Engenharia de Software

Prof. Wellington Roque

Aula 1 - Introdução à Engenharia de Software

Objetivos da Aula

- Compreender os princípios básicos da Engenharia de Software.
- Identificar as atividades do ciclo de vida do desenvolvimento de software.
- Conhecer os diferentes modelos de processos de desenvolvimento de software.

1 - Definição de Engenharia de Software

Engenharia de Software refere-se ao processo sistemático e disciplinado de projetar, desenvolver, testar e manter software de forma eficaz e eficiente.

A disciplina abrange todo o ciclo de vida do software, desde a concepção da ideia até a sua desativação.

2 - Objetivos da Engenharia de Software

Garantir a qualidade do software

Entregar o software dentro do prazo e do orçamento

Adaptação contínua às mudanças nos requisitos

3 - Atividades do Ciclo de Vida

As atividades do ciclo de vida no desenvolvimento de software referem-se às etapas que um projeto atravessa desde sua concepção até sua desativação.

Cada atividade tem objetivos específicos e contribui para a evolução do sistema, como **análise, design, implementação, teste e manutenção.**

3.1 - Análise do sistema

- Nesta fase, os requisitos do sistema são levantados e analisados.
- Isso inclui a compreensão das **necessidades dos usuários (requisitos)**, **restrições**, **recursos necessários (hardware/software/orçameto)** e qualquer outra informação relevante.
- O resultado é a especificação dos "**requisitos do sistema**".

Entendimento dos requisitos do sistema

- **Requisitos** são descrições detalhadas das funcionalidades e características que um sistema deve possuir para atender às necessidades dos usuários e das partes interessadas.

- Eles são essenciais para o desenvolvimento, implementação e manutenção de um sistema.

3.1.1 – Características do sistema

- A identificação de **características** e **restrições** é uma parte crucial do processo de análise e design de sistemas.
- Essa etapa envolve a identificação das características ou requisitos que o sistema deve atender, bem como das restrições que afetam o design e a implementação do sistema.

Requisitos Funcionais

- Identificar as funções e operações que o sistema deve realizar. Por exemplo, em um sistema de gerenciamento de biblioteca, uma característica funcional pode ser "Registrar novo livro" ou "Pesquisar livros disponíveis".

Requisitos Não Funcionais

- Os requisitos não funcionais são aspectos do sistema que não estão diretamente relacionados às funcionalidades específicas do software,
- Exemplos:
  - Desempenho
  - Usabilidade
  - Segurança
  - Escalabilidade
  - Confiabilidade
  - Manutenibilidade
  - Portabilidade

Exemplos de requisitos de sistema

1 - Requisitos de Hardware

- Processador mínimo: 2.0 GHz
- Memória RAM: 4 GB
- Espaço em disco: 20 GB
- Placa gráfica compatível com DirectX 11

2 - Requisitos de Sistema Operacional

- Windows 10 ou superior
- macOS 10.14 ou superior
- Linux Ubuntu 18.04 LTS ou superior

Exemplos de requisitos de sistema

3 - Requisitos de Software

- Navegador web compatível (por exemplo, Google Chrome, Mozilla Firefox)
- Java Runtime Environment (JRE) versão 8 ou superior

4 - Requisitos de Compatibilidade

- Compatibilidade com dispositivos móveis (iOS, Android)
- Suporte a navegadores web populares (Chrome, Safari, Firefox)

Exemplos de requisitos de sistema

5 - Requisitos de Segurança

- Autenticação de dois fatores obrigatória
- Criptografia SSL/TLS para comunicação segura

6 - Requisitos de Desempenho

- Tempo de resposta do sistema inferior a 2 segundos
- Capacidade de processar 100 transações por minuto

Exemplos de requisitos de sistema

7 - Requisitos de Interface do Usuário

- Interface intuitiva e amigável
- Suporte a diferentes resoluções de tela

8 - Requisitos Funcionais

- Cadastro de usuários com perfis diferentes (administrador, usuário comum)
- Geração de relatórios mensais de atividades
- Integração com API de terceiros

