à disponibilização de ferramentas apropriadas de software para a equipe de desenvolvimento.

Ian Sommerville, Engenharia de Software, 8ª. edição. Capítulo 4

Boas práticas do RUP

- Desenvolver o software iterativamente
- Gerenciar requisitos
- Usar arquiteturas baseadas em componentes
- Modelar o software visualmente
- Verificar a qualidade de software
- Controlar as mudanças do software

Leituras recomendadas

- SOMMERVILLE, I. Engenharia de Software. 9ª. Ed. São Paulo: Pearson Education, 2011
  - Capítulo 4

- Referência adicional
  - Barry W. Boehm. **A Spiral Model of Software Development and Enhancement**. IEEE Computer, vol. 21, número 5, Maio de 1988.
    - http://dx.doi.org/10.1109/2.59