

Introdução à Eng. Software

Definição:

Um software novo pode ser criado por meio do **desenvolvimento** de novos programas, da configuração de sistemas de softwares genéricos ou da **reutilização** de um software existente. Engenheiros de software devem adotar uma postura sistemática e organizada para seus trabalhos e utilizar processos, técnicas e ferramentas apropriadas, dependendo do problema a ser resolvido, das restrições no desenvolvimento e dos recursos disponíveis.

As atividades genéricas em todos os processos de software são:

- **Especificação** – o que o sistema deve fazer e as restrições de desenvolvimento;
Requisitos **Não funcionais** = restrições do sistema
Requisitos **Funcional** = os serviços que o sistema deve oferecer
- **Desenvolvimento** – **produção** do sistema de software;
- **Validação** – avaliar se o software **é o que o cliente deseja**;
- **Evolução** – mudar o software em resposta às necessidades de mudanças.
Cerca de **60%** dos custos são de **desenvolvimento**, **40%** são custos de **teste**.

• O software deve fornecer as funcionalidades e o desempenho requeridos para o usuário e deve ser fácil de manter, ser confiável e utilizável

- **Manutenibilidade**

– O software deve **evoluir para atender às necessidades de mudança**.

- **Confiabilidade**

– O software deve ser **confiável e seguro**.

- **Eficiência**

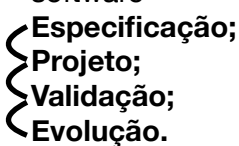
– O software **não deve fazer uso desnecessário de recursos** do sistema.

- **Usabilidade**

– O software deve ser **aceito pelos usuários para o qual foi projetado**. Isto significa que ele deve ser **compreensível, usável e compatível** com outros sistemas.

Processos de Software:

Um conjunto estruturado de atividades necessárias para o desenvolvimento de um sistema de software

- 
- Especificação;
 - Projeto;
 - Validação;
 - Evolução.

Modelo Cascata:

- **Fases separadas e distintas** de especificação e desenvolvimento

Análise e definição de requisitos

Projeto de sistema e software

Implementação e teste de unidade

Integração e teste de sistema

Operação e manutenção

Problemas: 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**. Portanto, este modelo é **apropriado somente quando os requisitos são bem compreendidos**, e quando as mudanças forem bastante limitadas durante o desenvolvimento do sistema.

Desenvolvimento Evolucionário:

- **Especificação, desenvolvimento e validação são intercalados.**

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.

Problemas: Falta de visibilidade de processo, os sistemas são freqüentemente mal estruturados e habilidades especiais 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) e para sistema com curto ciclo de vida.

Desenvolvimento Baseado 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:

- **Especificação** de Requisitos
- **Análise** de Componentes
- **Modificação** de Requisitos
- **Projeto de sistema** com reuso
- **Desenvolvimento** e integração

Iteração de Processo:

A iteração pode ser aplicada a qualquer um dos modelos genéricos do processo.

Duas abordagens (relacionadas):

- **Desenvolvimento 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.

Vantagens:

- 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 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.

Setores:

- **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 Genérico de Desenvolvimento:

- **Especificação de Software:**

O processo para **definir quais serviços são necessários e identificar as restrições** de operação e de desenvolvimento do sistema.

- **Projeto e Implementação de Software:**

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

- **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.** Esses testes envolvem a execução do sistema com casos de teste que são derivados da especificação de dados reais a serem processados por ele.

- **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.**

- **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.

Gerenciamento de Configuração:

Está relacionado com as políticas, processos e ferramentas para o gerenciamento de mudanças dos sistemas de software.

- **Item de Configuração:** **qualquer item** (código, documento, dados, etc.) **que estará sob o controle de configuração.**

- **Versão:** instância de um item de configuração que difere de outras instâncias deste item.

- **Baseline:** coleção de **versões de componentes** que compõe um sistema, **estas versões não podem ser alteradas.**

- **Release:** uma **versão** de um sistema **que foi liberada para uso pelos clientes.**

- **Workspace:** **área de trabalho** privado.

- **Build:** criação de uma versão de sistema executável pela **compilação e ligação de versões** adequadas de componentes e bibliotecas.

- **Branch:** criação de uma **versão de um item de configuração** que pode ser **trabalhada/modificada em paralelo à versão original.**

- **Merge:** criação de uma **versão** de um **item de configuração** pela **junção de versões** que estavam sendo trabalhadas em paralelo.