Processos de Software

Objetivos

- Apresentar modelos de processos de software
- Descrever três modelos genéricos de processo e quando eles podem ser usados
- Descrever, em linhas gerais, modelos de processo para engenharia de requisitos de software, o desenvolvimento de software, a realização de testes e evolução de software
- Apresentar dois exemplos de processos de software:
  - Processo Unificado (RUP)
  - Programação Extrema (XP)
- Apresentar a tecnologia CASE, usada para apoiar as atividades de processo de software

O processo de software

- Um conjunto estruturado de atividades, procedimentos, artefatos e ferramentas necessários para o desenvolvimento de um sistema de software
  - Especificação;
  - Projeto;
  - Validação;
  - Evolução.
- Exemplos: Processo Unificado (RUP), Programação Extrema, UML Components
- Diferente da definição do livro!!!
- 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.
- 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 formal e refinada durante os vários estágios para um projeto implementável.

Modelo cascata

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

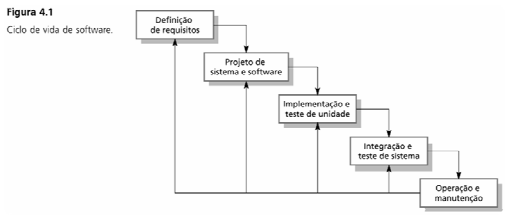


Figura 4.1
Ciclo de vida de software.

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

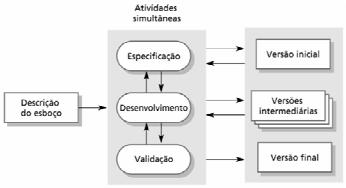
- Particionamento inflexível do projeto em estágios distintos, dificulta a resposta aos requisitos de mudança do cliente.
- Documentos "completamente elaborados" são necessários para fazer as transições entre estágios
- Apropriado somente quando os requisitos são bem compreendidos, e quando as mudanças forem raras
- 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

- Inicia-se a partir de um sistema pré-existente e, através de incrementos, constrói-se um novo sistema
- Duas modalidades:
  - **Desenvolvimento exploratório**: o objetivo é trabalhar com os clientes e desenvolver um sistema final a partir de uma especificação inicial.
  - **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 necessárias.
- Aplicabilidade
  - Para sistemas interativos de pequeno e médio portes;
  - Para partes de um sistema de grande porte (por exemplo, a interface de usuário);
  - Para sistemas com curto ciclo de vida.

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

Desenvolvimento orientado a reuso

Figura 4.3
Engenharia de software baseado em componentes.

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

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.

Vantangens do desenvolvimento incremental

- Incrementos podem ser entregues regularmente ao cliente 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.

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.
- Processo de desenvolvimento que usaremos ao longo da disciplina

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

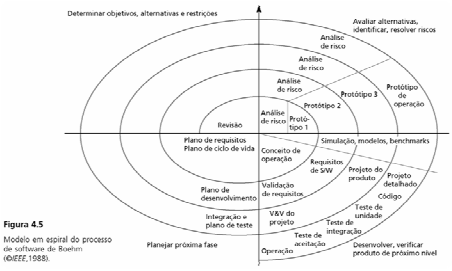


Figura 4.5
Modelo em espiral do processo de software de Boehm (©IEEE,1988).

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

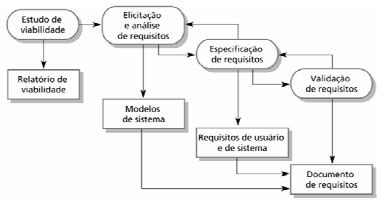
- Especificação de software
- Projeto e implementação de software
- Validação de software
- Evolução de software

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.



- Também pode envolver a prototipação de partes do sistema!

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 intecaladas.

Atividades do processo de projeto

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

Atividades do processo de projeto

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

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 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, verificação de modelos
- **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 serem 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.
- Teste de aceitação
  - Teste com dados do cliente para verificar se o sistema atende às suas necessidades.

Fases de teste

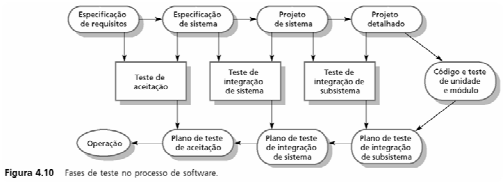


Figura 4.10 Fases de teste no processo de software.

Evolução de software

- O software é inerentemente flexível e pode mudar
- Requisitos mudam devido a diversos fatores e o software devem acompanhar essas mudanças
- Embora tenha havido uma separação entre desenvolvimento e evolução, isso é cada vez mais irrelevante à medida que cada vez menos sistemas são completamente novos
  - Processos e métodos iterativos (XP, RUP, Espiral) normalmente não fazem uma sepação explícita
- Evolução pode se dever a diversas razões:
  - Correções (*patches*)
  - Adição de novas funcionalidades
  - Mudanças de requisitos

Evolução de sistema

Figura 4.11
Evolução de sistema.

O Rational Unified Process

- É um modelo de processo moderno derivado da UML e do Processo Unificado de Desenvolvimento de Software
- Fortemente focado na documentação do sistema
  - Baseado na UML
- 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 boas práticas.

Modelo de fases do RUP

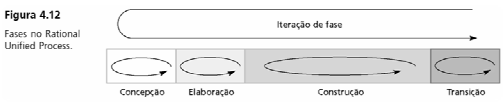


Figura 4.12
Fases no Rational Unified Process.

Fases do RUP

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

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

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 de 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.

Sobre os projetos...

- Quais são as equipes?
- 3 ou 4 membros
- Idealmente, cada equipe deve ter um número **par** de membros (=> 4 é melhor que 3)
- Isso é válido tanto para os alunos de **graduação** quanto para os de **mestrado**

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.

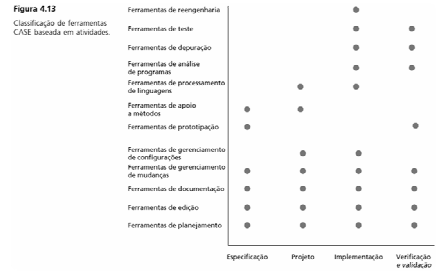
Classificação funcional de ferramentas

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.

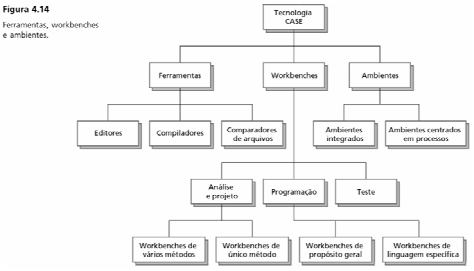


Integração de CASE

- 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.



Referência Adicional

- Barry W. Boehm. A Spiral Model of Software Development and Enhancement. **IEEE Computer**, vol. 21, número 5, Maio de 1988.