Capítulo 2

Processos de Software

Tópicos apresentados

- Modelos de processo de software.
- Atividades de processo.
- Lidando com mudanças.
- Rational Unified Process (RUP).
- Um exemplo de um processo de desenvolvimento de software moderno.

O processo de software

- Um conjunto estruturado de atividades necessárias para desenvolver um sistema de software.

- Existem vários processos de desenvolvimento de software diferentes mas todos envolvem:

  - ✓ especificação – definição do quê o sistema deve fazer;
  - ✓ projeto e implementação – definição da organização do sistema e implementação do sistema;
  - ✓ validação – checagem de que o sistema faz o que o cliente deseja;
  - ✓ evolução – evolução em resposta a mudanças nas necessidades do cliente.

- Um modelo de processo de desenvolvimento de software é uma representação abstrata de um processo. Ele apresenta uma descrição do processo de uma perspectiva em particular.

Descrições de processo de software

- Quando descrevemos e discutimos processos, geralmente falamos sobre as atividades desses processos, tais como especificação de modelo de dados, 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 antes e depois de uma atividade do processo ser executada, ou um produto produzido.

Processos dirigidos a planos e ágeis

- Processos dirigidos a planos são processos em que todas as atividades do processo são planejadas com antecedência e o progresso é medido em relação a esse plano.

- Nos processos ágeis o planejamento é incremental e é mais fácil modificar o processo para refletir alterações nos requisitos do cliente.

- Na realidade, os processos mais práticos incluem elementos dos processos ágeis e dirigidos a planos.

- Não existe processo de software certo ou errado.

Modelos de processo de software

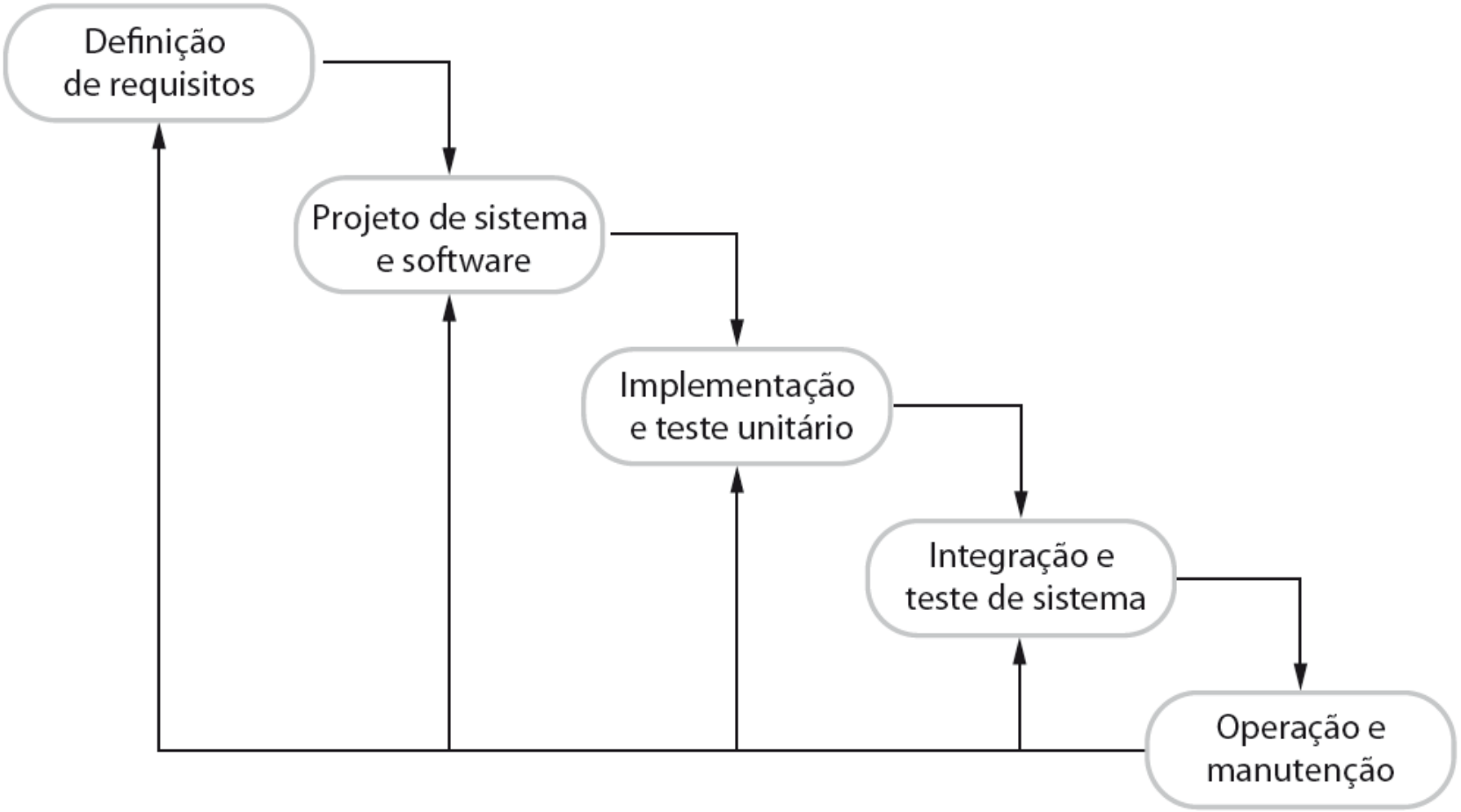
- Modelo Cascata – Modelo dirigido a planos. Fases de especificação e desenvolvimento separadas e distintas.

- Desenvolvimento Incremental – Especificação, desenvolvimento e validação são intercaladas. Pode ser dirigido a planos ou ágil.

- Engenharia de software orientada a reúso – O sistema é montado a partir de componentes já existentes. Pode ser dirigido a planos ou ágil.

Na realidade a maioria dos grandes sistemas são desenvolvidos usando um processo que incorpora elementos de todos esses modelos.

O modelo cascata



Fases do modelo cascata

- Existem fases identificadas e separadas no 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
- O principal inconveniente do modelo cascata é a dificuldade de acomodação de mudanças depois que o processo já foi iniciado. Em princípio, uma fase precisa ser completada antes de se mover para a próxima fase.

Problemas do modelo cascata

- Divisão inflexível do projeto em estágios distintos torna difícil responder às mudanças nos requisitos do cliente.