Exemplos de requisitos de sistema

9 - Requisitos Não Funcionais

- Disponibilidade do sistema de 99,9%
- Manutenibilidade do código fonte
- Documentação completa do sistema

10 - Requisitos de Backup e Recuperação

- Backup diário dos dados do sistema
- Procedimentos de recuperação em caso de falha do sistema

Restrições do Sistema

- São limitações impostas ao processo de desenvolvimento ou ao próprio sistema.
- Podem incluir restrições de **tempo**, **orçamento**, **hardware** ou **software** específico.

3.2 Design

- A fase de design desempenha um papel crucial na Engenharia de Software, pois é aqui que a estrutura e a lógica do sistema são definidas.
- Vamos explorar os dois principais aspectos desta fase: **a criação da arquitetura do sistema** e a **especificação de como o software atenderá aos requisitos**.

Criação de uma arquitetura para o sistema

- A arquitetura de software refere-se à estrutura global do sistema, incluindo componentes, módulos, relações e a organização geral.

Especificação de como o software atenderá aos requisitos.

- Esta parte do design foca na especificação detalhada de como o software irá atender aos requisitos identificados na fase de análise.

Nível Físico ou de Implementação

Móveis Angular

Avenida Vale do Silício, 751, Bairro Estudante
Uberlândia-MG

NOTA FISCAL

Pedido n°:	1001
Data	25/06/2017

Detalhes do pedido:

Produto	Descrição	Preço unitário	Quantidade	Desconto	Subtotal
8021	Cadeira simples	400.00	4	0%	1600.00
8055	Mesa retangular	1500.00	1	10%	1350.00
8014	Estante	2000.00	1	5%	1900.00

