Processos de Software

O processo de software

- Um conjunto estruturado de atividades necessárias para o desenvolvimento de um sistema de software
  - Especificação;
  - Projeto;
  - Validação;
  - Evolução.
- Um modelo de processo de software é uma representação abstrata do processo. Ele apresenta a descrição de um processo a partir de uma perspectiva particular.

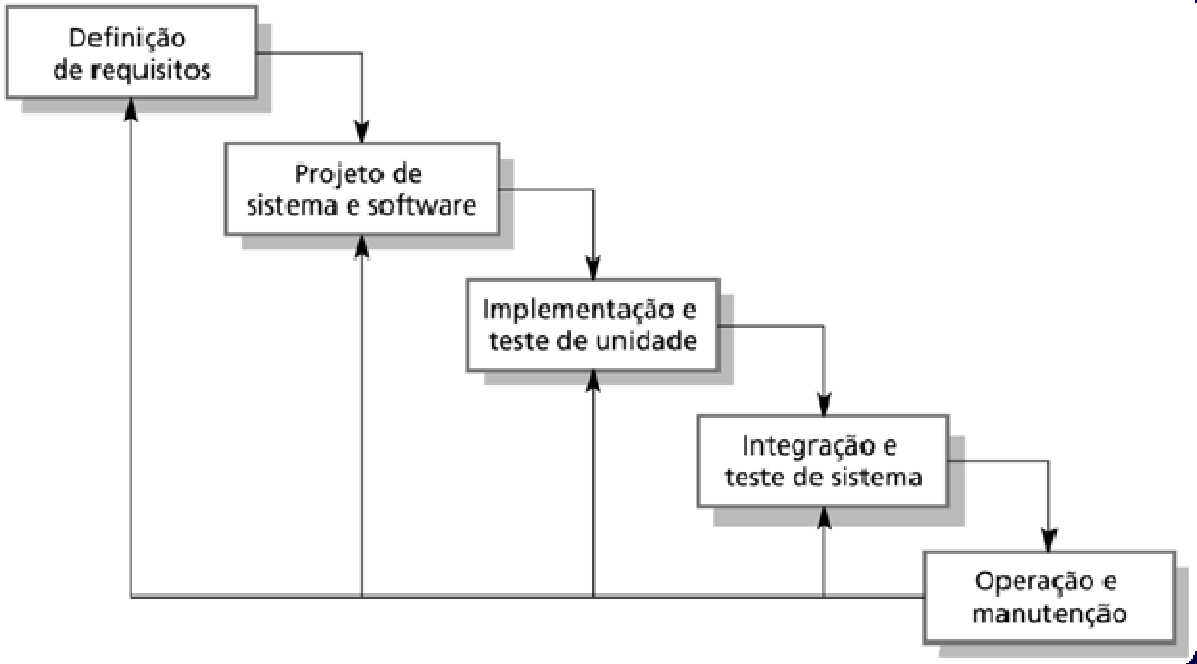
Modelos genéricos de processo de software

- O modelo cascata
  - Fases separadas e distintas de especificação e desenvolvimento.
- Desenvolvimento evolucionário
  - Especificação, desenvolvimento e validação são intercalados.
- Engenharia de software baseada em componentes
  - O sistema é montado a partir de componentes existentes.
- Existem muitas variantes destes modelos, por exemplo, o desenvolvimento formal onde um processo semelhante ao cascata é usado, mas a especificação é uma especificação formal, que é refinada durante os vários estágios para um projeto implementável.

Modelo cascata

Figura 4.1

Ciclo de vida de software.



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
- A principal desvantagem do modelo cascata é a dificuldade de acomodação das mudanças depois que o processo está em andamento. Uma fase tem de estar completa antes de passar para a próxima.

Problemas do modelo cascata

- Particionamento inflexível do projeto em estágios distintos, dificulta a resposta aos requisitos de mudança do cliente.
- Portanto, este modelo é apropriado somente quando os requisitos são bem compreendidos, e quando as mudanças forem bastante limitadas durante o desenvolvimento do sistema.
- Poucos sistemas de negócio têm requisitos estáveis.
- O modelo cascata é o mais usado em projetos de engenharia de sistemas de grande porte, onde um sistema é desenvolvido em várias localidades.

