

Processos de Software

- Conjunto coerente de atividades para especificar, projetar, implementar e testar sistemas de software

Objetivos

- Apresentar os modelos de processo de software
- Descrever os diferentes modelos de processos e quando eles podem ser utilizados
- Descrever em formas gerais os modelos de processo para engenharia de requisitos, desenvolvimento de software, testes e evolução
- Apresentar a tecnologia CASE para apoiar atividades do processo de software

Tópicos abordados

- Modelos de processo de software
- Iteração do processo
- Especificação de software
- Projeto e implementação de software
- Validação de software
- Evolução de software
- Apoio automatizado ao processo

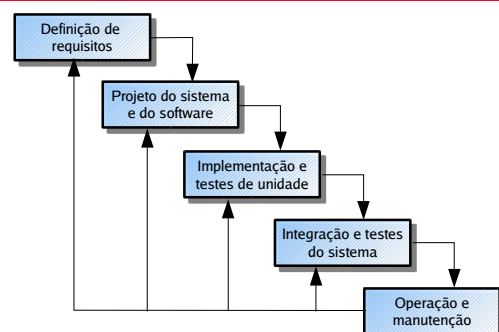
O processo de software

- Um conjunto estruturado de atividades exigidas para desenvolver um sistema de software
 - Especificação
 - Projeto
 - Validação
 - Evolução
- Um modelo de processo de software é uma representação abstrata de um processo. Ele apresenta uma descrição de um processo a partir de uma perspectiva específica

Modelos genéricos de modelos de processo de software

- O modelo cascata
 - Fases de especificação e desenvolvimento separadas e distintas
- Desenvolvimento evolucionário
 - Especificação e desenvolvimento são interfoliadas
- Desenvolvimento formal de sistemas
 - Um modelo matemático do sistema é transformado formalmente em uma implementação
- Desenvolvimento baseado em reuso
 - O sistema é montado a partir de componentes existentes

Modelo cascata



Fases do modelo cascata

- Definição e análise de requisitos
- Projeto do sistema e do software
- Implementação e testes de unidade
- Integração e testes do sistema
- Operação e manutenção
- A desvantagem do modelo cascata é a dificuldade de acomodar as mudanças após o processo ter sido iniciado

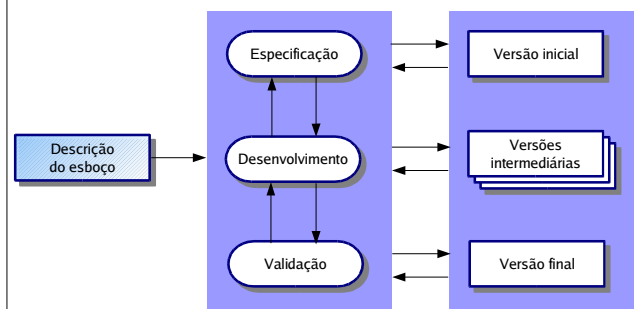
Problemas do modelo cascata

- Particionamento inflexível do projeto em fases distintas
- Isso torna difícil responder a requisitos do usuário que mudam
- Portanto, esse modelo é apropriado somente quando os requisitos são bem compreendidos

Desenvolvimento evolucionário

- Desenvolvimento exploratório
 - O objetivo é trabalhar com os clientes e evoluir um sistema final a partir de uma especificação genérica inicial. O desenvolvimento se inicia com as partes do sistema que estão compreendidas
- Fazer protótipos descartáveis
 - O objetivo é compreender os requisitos do sistema. O protótipo se concentra em fazer experimentos com partes dos requisitos que estejam mal compreendidas

Desenvolvimento evolucionário



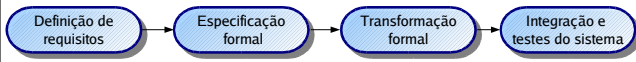
Desenvolvimento evolucionário

- Problemas
 - Falta de visibilidade do processo
 - Os sistemas freqüentemente possuem pouca estrutura
 - Podem ser exigidas habilidades especiais (p.ex. em linguagens para desenvolvimento rápido)
- Aplicabilidade
 - Para sistemas interativos pequenos ou de médio porte
 - Para partes de sistemas grandes (p.ex., a interface com o usuário)
 - Para sistemas de vida curta