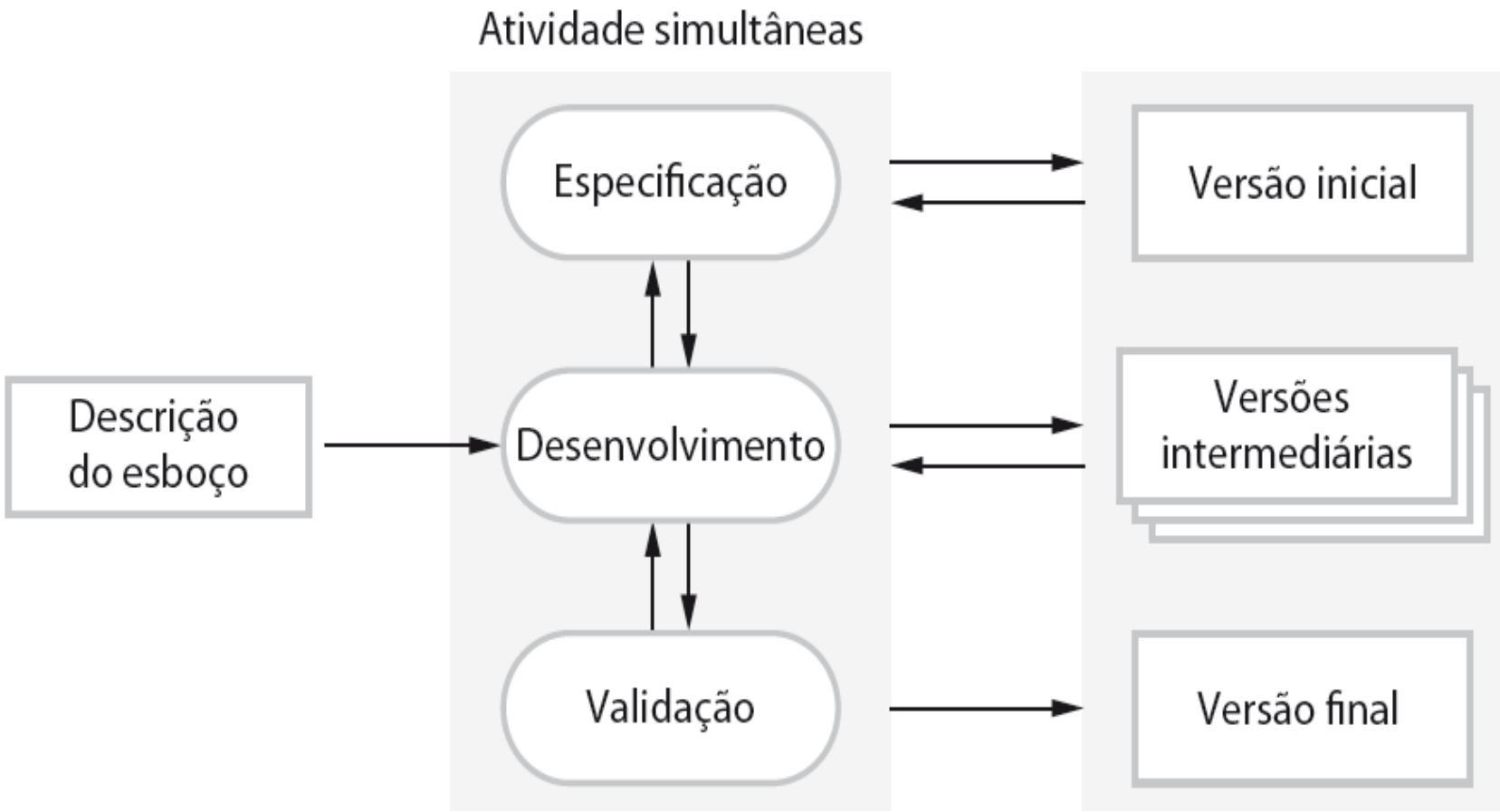
  - ✓ Por isso esse modelo só é apropriado quando os requisitos são bem entendidos e as mudanças durante o processo de projeto serão limitadas.

  - ✓ Poucos sistemas de negócio possuem requisitos estáveis.

- O modelo cascata é mais usado em projetos de engenharia de grandes sistemas onde o sistema é desenvolvido em vários locais.

  - ✓ Nessas circunstâncias, a natureza do modelo cascata dirigida a planos ajuda a coordenar o trabalho.

Desenvolvimento incremental



Benefícios do desenvolvimento incremental

- O custo para acomodar mudanças nos requisitos do cliente é reduzido.
  - ✓ A quantidade de análise e documentação que precisa ser feita é bem menor do que o necessária no modelo cascata.
- É mais fácil obter feedback do cliente sobre o trabalho de desenvolvimento que tem sido feito.
  - ✓ Os clientes podem comentar demonstrações do software e ver quanto foi implementado.
- Possibilidade de mais rapidez na entrega e implantação de software útil para o cliente.
  - ✓ Os clientes podem usar e obter ganhos do software mais cedo do que é possível no processo cascata.

Problemas do desenvolvimento incremental

- O processo não é visível.
  - ✓ Gerentes precisam de entregas regulares para medir o progresso. Se os sistemas são desenvolvidos de forma rápida, não é viável do ponto de vista do custo produzir documentação para refletir todas as versões do sistema.
- A estrutura do sistema tende a degradar conforme novos incrementos são adicionados.
  - ✓ A menos que tempo e dinheiro sejam gastos na reconstrução para melhorar o software, as mudanças regulares tendem a corromper a estrutura do sistema. A incorporação posterior de mudanças no software se torna progressivamente mais difícil e cara.

Engenharia de software orientada a reúso

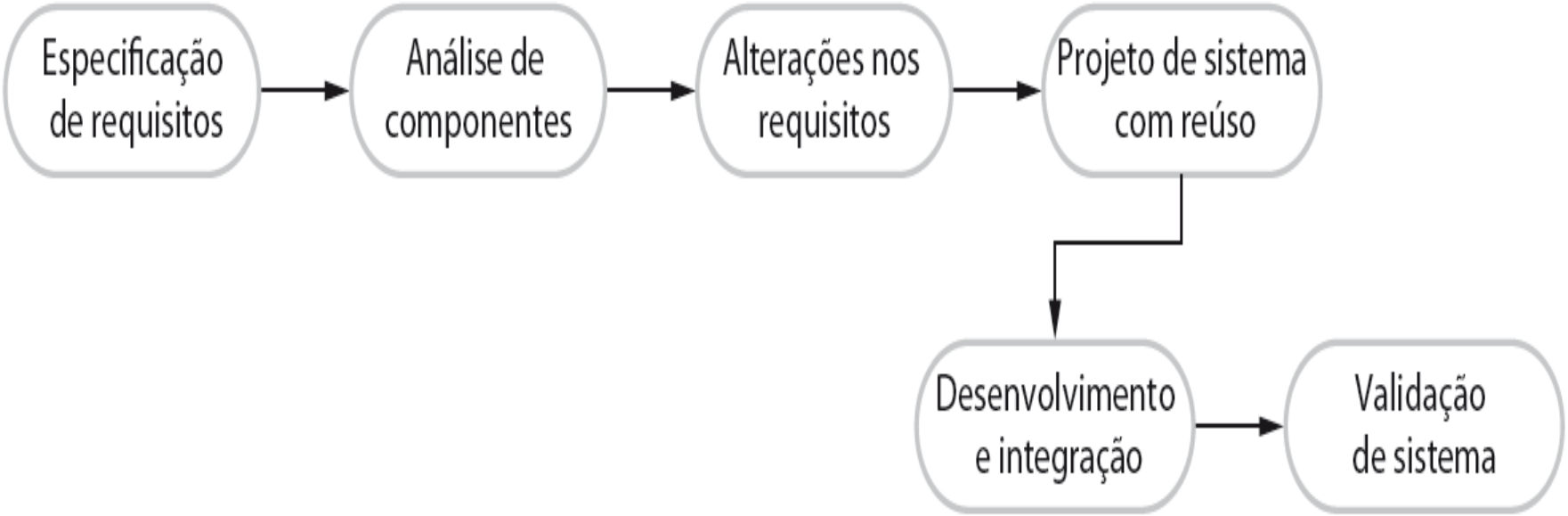
- Baseada no reúso sistemático em que os sistemas são integrados com componentes existentes ou sistemas COTS (Commercial-off-the-shelf).