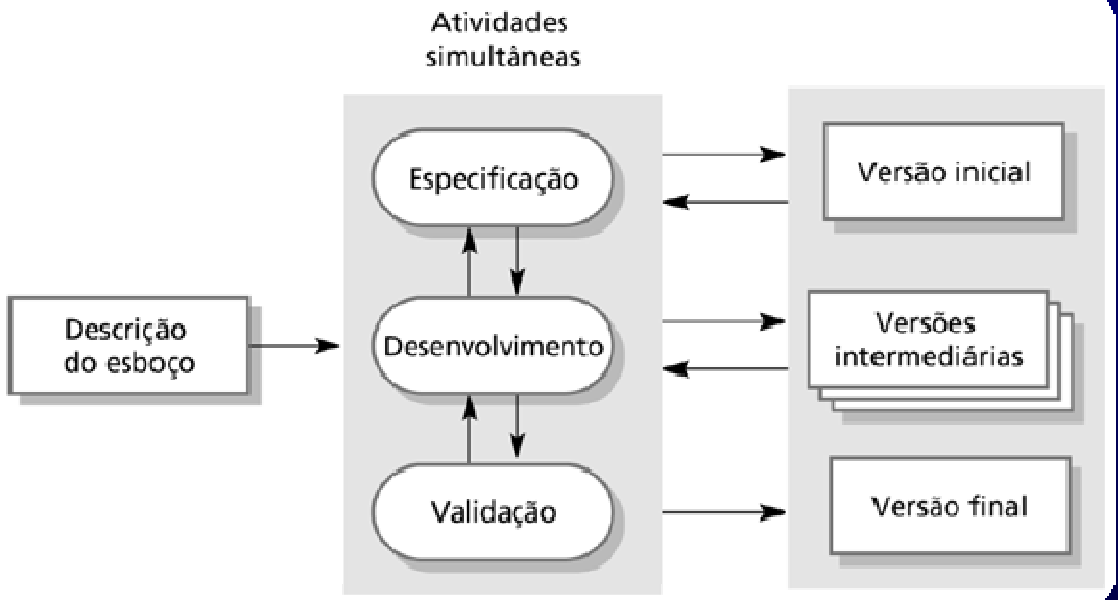
Desenvolvimento evolucionário

- Desenvolvimento exploratório
  - O objetivo é trabalhar com os clientes e desenvolver um sistema final a partir de uma especificação inicial. Deve iniciar com requisitos bem compreendidos e adicionar novas características à medida que forem propostas pelo cliente.
- Protótipação throwaway
  - O objetivo é compreender os requisitos de sistema. Deve iniciar com requisitos mal compreendidos para esclarecer o que é realmente necessário.

Desenvolvimento evolucionário

Figura 4.2

Desenvolvimento evolucionário.



Desenvolvimento evolucionário

- Problemas
  - Falta de visibilidade de processo;
  - Os sistemas são freqüentemente mal estruturados;
  - Habilidades especiais (por exemplo, em linguagens para prototipação rápida) podem ser solicitadas.
- Aplicabilidade
  - Para sistemas interativos de pequeno e médio portes;
  - Para partes de um sistema de grande porte (por exemplo, a interface de usuário);
  - Para sistema com curto ciclo de vida.

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.
- Duas abordagens (relacionadas)
  - Entrega incremental;
  - Desenvolvimento espiral.

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.

Desenvolvimento incremental

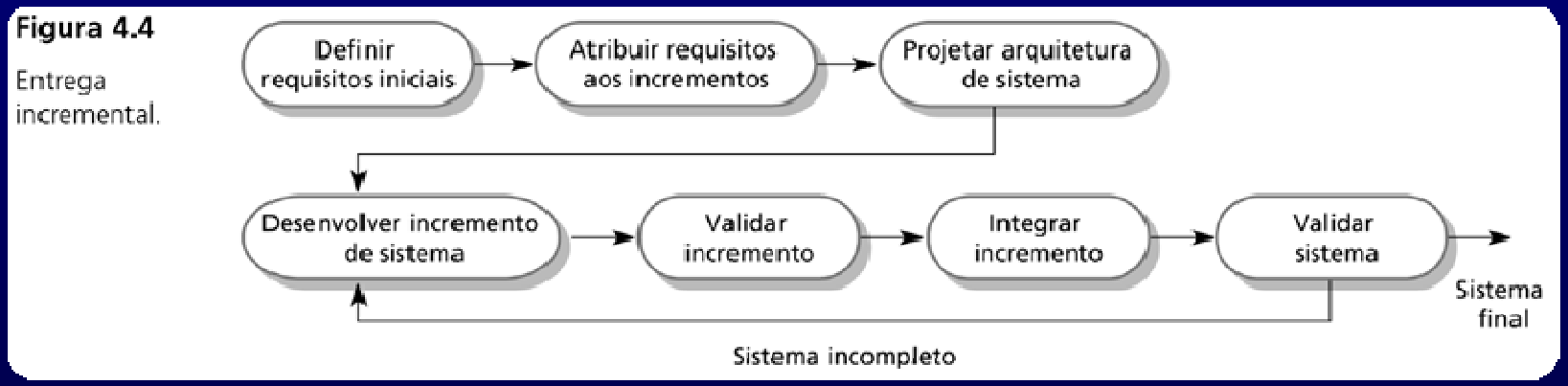


Figura 4.4 Entrega incremental.