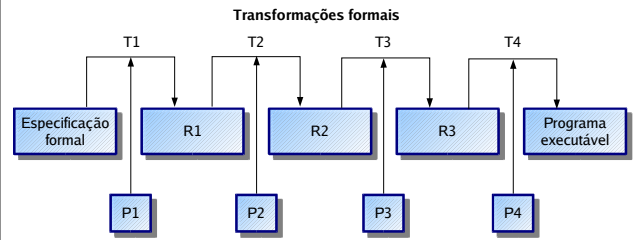
Desenvolvimento formal de sistemas

- Baseia-se na transformação de uma especificação matemática por meio de diferentes representações para um programa executável
- As transformações 'preservam a corretude', de tal forma que possa ser diretamente mostrado que o programa está de acordo com a sua especificação
- Embutida na abordagem de desenvolvimento de software chamada 'Cleanroom'

Desenvolvimento formal de sistemas



Transformações formais



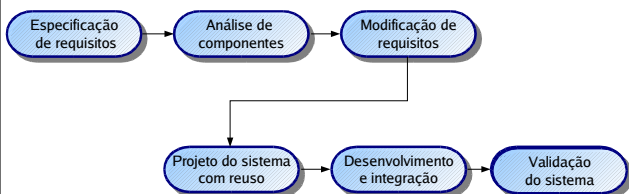
Desenvolvimento formal de sistemas

- **Problemas**
 - Necessidade de habilidades especiais e treinamento para aplicar a técnica
 - Dificuldade de especificar formalmente alguns aspectos do sistema, tais como a interface com o usuário
- **Aplicabilidade**
 - Sistemas críticos, especialmente aqueles onde um estudo de segurança deve ser feito antes de pôr o sistema em operação

Desenvolvimento orientado ao reuso

- Baseia-se no reuso sistemático, onde sistemas são integrados a partir de componentes existentes ou sistemas COTS (Commercial-off-the-shelf)
- **Estágios do processo**
 - Análise dos componentes
 - Modificação de requisitos
 - Projeto do sistema com reuso
 - Desenvolvimento e integração
- Esta abordagem está se tornando mais importante, mas ainda há uma experiência limitada com ela

Desenvolvimento orientado ao reuso



Iteração de processo

- Os requisitos do sistema SEMPRE evoluem ao longo de um projeto, portanto a iteração do processo, onde estágios iniciais são retrabalhados, é sempre parte do processo para sistemas grandes
- A iteração pode ser aplicada a qualquer um dos modelos genéricos de processo
- **Duas abordagens (relacionadas)**
 - Desenvolvimento incremental
 - Desenvolvimento espiral

Desenvolvimento incremental

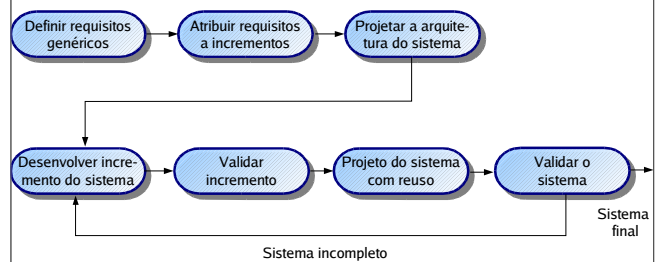
- Ao invés de entregar o sistema como uma única entrega, quebram-se o desenvolvimento e a entrega em incrementos, com cada incremento entregando parte da funcionalidade requerida
- Os requisitos do usuário são priorizados e os requisitos de prioridade mais alta são incluídos nos incrementos iniciais
- Uma vez que o desenvolvimento de um incremento é iniciado, os requisitos são congelados, ainda que os requisitos para incrementos posteriores continuem a evoluir

©Ian Sommerville 2000

Software Engineering, 6th edition, Chapter 3

Slide 19

Desenvolvimento incremental



©Ian Sommerville 2000

Software Engineering, 6th edition, Chapter 3

Slide 20

Vantagens do desenvolvimento incremental

- Cada incremento pode entregar valor para o cliente, portanto a funcionalidade do sistema está disponível mais cedo
- Incrementos iniciais atuam como um protótipo para ajudar a descobrir requisitos para os incrementos posteriores
- Menor risco de falha do projeto como um todo
- Os serviços de mais alta prioridade do sistema tendem a receber a maior parte dos testes

©Ian Sommerville 2000

Software Engineering, 6th edition, Chapter 3

Slide 21

Programação extrema (Extreme programming)

- Nova abordagem de desenvolvimento baseada no desenvolvimento e entrega de incrementos de funcionalidade muito pequenos
- Baseia-se em melhoria constante do código, envolvimento do usuário na equipe de desenvolvimento e programação em pares