- Estágios do processo:

  - ✓ Análise de componentes;
  - ✓ Modificação de requisitos;
  - ✓ Projeto de sistema com reúso;
  - ✓ Desenvolvimento e integração.

- Atualmente, o reúso é a abordagem padrão para a construção de vários tipos de sistemas de negócio.

Engenharia de software orientada a reúso



Tipos de componente de software

- Web services que são desenvolvidos de acordo com padrões de serviço e ficam disponíveis para chamada remota.

- Coleções de objetos que são desenvolvidas como um pacote para ser integrado com um framework como .NET ou J2EE.

- Sistemas de software stand-alone (COTS) que são configurados para uso em ambientes específicos.

Atividades do processo

- Processos de software reais são sequências intercaladas de atividades técnicas, colaborativas e gerenciais com o objetivo geral de especificar, projetar, implementar e testar um sistema de software.

- As quatro atividades de processo básicas, especificação, desenvolvimento, validação e evolução são organizadas de forma diferente em processos de desenvolvimento distintos.

- No modelo cascata, elas são organizadas em sequências, enquanto no desenvolvimento incremental são intercaladas.

Especificações de software

- O processo de estabelecer quais serviços são necessários e as restrições na operação e desenvolvimento do sistema.

- Processo de engenharia de requisitos

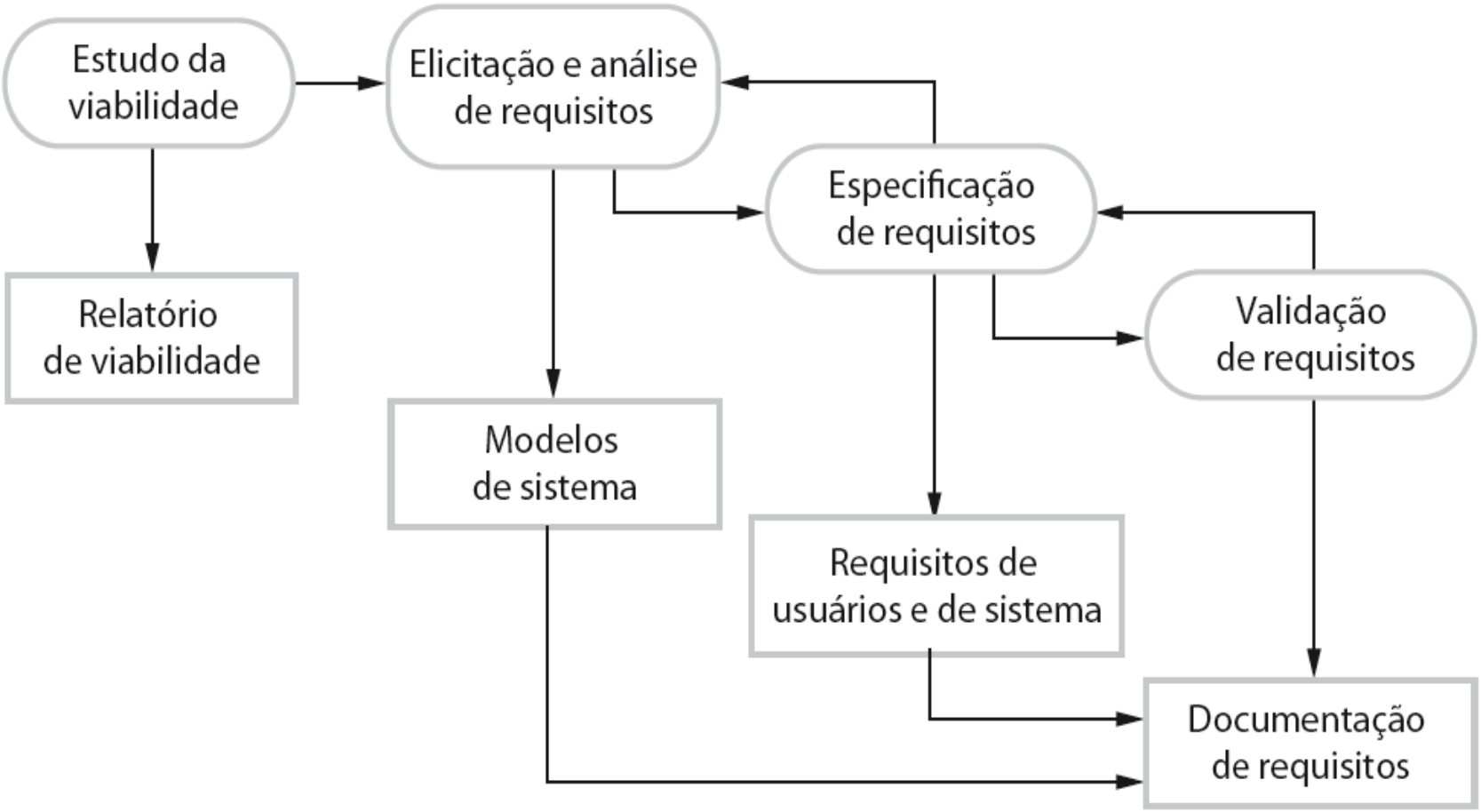
  ✓ Estudo de viabilidade
  É técnica e financeiramente viável construir o sistema?

  ✓ Elicitação e análise de requisitos
  O que os stakeholders do sistema precisam ou esperam do sistema?

  ✓ Especificação de requisitos
  Definição dos requisitos em detalhes.

  ✓ Validação de requisitos
  Verificação da completude dos requisitos.