Vantangens do desenvolvimento incremental

- O valor pode ser entregue para o cliente com cada incremento e, desse modo, a funcionalidade de sistema é disponibilizada mais cedo.
- 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 espiral

- O processo é representado como uma espiral ao invés de uma seqüência 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

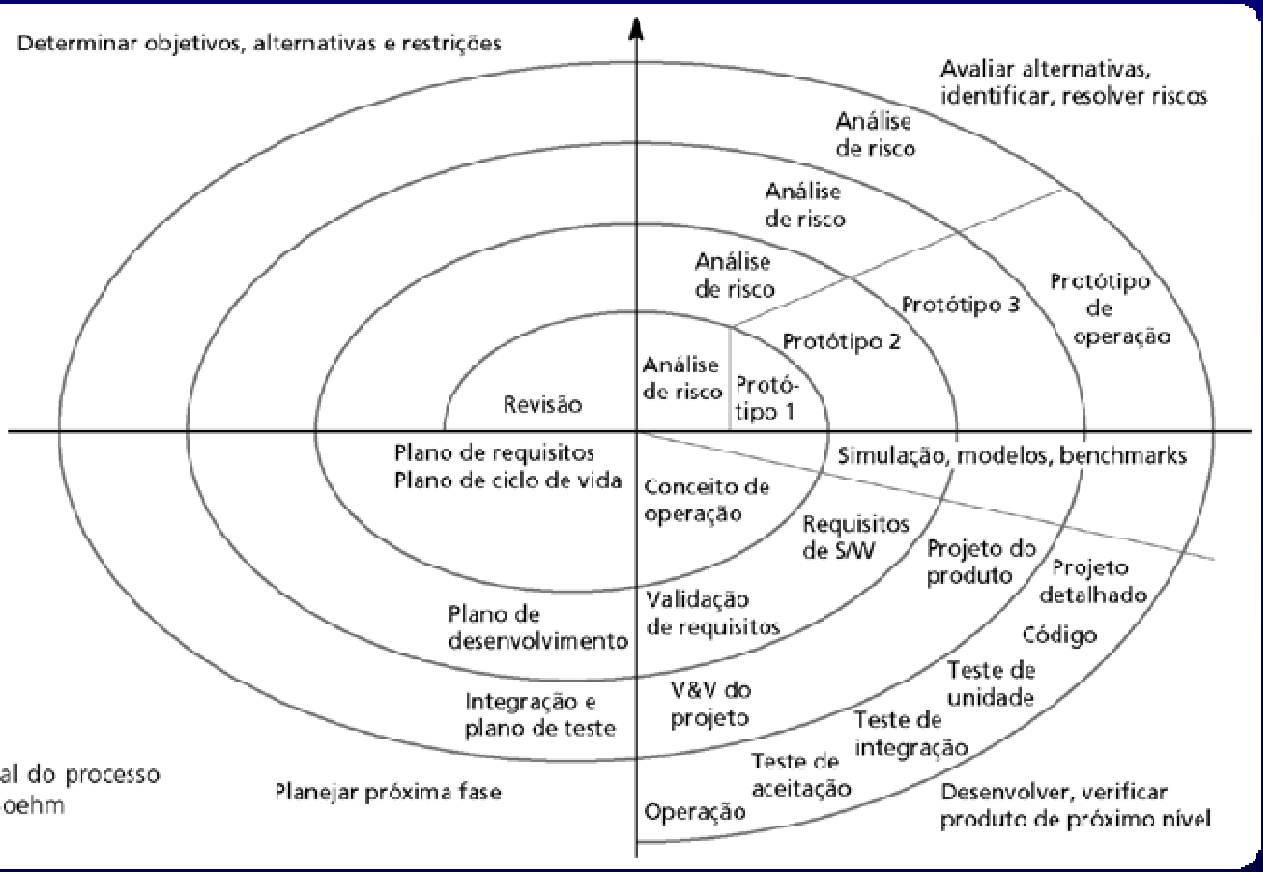


Figura 4.5

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

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.

