Processos Prescritivos de Software

Os **processos prescritivos** são modelos estruturados de desenvolvimento de software, nos quais cada etapa pode ser observada e validada. Diferente dos **modelos descritivos**, que mostram como o processo ocorre na prática, os **modelos prescritivos** indicam como ele **deveria ser executado**, servindo como uma recomendação que pode ser ajustada conforme a necessidade da equipe ou empresa.

Esses modelos incluem diretrizes para o planejamento estratégico e desenvolvimento de software, especificando tarefas e o fluxo de trabalho.

Principais Modelos Prescritivos

1. **Modelo em Cascata**
2. **Modelos Evolucionários:**
    - Prototipagem
    - Espiral
    - Concorrente
3. **Modelos Incrementais:**
    - RAD (Rapid Application Development)
    - Incremental
4. **Modelo baseado em Componentes**
5. **Modelo de Métodos Formais**
6. **Processo Unificado**

Cada um desses modelos tem características específicas e pode ser escolhido conforme o contexto do projeto.

Modelo em Cascata

O **Modelo em Cascata**, ou **ciclo de vida clássico**, é um processo sequencial de desenvolvimento de software que segue etapas definidas:

1. **Especificação de Requisitos**
2. **Planejamento**
3. **Modelagem**
4. **Construção**
5. **Implantação**
6. **Manutenção**

Esse modelo é um dos mais antigos na engenharia de software, mas é amplamente criticado porque raramente um projeto segue um fluxo linear. A principal limitação é a dificuldade dos clientes em definir todos os requisitos no início, o que pode gerar retrabalho, aumento de custos e atrasos no projeto.

Modelos Evolucionários de Processo

Os **modelos evolucionários** permitem o desenvolvimento iterativo do software, liberando versões cada vez mais completas ao longo do tempo. São ideais quando:

- O prazo para desenvolvimento é curto.
- É necessário lançar rapidamente uma versão inicial para o mercado.
- O software evolui com base no feedback dos usuários.

Os principais modelos evolucionários são:

1. **Prototipação**
2. **Modelo Espiral**
3. **Modelo Concorrente**

Prototipação

A **prototipação** consiste em criar uma versão inicial (protótipo) do software para validar requisitos antes do desenvolvimento completo.
✅ **Vantagens**:

- Reduz custos e erros ao envolver o usuário desde o início.
- Melhora a comunicação entre cliente e desenvolvedores.
- Aumenta a satisfação ao priorizar usabilidade.

⚠ **Riscos**:

- O cliente pode acreditar que o protótipo é a versão final.
- O desenvolvedor pode utilizar soluções temporárias inadequadas para o produto final.
- Pode comprometer a qualidade e a manutenibilidade do software.

A **prototipação** é útil, mas deve ser usada com clareza sobre seu propósito: definir requisitos, e não servir como produto final.

Modelo Espiral

Criado por **Barry Boehm (1988)**, combina as melhores características do **modelo em cascata** e da **prototipação**, acrescentando a **análise de riscos**.

🌀 **Principais características:**

- Iterativo e focado na identificação e mitigação de riscos.
- Cada fase começa com um objetivo e termina com uma revisão do progresso.
- A primeira iteração pode gerar uma especificação ou protótipo.
- Permite a criação de versões incrementais melhoradas ao longo do tempo.

- Indicado para **softwares de grande porte**, pois permite adaptações constantes.

✅ **Vantagens:**

- Detecta problemas críticos cedo.
- Flexível para mudanças.
- Oferece visibilidade progressiva das necessidades do projeto.

⚠ **Desvantagens:**

- Exige **experiência para análise de riscos**.
- Pode ser difícil convencer clientes sobre a abordagem iterativa.

Modelo de Desenvolvimento Concorrente

Também chamado de **engenharia concorrente**, permite que diversas **atividades ocorram em paralelo**, com estados interdependentes.

🔄 **Principais características:**

- As atividades **não seguem uma sequência fixa**, como no modelo em cascata.
- **Estados dinâmicos**, podendo estar em desenvolvimento, aguardando modificações, sob inspeção, etc.
- Definição de eventos que **disparam mudanças de estado** nas atividades.
- Indicado para **projetos complexos**, onde diferentes partes do software são desenvolvidas simultaneamente.

✅ **Vantagens:**

- Maior flexibilidade e adaptação a mudanças contínuas.
- Melhora a **visibilidade do status do projeto**.
- Adequado para **desenvolvimento ágil e software moderno**.

⚠ **Desvantagens:**

- Planejamento pode ser mais complexo.
- A ênfase na flexibilidade pode comprometer o **controle da qualidade**.

🔹 **Resumo geral:** O **Modelo Espiral** é ideal para projetos grandes e críticos, onde a mitigação de riscos é essencial. O **Modelo Concorrente** se adapta bem a desenvolvimentos modernos, permitindo que múltiplas atividades aconteçam simultaneamente.