**Aula 14 – Modelo de desenvolvimento de Software – Modelo Cascata**

---

O **Modelo Cascata** é um processo linear de desenvolvimento de software, no qual cada fase precisa ser concluída antes que a próxima seja iniciada. Criado por *Winston W. Royce (1970),* ele é composto por cinco fases principais: **Requisitos, Design, Implementação, Verificação e Manutenção**.

A primeira fase é a de **Levantamento de Requisitos**. Nessa etapa, o objetivo é entender tudo o que o sistema precisa fazer. Isso envolve conversar com os usuários, gestores, clientes e todos os envolvidos no projeto. É como montar uma lista detalhada das funcionalidades desejadas.
Por exemplo, ao criar um sistema escolar, é necessário saber se os professores querem lançar notas, se a secretaria deseja controlar faltas e se a direção precisa de relatórios.
Um erro nessa fase pode comprometer o projeto inteiro, pois mudar algo depois é bem mais difícil e caro.

Depois vem a fase de **Design do Sistema**, onde se planeja como o sistema será construído. Aqui são definidos a arquitetura, o banco de dados, as telas, as interações entre os módulos, entre outros detalhes técnicos.
Se na fase anterior foi decidido que será necessário um módulo de notas, nesta fase será desenhado como ele vai funcionar, onde os dados serão salvos e como será a aparência da tela.
É como desenhar a planta de uma casa antes de começar a construí-la. Um design mal feito pode gerar retrabalho e dificuldades técnicas na hora de codificar.

Com o projeto bem definido, o time entra na fase de **Implementação**, que é quando os programadores começam a escrever o código do sistema. Todo o planejamento feito nas fases anteriores agora ganha forma em linguagem de programação.
Por exemplo, o módulo de cadastro de alunos começa a funcionar, assim como a parte de geração de boletins ou controle de presença.

Essa costuma ser a fase mais longa, porque envolve muitos detalhes técnicos. Qualquer erro na lógica do código ou na compreensão do design pode impactar negativamente o resultado final.

Após o sistema ser desenvolvido, é necessário testá-lo, o que nos leva à fase de **Verificação**.
Nessa etapa, o sistema passa por vários testes para garantir que tudo funciona corretamente.
Os testes vão desde unidades pequenas do sistema até a verificação do funcionamento geral.
Por exemplo, é testado se o cálculo da média do aluno está correto, se as informações estão sendo salvas corretamente, e se o sistema responde bem quando é utilizado por várias pessoas ao mesmo tempo.
Também é comum fazer testes com usuários reais, como professores e administradores da escola, para verificar se o sistema atende às suas necessidades.
Por fim, temos a fase de **Manutenção**, que acontece após a entrega do sistema. Mesmo com todos os testes, sempre podem surgir ajustes, correções ou novas necessidades.
Às vezes o sistema precisa ser adaptado a novas regras do governo, ou talvez os usuários percebam que uma funcionalidade poderia ser melhor.
A manutenção pode ser corretiva, quando há um erro a ser resolvido; adaptativa, quando há mudanças externas que exigem ajustes; ou evolutiva, quando se deseja adicionar novas funcionalidades.
O modelo cascata tem como vantagem a clareza e a organização, sendo ideal para projetos com requisitos bem definidos desde o início. No entanto, uma de suas principais desvantagens é a dificuldade de lidar com mudanças no meio do caminho. Por isso, ele é mais indicado para projetos estáveis, como sistemas governamentais ou de controle embarcado, e menos eficaz para projetos ágeis, como aplicativos móveis ou startups, onde tudo muda rapidamente.

**VANTAGENS**

- Clareza no processo
- Boa documentação

- Ideal para projetos pequenos e estáveis

---

**DESVANTAGENS**

- Pouca flexibilidade para mudanças
- Testes tardios (só no final)
- Feedback do usuário é tardio

---

**Quando usar?**

✓ Projetos com **requisitos estáveis**

✓ Sistemas governamentais ou **críticos**, como **controle de aviões**

---

**Quando NÃO usar?**

✗ Projetos com **requisitos variáveis**

✗ Ambientes **ágeis** ou que exigem **respostas rápidas ao usuário**

---

**Exemplo de Aplicação – Projeto Real**

**Projeto:** Aplicativo de Delivery para Restaurantes

**Fase Requisitos:** cadastro de cardápio, formas de pagamento

**Fase Design:** interface mobile, banco de pedidos

**Fase Codificação:** integração com API de pagamento

**Fase Teste:** simulação de pedidos

**Fase Manutenção:** incluir nova forma de entrega via drone