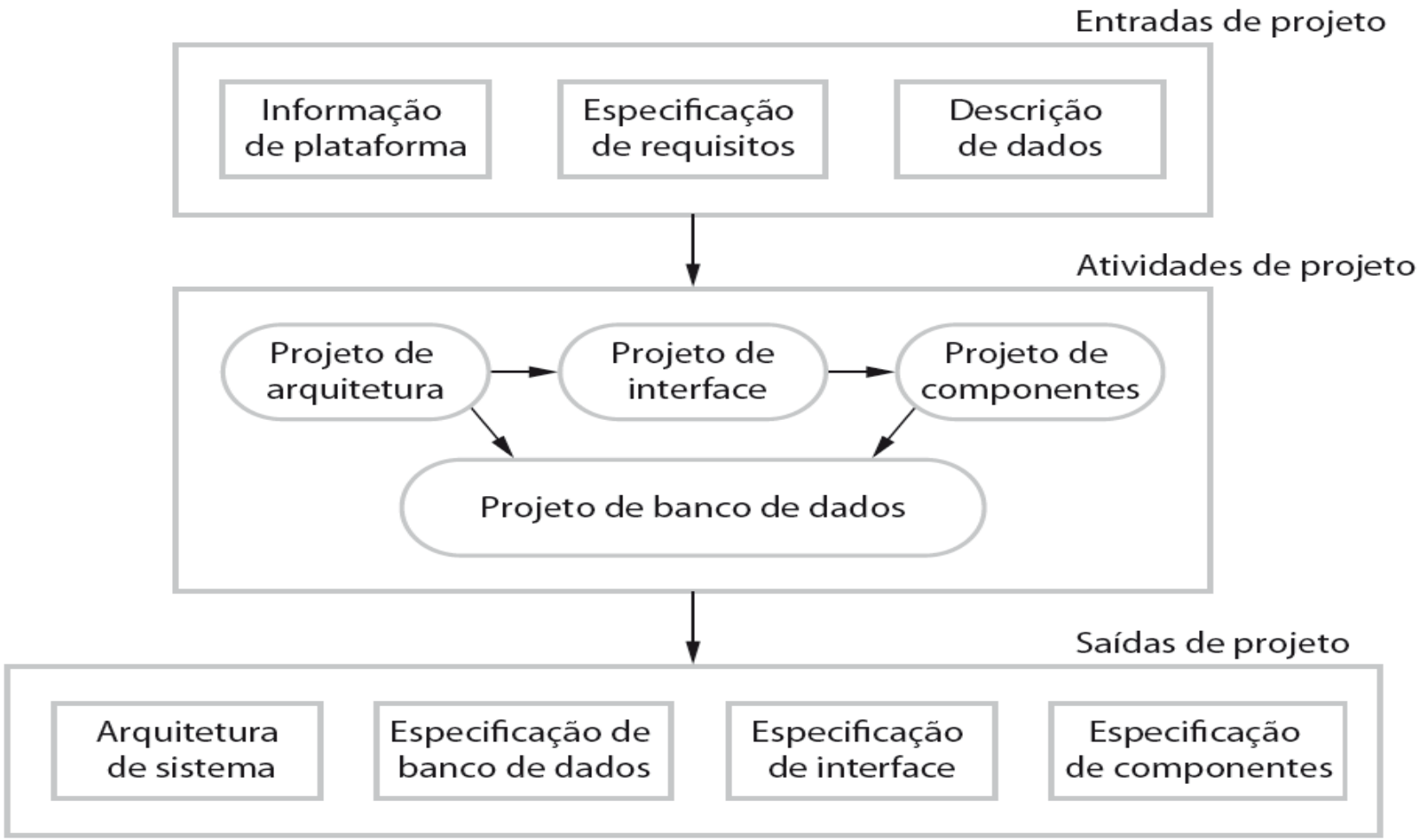
O processo de engenharia de requisitos



Projeto e implementação de software

- O processo de converter a especificação de sistema em um sistema executável.
- Projeto de software
  - ✓ Design de uma estrutura de software que materialize a especificação;
- Implementação
  - ✓ Transformar essa estrutura em um programa executável;
- As atividades de projeto e implementação são intimamente ligadas e podem ser intercaladas.

Modelo geral do processo de design



Atividades de projeto

- *Projeto de arquitetura,* em que você identifica a estrutura geral do sistema, os componentes principais (as vezes chamados sub-sistemas ou módulos), seus relacionamentos e como são distribuídos.

- *Projeto de interface,* em que você define as interfaces entre os componentes do sistema.

- *Projeto de componente,* em que você projeta como cada componente do sistema irá operar separadamente.

- *Projeto de banco de dados,* em que você projeta as estruturas de dados do sistema e como essas serão representadas no banco de dados.

Validação de software

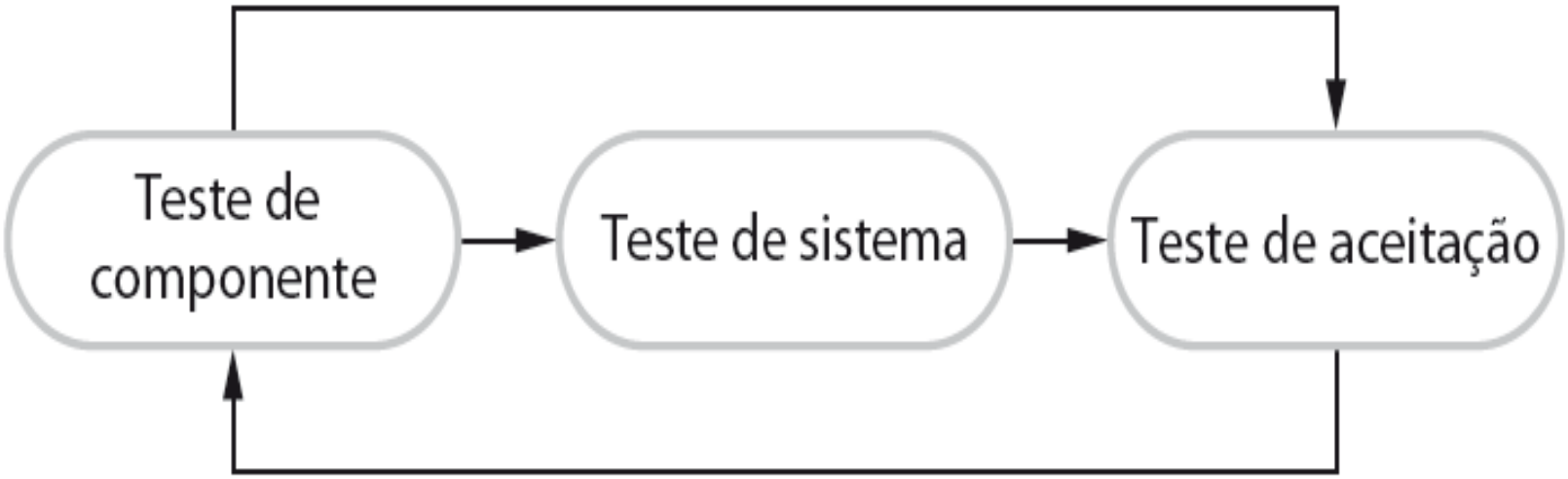
- Verificação e validação (V & V) serve para mostrar que o sistema está em conformidade com sua especificação e está de acordo com os requisitos do cliente.

- Envolve processos de inspeção e revisão, e testes do sistema.

- Testes do sistema envolvem executar o sistema com casos de teste. São provenientes de especificações dos dados reais que deverão ser processados pelo sistema.

- O teste é a atividade de V & V mais usada.

Estágios de teste



Estágios de teste

- Teste de desenvolvimento ou de componente

  - ✓ Componentes individuais são testados independentemente;
  - ✓ Componentes podem ser funções ou objetos, ou agrupamentos coerentes dessas entidades.