Atividades de processo

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

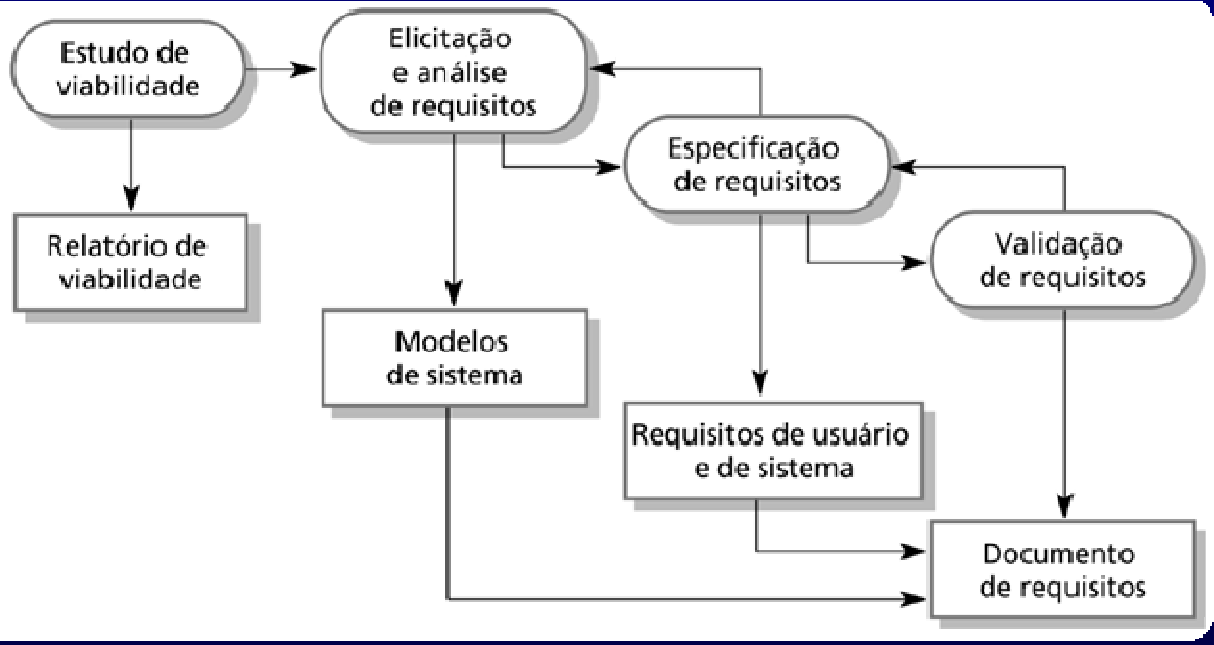
Especificação de software

- 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;
  - Elicitação e análise de requisitos;
  - Especificação de requisitos;
  - Validação de requisitos.

O processo de engenharia de requisitos

Figura 4.6

Processo de engenharia de requisitos.



Projeto e implementação de software

- É o processo de conversão da especificação de sistema em um sistema executável.
- 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 intercaladas.