©Ian Sommerville 2000

Software Engineering, 6th edition, Chapter 3

Slide 22

Desenvolvimento espiral

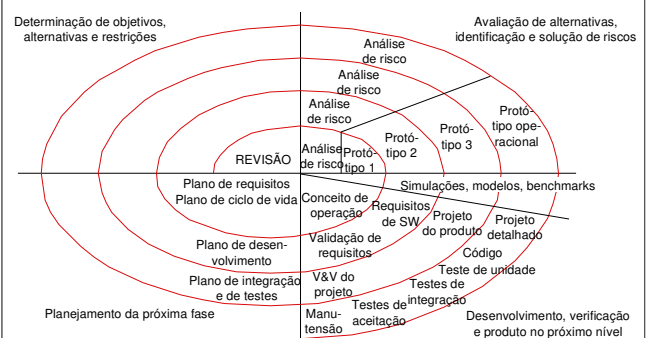
- O processo é representado como uma espiral, em vez de uma seqüência de atividades com caminhos de retorno
- Cada volta na espiral representa uma fase no processo
- Não há fases fixas, tais como especificação ou projeto
 - As voltas na espiral são escolhidas dependendo do que for exigido
- Os riscos são explicitamente avaliados e resolvidos durante todo o processo

©Ian Sommerville 2000

Software Engineering, 6th edition, Chapter 3

Slide 23

Modelo espiral do processo de software



©Ian Sommerville 2000

Software Engineering, 6th edition, Chapter 3

Slide 24

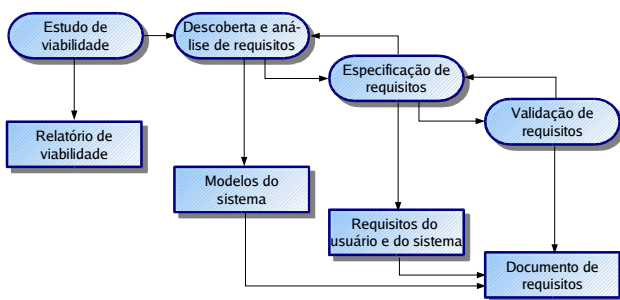
Setores do modelo espiral

- Definição do objetivo
 - Identificam-se os objetivos específicos da fase
- Avaliação e redução de risco
 - Os riscos são avaliados e são adotadas as atividades para reduzir os riscos principais
- Desenvolvimento e avaliação
 - É escolhido um modelo de desenvolvimento para o sistema, que pode ser qualquer um dos modelos genéricos
- Planejamento
 - O projeto é revisado e a próxima fase da espiral é planejada

Especificação de software

- O processo de estabelecer quais serviços são exigidos e as restrições na operação e no desenvolvimento do sistema
- Processo de engenharia de requisitos
 - Estudo de viabilidade
 - Descoberta e análise de requisitos
 - Especificação de requisitos
 - Validação de requisitos

O processo de engenharia de requisitos



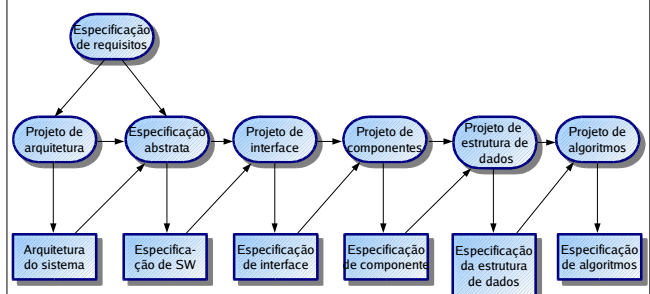
Projeto e implementação de software

- O processo de converter a especificação do sistema em um sistema executável
- Projeto de software
 - Projetar uma estrutura de software que realize a especificação
- Implementação
 - Traduzir essa estrutura em um programa executável
- As atividades de projeto e implementação são proximamente relacionadas e podem ser interfoliadas

Atividades do processo de projeto

- Projeto de arquitetura
- Especificação abstrata
- Projeto de interface
- Projeto de componentes
- Projeto de estrutura de dados
- Projeto de algoritmos

O processo de projeto de software



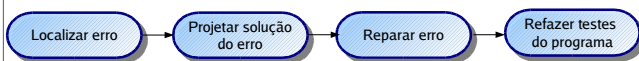
Métodos de projeto

- Abordagens sistemáticas para desenvolver um projeto de software
- O projeto normalmente documentado como um conjunto de modelos gráficos
- Modelos possíveis
 - Modelo de fluxo de dados
 - Modelo entidade-relacionamento-atributo
 - Modelo estrutural
 - Modelos de objeto