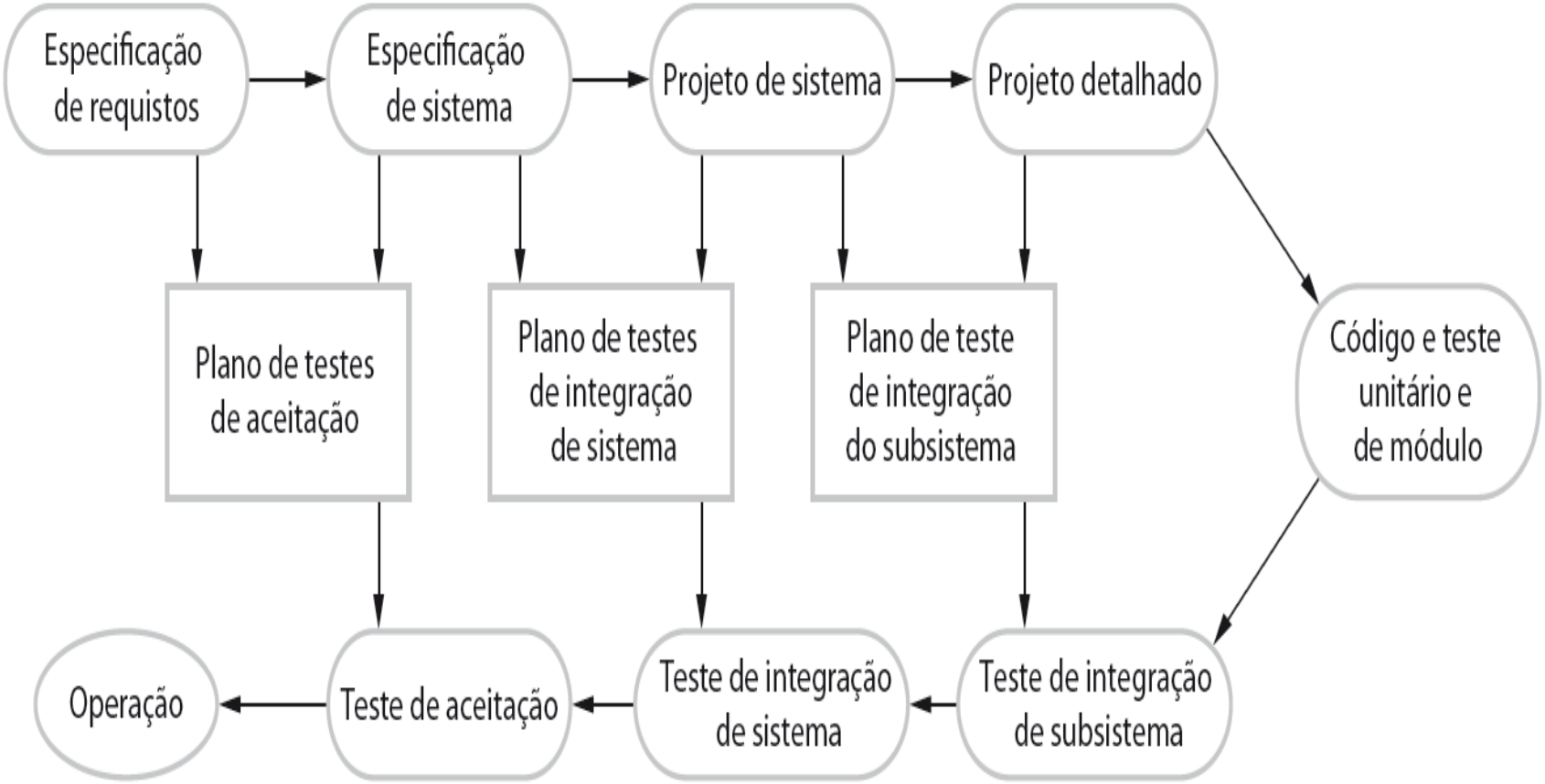
- Teste de sistema

  - ✓ Teste do sistema como um todo. Teste de propriedades emergentes são particularmente importantes.

- Teste de aceitação

  - ✓ Teste com dados do cliente para checar se o sistema está de acordo com as necessidades do cliente.

Fases de teste em um processo de software dirigido a planos



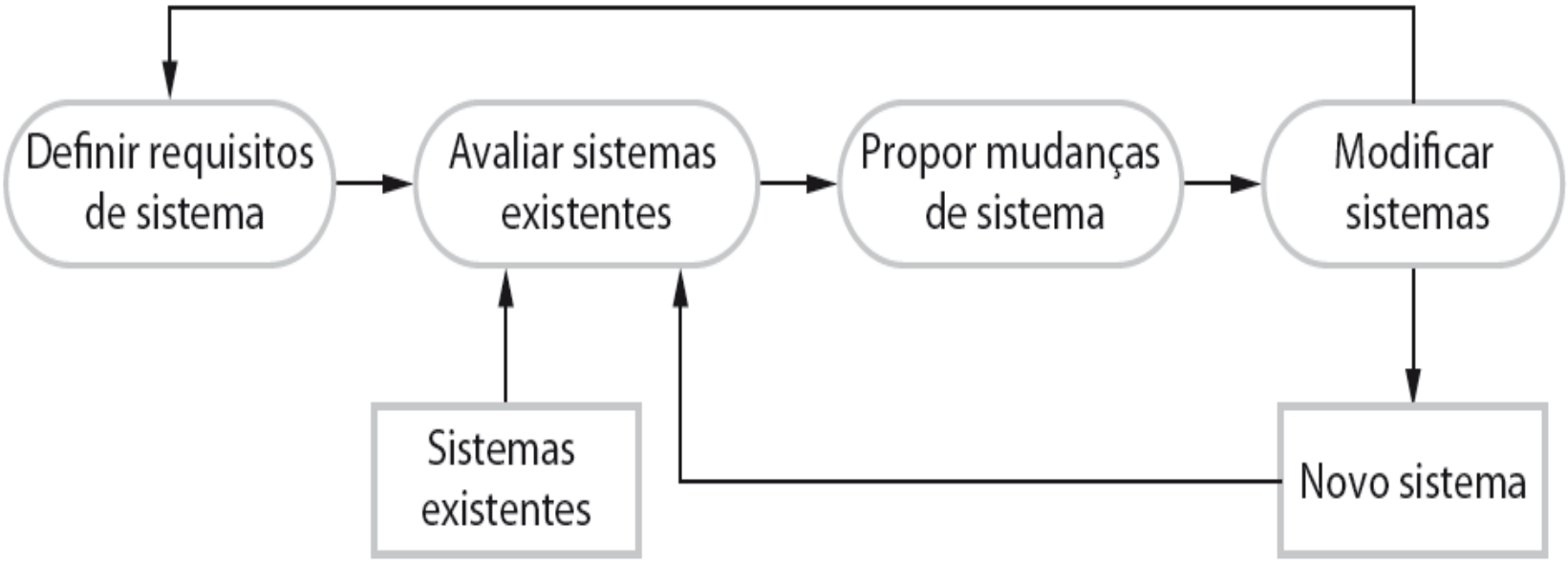
Evolução do software

- Os softwares são inerentemente flexíveis e podem mudar.

- Conforme os requisitos mudam, conforme mudam as circustâncias do negócio, o software que dá suporte ao negócio também deve evoluir e mudar.

- Apesar de ter acontecido uma demarcação entre desenvolvimento e evolução (manutenção) essa precisa se tornar cada vez mais irrelevante já que tem diminuído a quantidade de sistemas completamente novos.

Evolução do sistema



Pontos Importantes

- Os processos de software são as atividades envolvidas na produção de um sistema de software. Os modelos de processo de software são representações abstratas desses processos.

- Modelos de processo gerais descrevem a organização dos processos de software. Exemplos desses processos gerais incluem o modelo 'cascata', desenvolvimento incremental e desenvolvimento orientado a reúso.

- A engenharia de requisitos é o processo de desenvolver uma especificação de software.

Pontos Importantes

- Processos de projeto e implementação se preocupam em transformar uma especificação de requisitos em um sistema de software executável.

- A validação de software é o processo de checar se o sistema está em conformidade com sua especificação e se esse está de acordo com as necessidades reais do usuário do sistema.

- A evolução de software ocorre quando você altera sistemas de software existentes para adequá-los a novas necessidades. O software precisa evoluir para continuar útil.