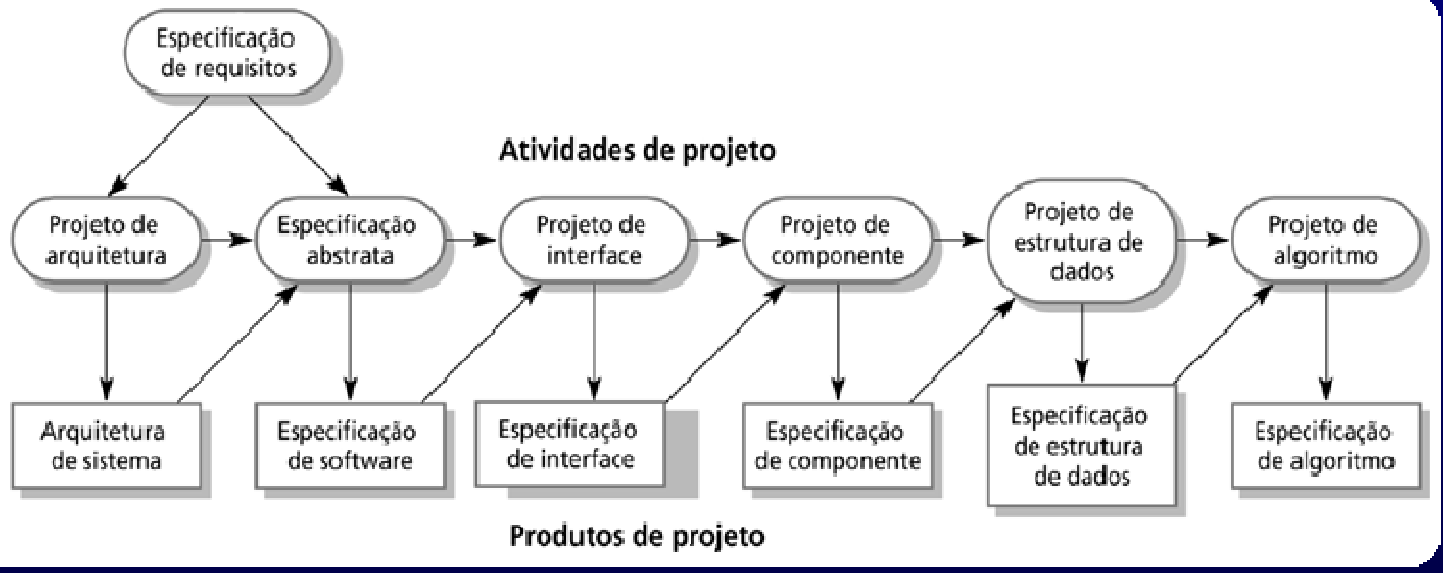
Atividades do processo de projeto

- Projeto de arquitetura
- Especificação abstrata
- Projeto de interface
- Projeto de componente
- Projeto de estrutura de dados
- Projeto de algoritmo

O processo de projeto de software

Figura 4.7

Modelo geral de processo de projeto.



Métodos estruturados

- Abordagens sistemáticas para o desenvolvimento de projetos de software.
- O projeto é, em geral, documentado como um conjunto de modelos gráficos.
- Modelos possíveis
  - Modelo de objeto;
  - Modelo de seqüência;
  - Modelo de transição de estado;
  - Modelo estruturado;
  - Modelo de fluxo de dados.

Programação e *debugging*

- 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.
- Programadores realizam alguns testes para descobrir defeitos no programa e removem esses defeitos no processo de *debugging*.

O processo de *debugging*

Figura 4.8
Processo de debugging.

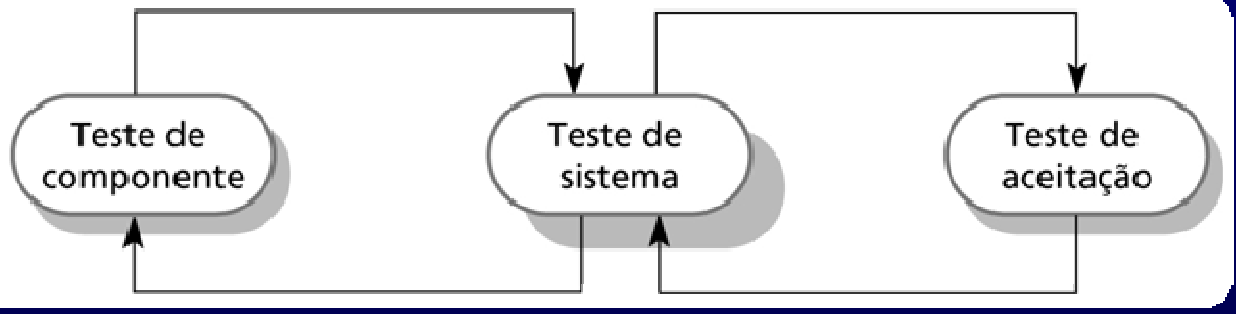
Validação de software

- Verificação e validação (V & V) tem a intenção de mostrar que um sistema está em conformidade com a sua especificação e que atende aos requisitos do cliente do.
- Envolve processos de verificação e revisão, além de testes de sistema.
- Esses testes envolvem a execução do sistema com casos de teste que são derivados da especificação de dados reais a serem processados por ele.

O processo de teste

Figura 4.9

Processo de teste.



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.
- Teste de aceitação
  - Teste com dados do cliente para verificar se o sistema atende às suas necessidades.