Total do pedido:	4850.00

Preso ao paradigma: **orientado a objetos**

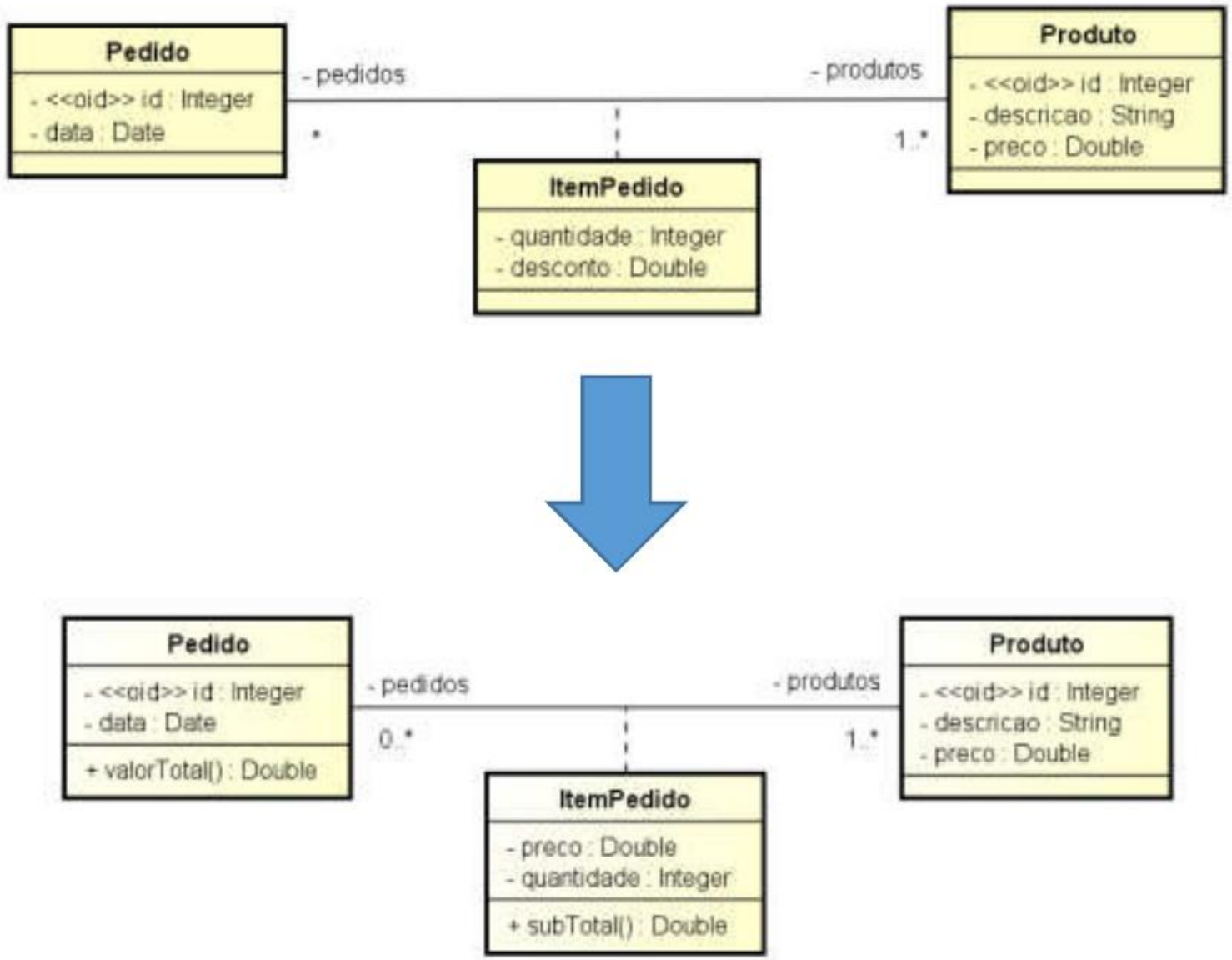
Preso à tecnologia: **Java**

```
public class Produto {
	private Integer id;
	private String descricao;
	private Double preco;

	public Produto(Integer id, String descricao, Double preco) {
		this.id = id;
		this.descricao = descricao;
		this.preco = preco;
	}

	(...)
```

Paradigma orientado a objetos



3.3 Implementação

- A fase de implementação é onde o design do sistema é traduzido em código executável.
- Essa etapa envolve a criação real do software com base nas decisões de design previamente estabelecidas.

3.4 Testes

- A fase de testes é uma etapa crítica no ciclo de vida do desenvolvimento de software, dedicada a verificar se o sistema atende aos requisitos estabelecidos, se comporta conforme esperado e se está livre de defeitos.
- Esta fase é subdividida em diferentes tipos de testes para abordar aspectos específicos do sistema.

Principais Tipos de Testes

- Testes Unitários
- Testes de Integração
- Testes de Sistema
- Testes de Aceitação do Usuário
- Testes de Desempenho
- Testes de Segurança

3.5 Manutenção

- A fase de manutenção é uma parte essencial do ciclo de vida do desenvolvimento de software, dedicada a melhorar, corrigir defeitos e adaptar o sistema após sua entrega.
- Esta fase reconhece que as necessidades dos usuários e o ambiente operacional podem mudar ao longo do tempo, exigindo ajustes contínuos no software.

Principais Tipos de Manutenção

- Manutenção Corretiva
- Manutenção Adaptativa
- Manutenção Evolutiva
- Manutenção Preventiva

Modelos de Processos de Desenvolvimento

- Os Modelos de Processos de Desenvolvimento de Software são abordagens ou estruturas que definem a maneira como o desenvolvimento de software é organizado, planejado e executado.
- Eles fornecem uma estrutura para orientar as atividades de desenvolvimento desde a concepção do projeto até a entrega do software.

Modelos de Processos de Desenvolvimento

- Existem vários modelos de processos de desenvolvimento de software, e cada um tem suas características, vantagens e desvantagens.