Programação e depuração (*debugging*)

- Traduzir um projeto em um programa e remover erros do programa
- Programação é uma atividade pessoal – não há um processo genérico de programação
- Os programadores desenvolvem alguns testes do programa para descobrir falhas em um programa e remover essas falhas em um processo de depuração

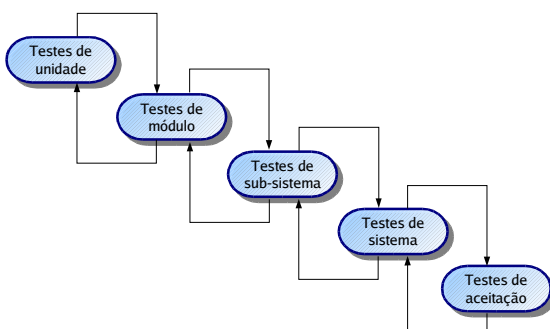
O processo de debugging



Validação de software

- Verificação e validação destinam-se a mostrar que um sistema conforma com a sua especificação e satisfaz os requisitos do cliente
- Envolve processos de verificação e revisão e testes do sistema
- O processo de testes do sistema envolve executar o sistema com casos de teste que são derivados da especificação dos dados reais a serem processados pelo sistema

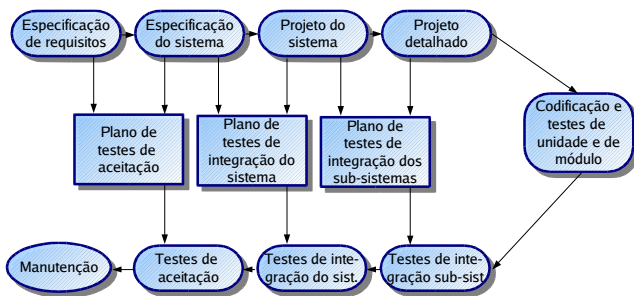
O processo de testes



Estágios do processo de testes

- Testes de unidade
 - Componentes individuais são testados
- Testes de módulo
 - Coleções relacionadas de componentes dependentes são testadas
- Testes de sub-sistemas
 - Módulos são integrados em sub-sistemas e testados. O foco aqui deve ser no teste das interfaces
- Testes de sistema
 - Testar o sistema como um todo. Teste de propriedades emergentes
- Testes de aceitação

Fases do processo de testes



©Ian Sommerville 2000

Software Engineering, 6th edition, Chapter 3

Slide 37

Evolução de software

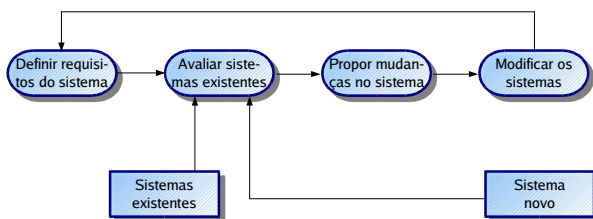
- O software é inerentemente flexível e pode mudar
- À medida em que os requisitos mudam por causa de mudanças nas circunstâncias de negócios, o software que apóia os negócios também deve evoluir e mudar
- Embora tenha existido uma demarcação entre desenvolvimento e evolução (manutenção), isso é cada vez mais irrelevante, uma vez que um número decrescente de sistemas são completamente novos

©Ian Sommerville 2000

Software Engineering, 6th edition, Chapter 3

Slide 38

Evolução do sistema



©Ian Sommerville 2000

Software Engineering, 6th edition, Chapter 3

Slide 39

Apoio automatizado ao processo (CASE)

- Computer-aided software engineering (CASE) é um software para apoiar o desenvolvimento de software e o processo de evolução
- Automação de atividades
 - Editores gráficos para desenvolvimento de modelos do sistema
 - Dicionário de dados para gerenciar entidades do projeto
 - Construtor de IU gráfico para construir a interface com o usuário
 - Debuggers para apoiar a busca por falhas no programa
 - Tradutores automatizados para gerar novas versões de um programa

©Ian Sommerville 2000

Software Engineering, 6th edition, Chapter 3

Slide 40

Tecnologia CASE

- A tecnologia CASE tem levado a melhorias significativas no processo de software, embora não na ordem de magnitude das melhorias que haviam sido previstas
 - Engenharia de software exige pensamento criativo – isso não é prontamente automatizável
 - Engenharia de software é uma atividade de equipe e, para projetos grandes, muito tempo é gasto em interações da equipe. A tecnologia CASE não suporta prontamente isso