Lidando com mudanças

- As mudanças são inevitáveis em todos grandes projetos de software.
  - ✓ Mudanças no negócio levam a novos e diferentes requisitos de sistema.
  - ✓ Novas tecnologias abrem novas possibilidades para melhorar implementações.
  - ✓ Mudanças de plataforma requerem mudanças na aplicação.
- As mudanças geram retrabalho, o que faz com que o custo das mudanças inclua o retrabalho (p.ex. reanálise dos requisitos) assim como o custo de implementação de novas funções.

Reduzindo o custo de retrabalho

- Prevenção de mudanças, quando o processo de software inclui atividades que podem antecipar possíveis mudanças antes que o retrabalho se torne necessário.

  - ✓ Por exemplo, um protótipo de sistema pode ser desenvolvido para mostrar algumas características fundamentais do sistema para os clientes.

- Tolerância a mudanças, quando o processo é desenvolvido para que mudanças possam ser acomodadas a um custo relativamente baixo.

  - ✓ Geralmente envolve alguma forma de desenvolvimento incremental. As mudanças propostas podem ser implementadas em incrementos que ainda não foram desenvolvidos. Se isso é impossível, então um incremento único (uma pequena parte do sistema) pode ser alterada para incorporar a mudança.

Prototipação de software

- Um protótipo é uma versão inicial de um sistema usada para demonstrar conceitos e testar opções de projeto.

- Um protótipo pode ser usado:

  - ✓ No processo de engenharia de requisitos para ajudar na elicitação e validação de requisitos;

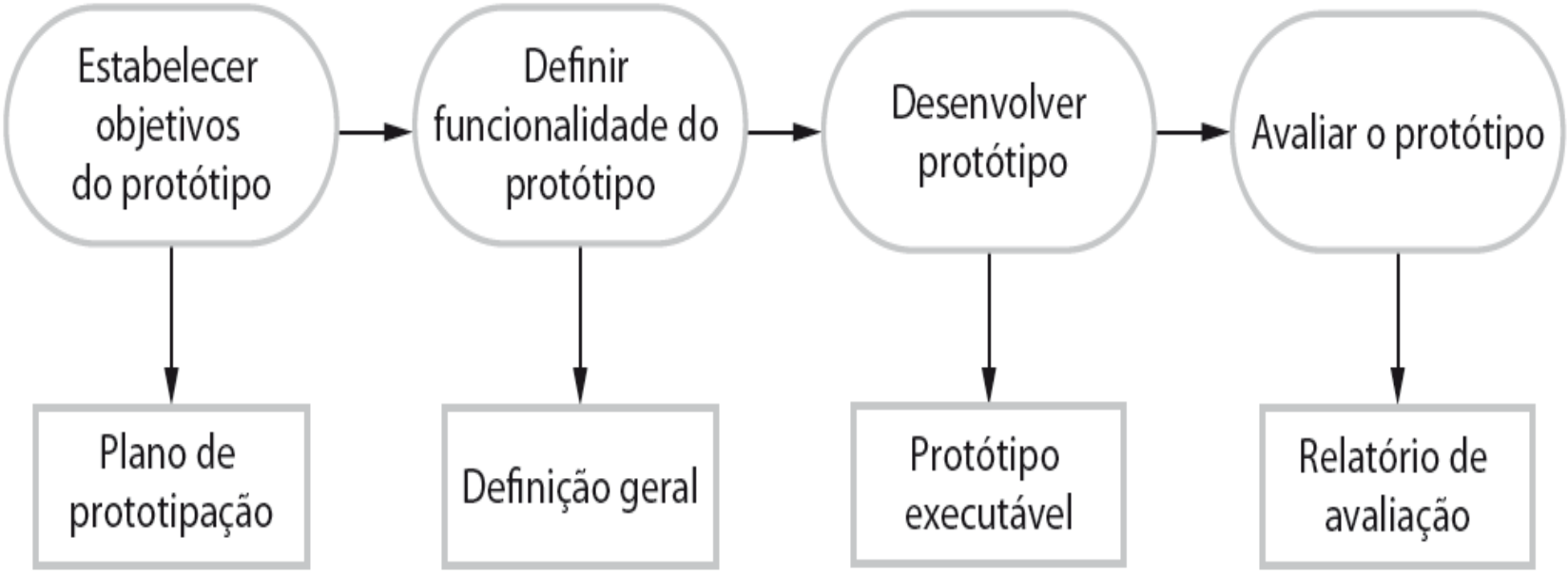
  - ✓ Nos processos de projeto para explorar opções e desenvolver um projeto de interface de usuário;

  - ✓ No processo de testes para executar testes fim-a-fim.

Benefícios da prototipação

- Melhoria do uso do software.
- Maior proximidade com as necessidades do usuário.
- Melhorias na qualidade do projeto.
- Maior manutenibilidade.
- Reduzir esforços de desenvolvimento.

O processo de desenvolvimento de protótipo



Desenvolvimento de protótipos

- Pode ser baseado em linguagens ou ferramentas de prototipagem rápida.
- Pode deixar a funcionalidade de fora do teste.
  - ✓ A prototipação deve focar em áreas do produto que não são bem entendidas;
  - ✓ A checagem de erros e recuperação podem não estar incluídas no protótipo;
  - ✓ O foco deve ser em requisitos funcionais ao invés de não funcionais como por exemplo, a confiabilidade e a segurança.

Descarte de protótipos

- Os protótipos devem ser descartados depois do desenvolvimento, pois não são uma boa base para um sistema em produção:

  - ✓ Pode ser impossível ajustar o sistema para alcançar requisitos não funcionais;
  - ✓ Geralmente os protótipos não possuem documentação;
  - ✓ Geralmente a estrutura do protótipo é degradada por mudanças rápidas;
  - ✓ Provavelmente o protótipo não irá alcançar os padrões normais de qualidade organizacional.

Entrega incremental

- Ao invés de entregar o sistema em uma única entrega, o desenvolvimento e a entrega são distribuídos em incrementos, nos quais cada incremento entrega parte da funcionalidade necessária.