- Aqui estão alguns dos modelos mais comuns.

Modelo Cascata

- Abordagem linear e sequencial.
- As fases são executadas em ordem: requisitos, projeto, implementação, teste, instalação e manutenção.
- É fácil de entender e usar, mas pode ser inflexível quando há mudanças nos requisitos.
- Falta de flexibilidade.

Modelo Incremental

- Divide o projeto em pequenos incrementos ou partes.
- Cada incremento passa por todas as fases do modelo cascata.
- Cada incremento adiciona funcionalidades ao software.
- Permite entregas mais rápidas de partes funcionais.
- Gerenciamento cuidadoso das integrações

Modelo Espiral

- Combina elementos do modelo cascata e prototipagem.
- Ciclos iterativos e incrementais.
- Enfoca a gestão de riscos ao longo do desenvolvimento.
- Bem adaptado para projetos grandes e complexos.

Modelo V-Model (Modelo em V)

Relacionado ao modelo cascata.

Cada fase de desenvolvimento tem uma fase de teste correspondente.

As atividades de teste são planejadas paralelamente às atividades de desenvolvimento.

Desenvolvimento Rápido de Aplicações (RAD)

- Centrado na rápida entrega de protótipos.
- Envolve iterações e feedback contínuo do cliente.
- Ênfase na colaboração e comunicação eficaz.

Modelo Ágil

- Enfatiza a colaboração, adaptação a mudanças e entrega contínua.
- Ciclos curtos de desenvolvimento (**sprints**) com feedback constante do cliente.
- Valoriza indivíduos e interações mais do que processos e ferramentas.
- Reuniões regulares, como o Daily Standup, são comuns.
- Scrum e Kanban

DevOps (Desenvolvimento e Operações)

Integração estreita entre desenvolvimento e operações.

Automatiza processos de desenvolvimento, teste e implementação.

Busca melhorar a colaboração e eficiência entre equipes.

Entrega Contínua (CI/CD)

Modelo Iterativo

- Baseado em repetições (iterações) do ciclo de vida do desenvolvimento.
- Cada iteração resulta em uma versão mais completa do software.
- Permite adaptação a mudanças nos requisitos.

Modelos de Processos de Desenvolvimento

Cada modelo tem suas próprias características, e a escolha do modelo de processo depende das necessidades específicas do projeto, requisitos do cliente, prazos e recursos disponíveis.

Muitas equipes optam por combinar elementos de diferentes modelos para criar uma abordagem personalizada que atenda às suas necessidades.

Exercício de Fixação

Atividade em Grupo
“Estudo de Caso”

Os alunos escolherão um modelo de processo e criarão um cronograma de desenvolvimento hipotético para um sistema específico.

Discussão em Grupo

Compartilhamento dos cronogramas desenvolvidos pelos alunos.

Discussão sobre os desafios enfrentados durante o exercício.

Avaliação

- Os alunos serão avaliados com base na participação no exercício e na compreensão dos conceitos discutidos durante a aula.

Exemplo 1

- Exemplo 1 – Estudo de Caso: Desenvolvimento de um Sistema de Gerenciamento de Tarefas usando o Modelo Cascata

- Objetivo
  - Desenvolver um Sistema de Gerenciamento de Tarefas para equipes de projetos colaborativas, permitindo a criação, atribuição e monitoramento de tarefas.

- Modelo de Processo Escolhido: Modelo Cascata

Exemplo 2

- Exemplo 2 – Estudo de Caso: Desenvolvimento de um Sistema de Gerenciamento de Biblioteca usando o Modelo Iterativo

- Objetivo
  - Desenvolver um Sistema de Gerenciamento de Biblioteca para uma instituição educacional, proporcionando aos usuários a capacidade de catalogar, pesquisar e gerenciar recursos de biblioteca de maneira eficiente.

- Modelo de Processo Escolhido: Modelo Iterativo