©Ian Sommerville 2000

Software Engineering, 6th edition, Chapter 3

Slide 41

Classificação CASE

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

©Ian Sommerville 2000

Software Engineering, 6th edition, Chapter 3

Slide 42

Classificação funcional das ferramentas

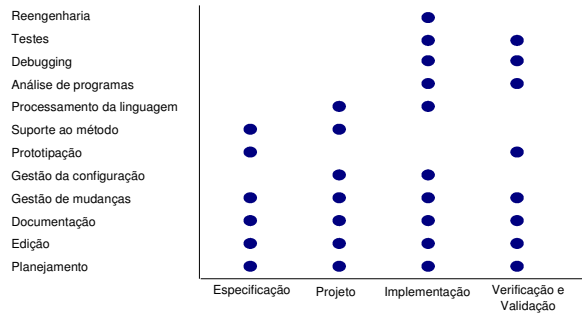
Tipo da ferramenta	Exemplos
Planejamento	Ferramentas PERT, de estimativas, planilhas de cálculo
Edição	Editores de texto, editores de diagramas, processadores de texto
Gestão de mudanças	Acompanhamento de requisitos, sistemas de controle de mudanças
Gestão da configuração	Sistemas de controle de versões, ferramentas de construção de sistemas
Prototipação	Linguagens de mais alto nível, geradores de interface com o usuário
Apoio ao método	Editores de projeto, dicionários de dados, geradores de código
Processamento de linguagem	Compiladores, interpretadores
Análise de programas	Geradores de ref. cruzadas, analisadores estáticos, analisadores dinâmicos
Ferramentas de testes	Geradores de dados de teste, comparadores de arquivos
Debugging	Sistemas de debugging interativos
Documentação	Programas de layout de páginas, editores de imagens
Reengenharia	Sistemas de referências cruzadas, sistemas de reestruturação de programas

©Ian Sommerville 2000

Software Engineering, 6th edition, Chapter 3

Slide 43

Classificação baseada em atividades



©Ian Sommerville 2000

Software Engineering, 6th edition, Chapter 3

Slide 44

Integração CASE

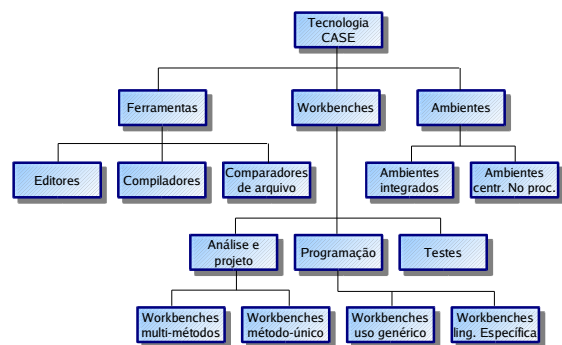
- **Ferramentas**
 - Apoiam tarefas do processo individuais, tais como verificação de consistência do projeto, edição de texto, etc.
- **Workbenches**
 - Apoiam uma fase do projeto, tais como especificação ou projeto. Normalmente, incluem um certo número de ferramentas integradas
- **Ambientes**
 - Apoiam todo ou uma parte substancial de um processo de software inteiro. Normalmente, incluem vários workbenches integrados

©Ian Sommerville 2000

Software Engineering, 6th edition, Chapter 3

Slide 45

Ferramentas, workbenches, ambientes



©Ian Sommerville 2000

Software Engineering, 6th edition, Chapter 3

Slide 46

Pontos principais

- Processos de software são as atividades envolvidas em produzir e evoluir um sistema de software. Eles são representados em um modelo de processo de software
- Atividades gerais são a especificação, o projeto e a implementação, a validação e a evolução
- Modelos de processo genéricos descrevem a organização de processos de software
- Modelos de processo iterativos descrevem o processo de software como um ciclo de atividades

©Ian Sommerville 2000

Software Engineering, 6th edition, Chapter 3

Slide 47

Pontos principais

- Engenharia de requisitos é o processo de desenvolver uma especificação de software
- Os processos de projeto e implementação transformam a especificação em um programa executável
- Validação envolve verificar que o sistema satisfaz a sua especificação e as necessidades do usuário
- A evolução preocupa-se em modificar o sistema após ele estar em uso
- A tecnologia CASE apóia as atividades do processo de software

©Ian Sommerville 2000

Software Engineering, 6th edition, Chapter 3

Slide 48