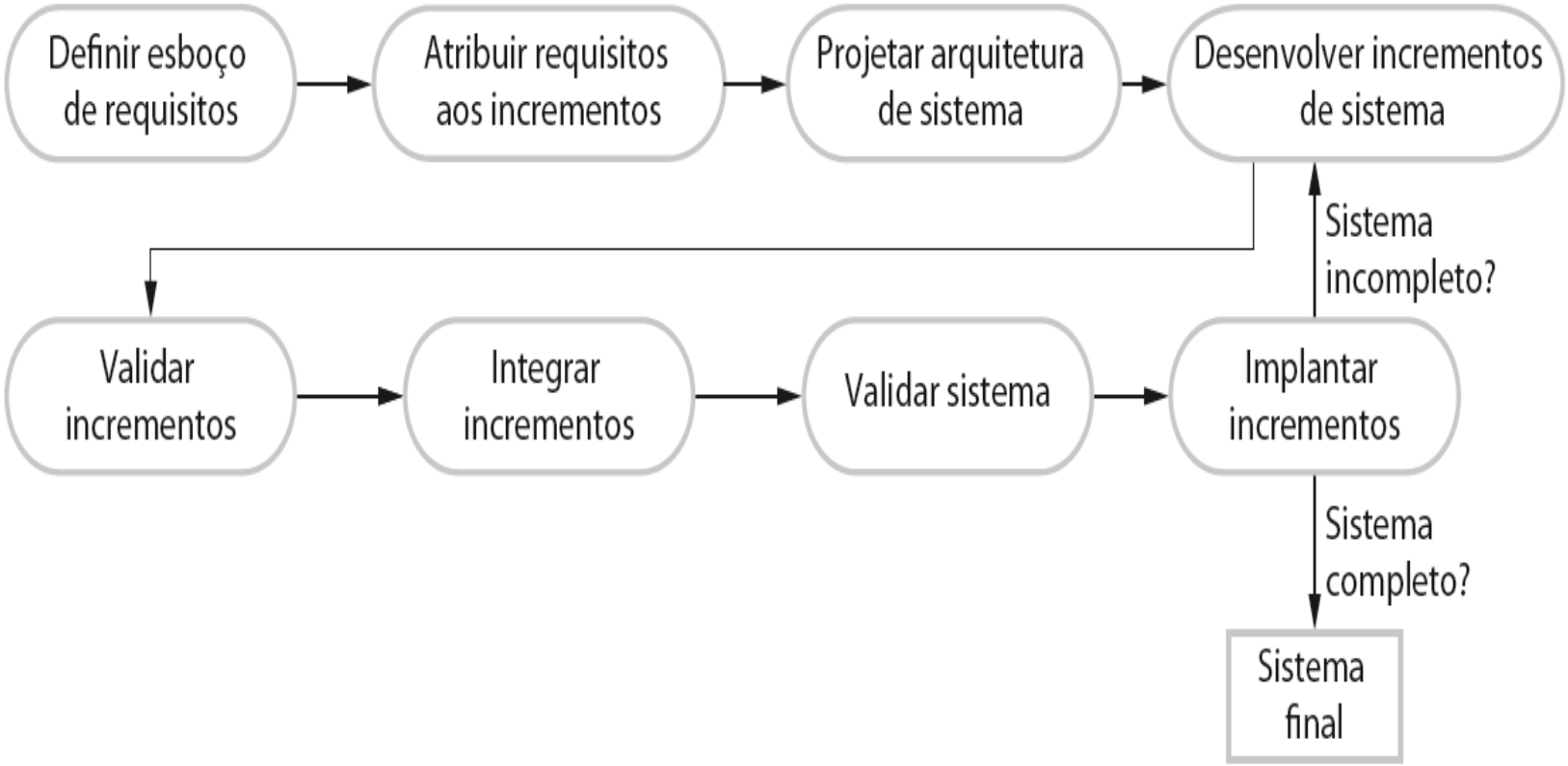
- Os requisitos do usuário são priorizados e os requisitos de mais alta prioridade são incluídos nos primeiros incrementos.

- Assim que o desenvolvimento de um incremento é iniciado os requisitos são congelados, mas os requisitos dos incrementos posteriores podem continuar a evoluir.

Desenvolvimento e entrega incremental

- Desenvolvimento incremental
  - ✓ Desenvolve o sistema em incrementos e avalia cada incremento antes de proceder com o desenvolvimento do próximo incremento;
  - ✓ Abordagem normalmente usada em métodos ágeis;
  - ✓ Avaliação feita por representantes do usuário/cliente.
- Entrega incremental
  - ✓ Implanta um incremento para uso do usuário-final;
  - ✓ Avaliação mais realística sobre o uso prático do software;
  - ✓ Difícil de implementar para sistemas substitutos devido aos incrementos possuírem menos funções do que o sistema que está sendo substituido.

Entrega incremental



Vantagens da entrega incremental

- Os valores podem ser entregues ao cliente junto com cada incremento, e funções do sistema ficam disponíveis mais rápidamente.
- Primeiros incrementos agem como protótipos para ajudar a deduzir requisitos para incrementos posteriores.
- Menor risco de falha geral do projeto.
- Os serviços mais prioritários do sistema tendem a serem mais testados.

Problemas da entrega incremental

- A maioria dos sistemas requer um conjunto de funções básicas que são usadas por diferentes partes do sistema.

  - ✓ Como os requisitos não são definidos em detalhes até que um incremento seja implementado, pode ser difícil identificar funções comuns que são necessárias a todos os incrementos.

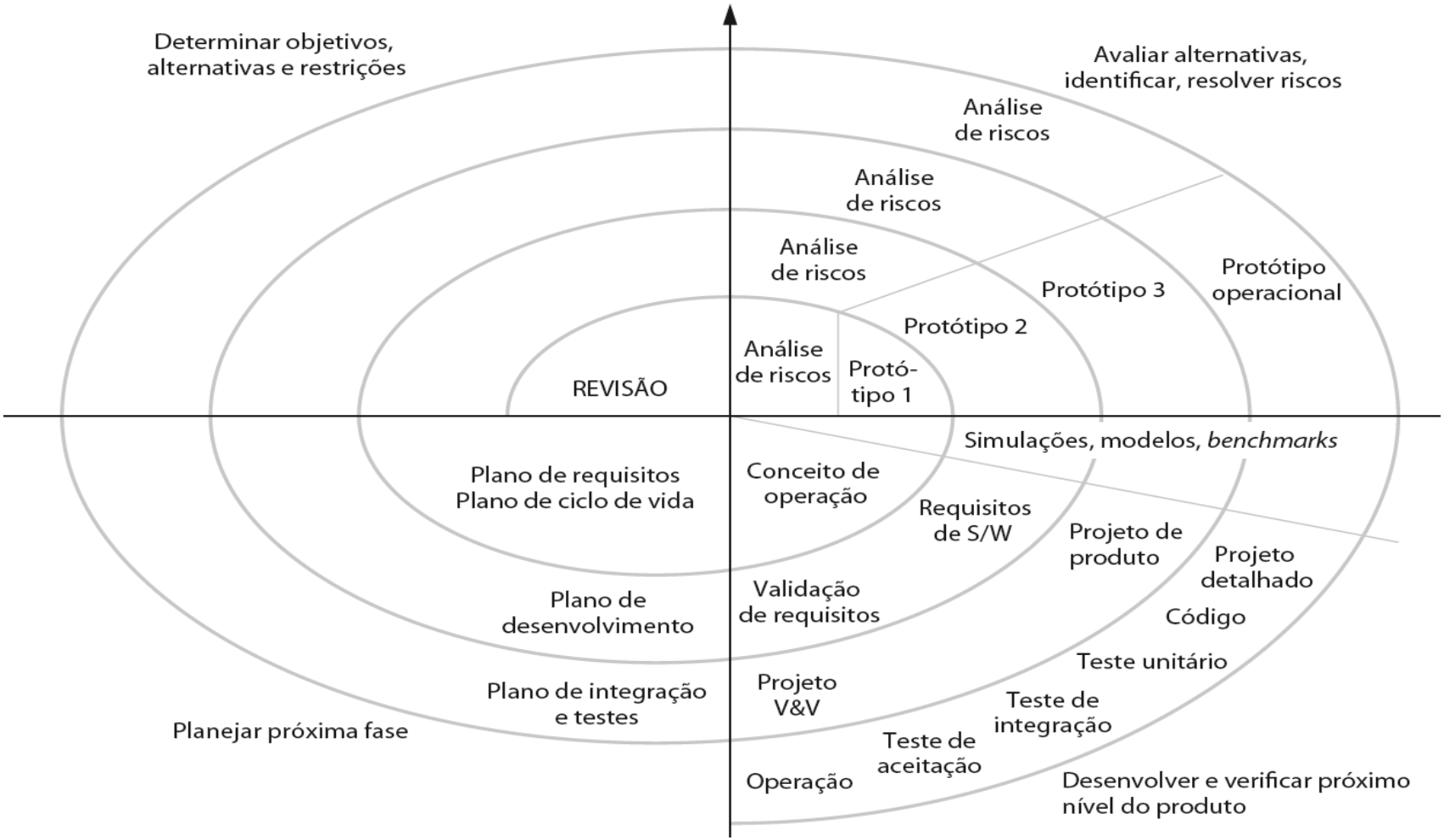
- A essência dos processos iterativos é que a especificação seja desenvolvida em conjunto com o software.