Fases de teste

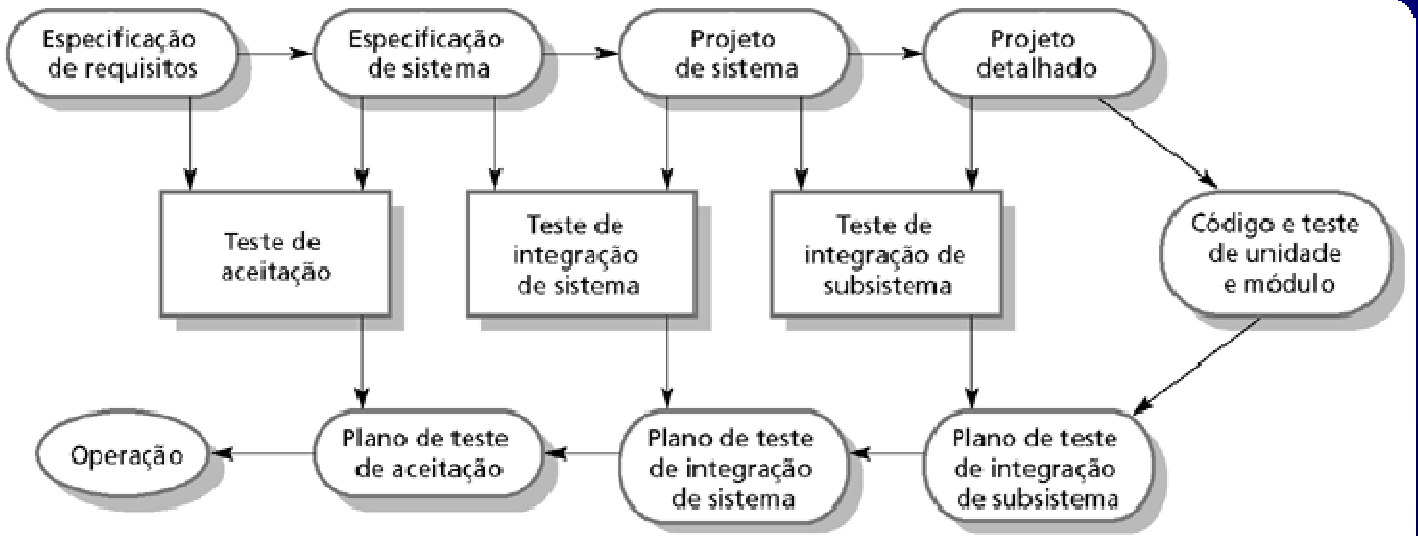


Figura 4.10 Fases de teste no processo de software.

Evolução de software

- O software é inerentemente flexível e pode mudar.
- Como os requisitos mudam durante as mudanças de circunstâncias de negócio, o software que apóia o negócio deve também evoluir e mudar.
- Embora tenha havido uma separação entre desenvolvimento e evolução (manutenção), isso é cada vez mais irrelevante à medida que cada vez menos sistemas são completamente novos.

Evolução de sistema

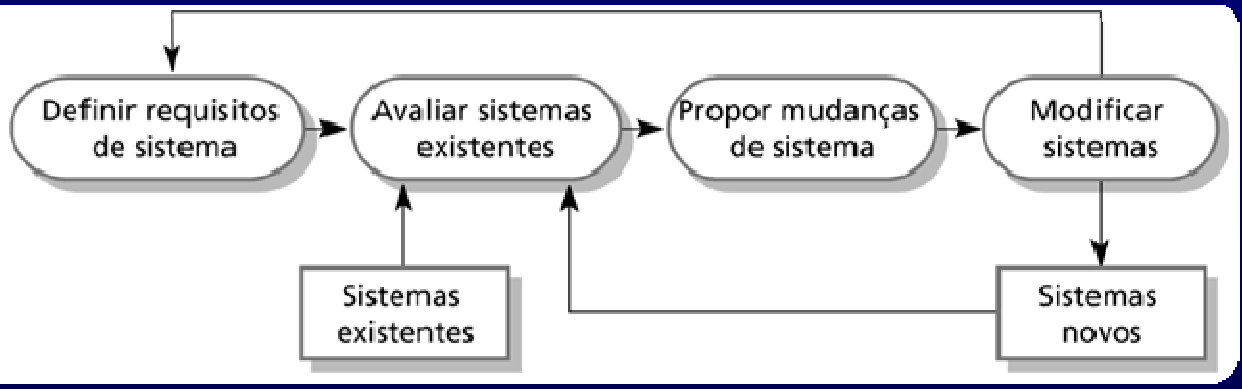


Figura 4.11

Evolução de sistema.

Engenharia de software auxiliada por computador

- A engenharia de software auxiliada por computador (CASE) é um software usado para apoiar as atividades de processo de desenvolvimento e evolução de software.
- Automação de atividades
  - Editores gráficos para o desenvolvimento de modelos de sistema;
  - Dicionário de dados para gerenciar entidades de projeto;
  - Construtores de UI (Interfaces de Usuário) gráficos para a construção de interfaces;
  - *Debuggers* para apoiar a descoberta de defeitos de programa;
  - Tradutores automáticos para gerar novas versões de um programa.

Tecnologia CASE

- Tecnologia CASE tem conduzido a melhorias significativas do processo de software. Embora, estas não sejam de ordem de magnitude de melhorias que foram uma vez previstas
  - Engenharia de software requer pensamento criativo – isto não é prontamente automatizado;
  - Engenharia de software é uma atividade de equipe e, para projetos de grande porte, muito tempo é dispendido nas interações de equipe. Tecnologia CASE não apóia realmente estas interações.

Classificação de CASE

- A classificação nos ajuda a compreender os diferentes tipos de ferramentas CASE e seu apoio às atividades de processo.
- Perspectiva funcional
  - As ferramentas são classificadas de acordo com a sua função específica.
- Perspectiva de processo
  - As ferramentas são classificadas de acordo com atividades de apoio que fornecem.
- Perspectiva de integração
  - Ferramentas são classificadas de acordo com sua organização em unidades integradas.