  - ✓ No entanto, essa pode entrar em conflito com o modelo de aquisição de muitas organizações, nos quais a especificação completa do sistema é parte do contrato de desenvolvimento do sistema.

Modelo espiral de Boehm

- O processo é representado como uma espiral ao invés de uma sequência de atividades com retornos.
- Cada loop na espiral representa uma fase do processo.
- Não existem fases fixas como especificação ou projeto – os loops na espiral são escolhidos de acordo com a necessidade.
- Os riscos são avaliados explicitamente e resolvidos no decorrer do processo.

O modelo de processo de software espiral de Boehm



Setores do modelo espiral

- Definição de objetivos
  - ✓ São identificados os objetivos específicos para cada fase.

- Avaliação e redução de riscos
  - ✓ Os riscos são avaliados e atividades executadas para reduzir os principais riscos.

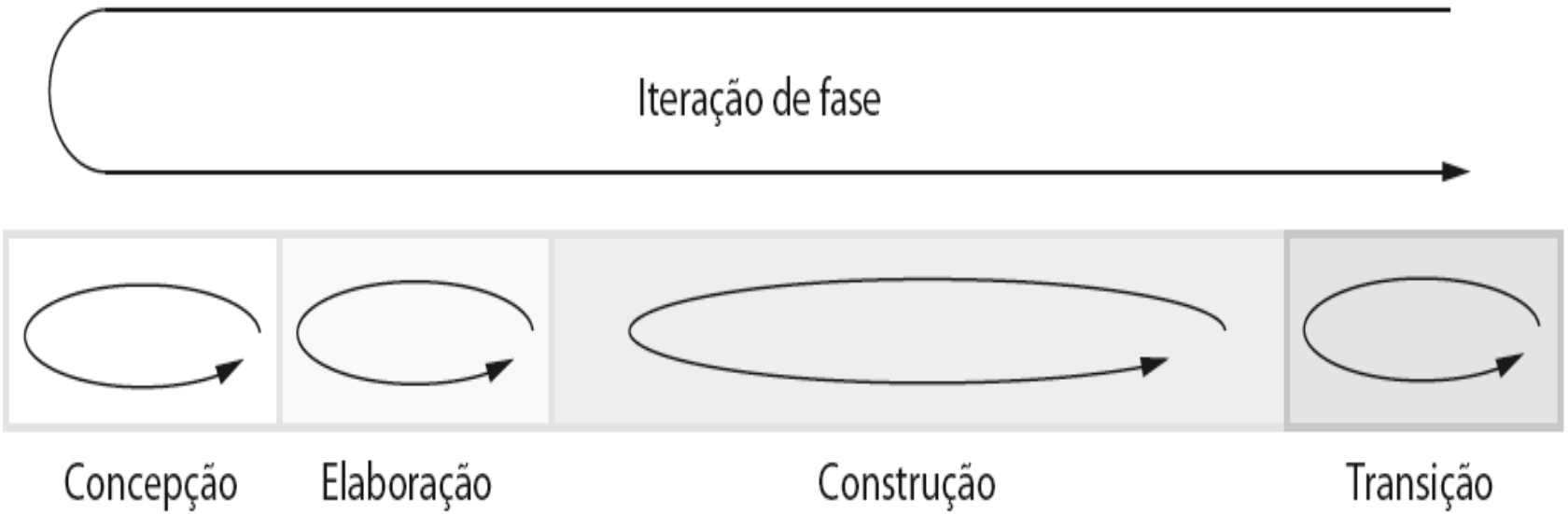
- Desenvolvimento e validação
  - ✓ Um modelo de desenvolvimento para o sistema é escolhido, pode ser qualquer um dos modelos genéricos.

- Planejamento
  - ✓ O projeto é revisto e a próxima fase da espiral é planejada.

Rational Unified Process (RUP)

- É um processo genérico moderno, derivado do trabalho em UML e processos associados.
- Reúne aspectos dos 3 modelos genéricos discutidos previamente.
- Geralmente descrito por 3 perspectivas:
  - ✓ Uma perspectiva dinâmica que mostra fases no tempo;
  - ✓ Uma perspectiva estática que mostra atividades do processo;
  - ✓ Uma perspectiva prática que sugere boas práticas.

Fases no Rational Unified Process



Fases do RUP

- Concepção
  - ✓ Estabelece o *business case* para o sistema.
- Elaboração
  - ✓ Desenvolve um entendimento da extensão do problema e da arquitetura do sistema.
- Construção
  - ✓ Projeta o sistema, programa e testa o sistema.
- Transição
  - ✓ Implanta o sistema no seu ambiente de operação.

Iteração do RUP

- Iteração Intra-fase
  - ✓ Cada fase é iterativa aos resultados desenvolvidos incrementalmente
- Iteração Inter-fase
  - ✓ Como mostrado pelo loop no modelo RUP, o conjunto todo de fases pode ser executado incrementalmente.

Workflows estáticos no RUP

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.

Workflows estáticos no RUP

WORKFLOW	DESCRIÇÃO
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.

Boas práticas do RUP

- Desenvolver software iterativamente
  - ✓ Planejar incrementos baseando-se nas prioridades do cliente e entregar as de prioridade mais alta primeiro.
- Gerenciar os requisitos
  - ✓ Documentar explicitamente os requisitos do cliente e manter registros de mudanças desses requisitos.
- Usar arquiteturas baseadas em componentes
  - ✓ Organizar a arquitetura do sistema como um conjunto de componentes reusáveis.

Boas práticas do RUP

- Modelar o software visualmente
  - ✓ Use modelos de gráficos UML para representar visões dinâmicas e estáticas do software.
- Verificar a qualidade do software
  - ✓ Garanta que o software atenda aos padrões de qualidade organizacional.
- Controlar as mudanças do software
  - ✓ Gerencie as mudanças no software usando um sistema de gerenciamento de mudanças e ferramentas de gerenciamento de configuração.

Pontos Importantes

- Os processos devem incluir atividades para lidar com mudanças. O que pode envolver uma fase de protipação que ajuda a evitar más escolhas nos requisitos e no projeto.

- Os processos devem ser estruturados para evolução e entrega iterativa, para que as mudanças possam ser feitas sem causar problemas ao sistema como um todo.

- O Rational Unified Process é um modelo de processo genérico moderno, organizado em fases (concepção, elaboração, construção e transição) mas que separa as atividades dessas fases (requisitos, análise e projeto, etc.) .