Classificação funcional de ferramentas

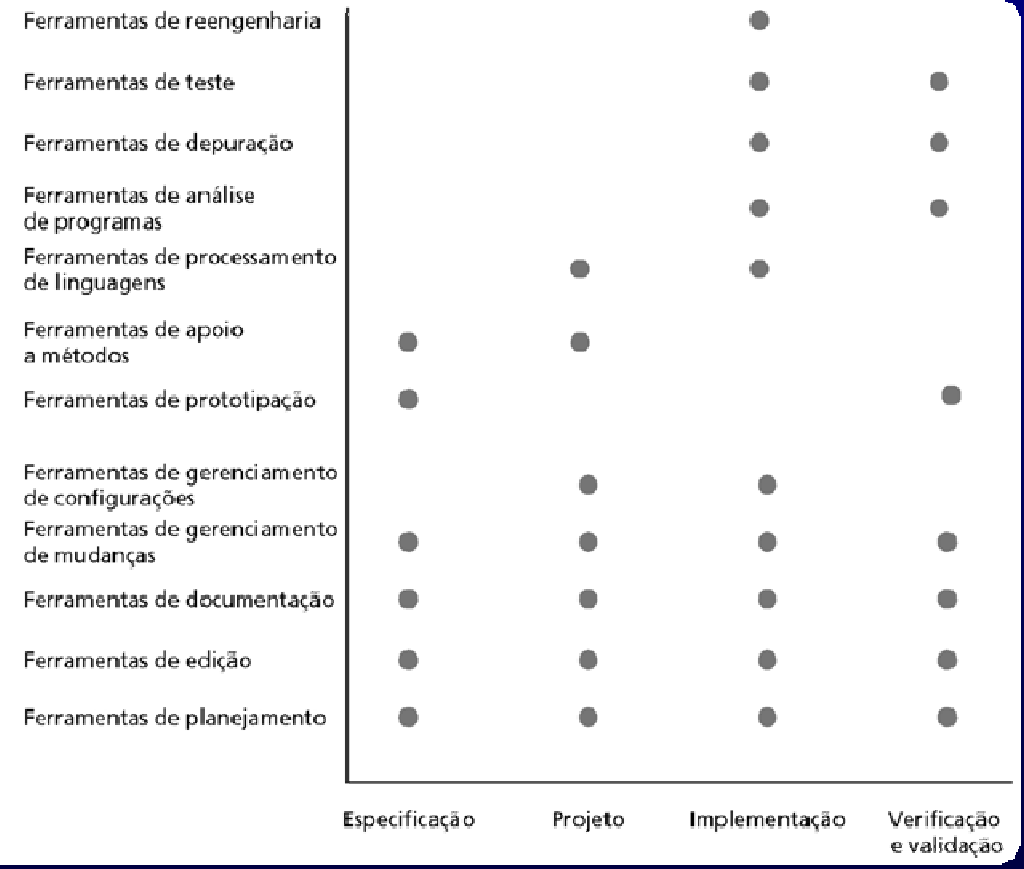
Tabela 4.2 Classificação funcional de ferramentas CASE.

Tipo de ferramenta	Exemplos
Ferramentas de planejamento	Ferramentas PERT, ferramentas para estimativas, planilhas
Ferramentas de edição	Editores de texto, editores de diagramas, processadores de texto
Ferramentas de gerenciamento de mudanças	Ferramentas de controle de requisitos, sistemas de controle de mudanças
Ferramentas de gerenciamento de configuração	Sistemas de gerenciamento de versões, ferramentas de construção de sistemas
Ferramentas de prototipação	Linguagens de nível muito alto, geradores de interface com o usuário
Ferramentas de apoio a métodos	Editores de projeto, dicionários de dados, geradores de código
Ferramentas de processamento de linguagens	Compiladores, interpretadores
Ferramentas de análise de programa	Geradores de referências cruzadas, analisadores estáticos, analisadores dinâmicos
Ferramentas de teste	Geradores de dados de teste, comparadores de arquivos
Ferramentas de depuração	Sistemas de depuração interativos
Ferramentas de documentação	Programas de formatação de páginas, editores de imagens
Ferramentas de reengenharia	Sistemas de referência cruzada, sistemas de reestruturação de programas

Classificação de ferramentas baseada em atividades

Figura 4.13

Classificação de ferramentas CASE baseada em atividades.



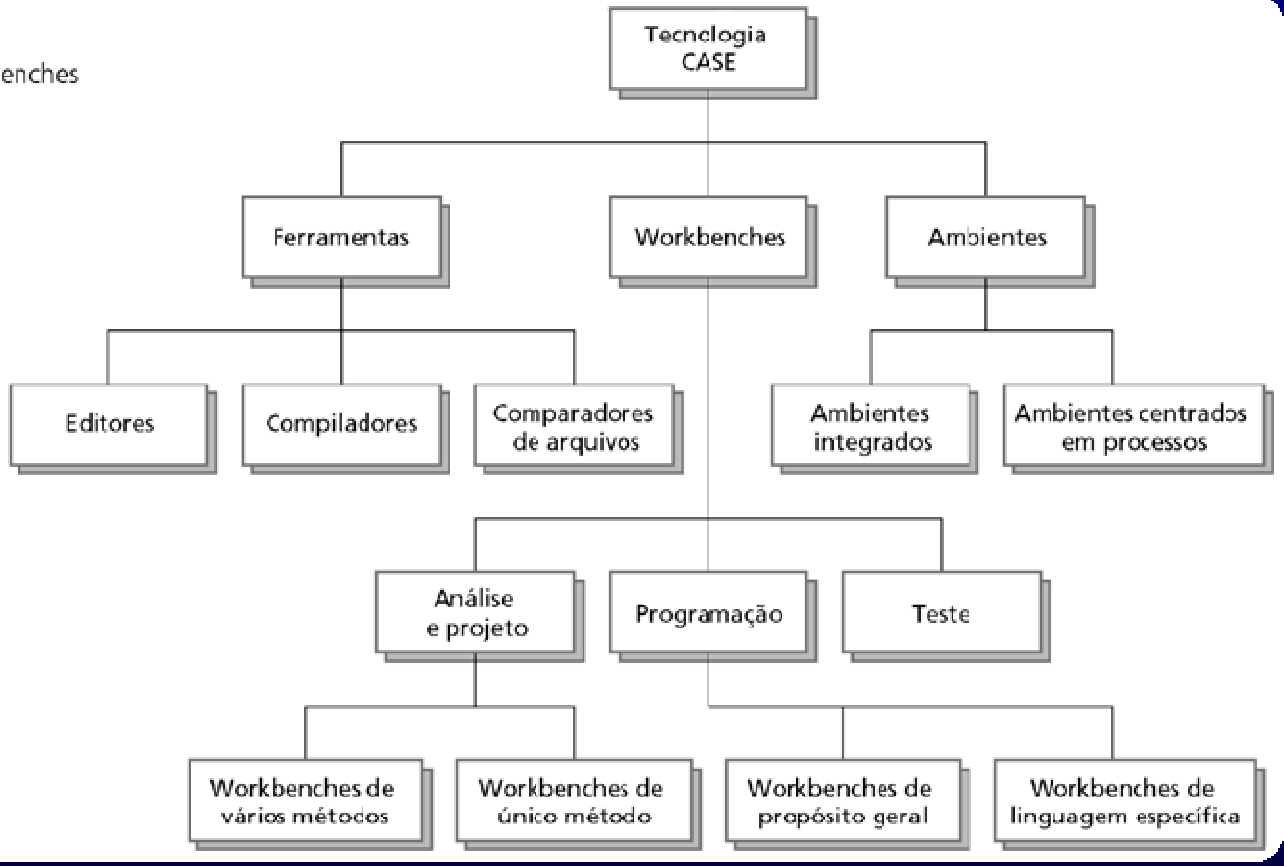
Integração de CASE

- Ferramentas
  - Apóiam tarefas individuais de processo, tais como verificação de consistência de projeto, edição de texto, etc.
- *Workbenches*
  - Apóiam as fases do processo, tais como especificação e projeto. Normalmente, incluem uma série de ferramentas integradas.
- Ambientes
  - Apoiar todo ou uma parte substancial do processo inteiro de software. Normalmente, incluem vários *workbenches* integrados.

Ferramentas, *workbenches*, ambientes

Figura 4.14

Ferramentas, workbenches e ambientes.



Pontos-chave

- Processos de software são atividades envolvidas na produção e na evolução de um sistema de software.
- Modelos de processo de software são representações abstratas destes processos.
- Atividades gerais incluem especificação, projeto e implementação, validação e evolução de software.
- Modelos genéricos de processo descrevem a organização dos processos de software. Exemplos incluem o modelo cascata, o desenvolvimento evolucionário e engenharia de software baseada em componentes.
- Modelos de processo iterativos apresentam o processo de software como um ciclo de atividades.

Pontos-chave

- Engenharia de requisitos é o processo de desenvolvimento de uma especificação do software.
- Processos de projeto e implementação transformam a especificação em um programa executável.
- A validação envolve a verificação para saber se o sistema atende à sua especificação e às necessidades do usuário.
- Evolução está relacionada à modificação do sistema depois que ele estiver em uso.
- O Rational Unified Process é um modelo genérico de proceso que separa atividades a partir das fases.
- A tecnologia CASE apóia as atividades de processo de software.